Toxic Comment Classification and Unintended Bias

Chady Ben Hamida
chadybh@stanford.edu

Victoria Ge
vge@stanford.edu

Nolan Miranda
mirandan@stanford.edu

June 12, 2019

1 Introduction

Discussion on online fora can often devolve into abuse and harassment due to the anonymity of users. Internet users find it much easier to propagate harmful stereotypes and toxic commentary in comment sections [1]. Platforms often struggle to facilitate conversation effectively forcing many communities to shut down user comments. Some developers like Conversation AI at Alphabet are working on tools to help improve online behaviors and minimize toxicity [2]. However, many rudimentary models learn to associate certain gender identities with toxicity, thereby flagging comments such as "that's so gay" together with "as a gay woman." The motivation for our project is to build a model that can detect toxic comments and find the bias with respect to the mention of select identities.

2 Related Work

Word embeddings have become a predominant technique in natural language processing. J. Pennington, R. Socher, and C. Manning developed the GloVe model for distributed word representation [3] which is a mapping of words into a semantic space where vectors between words are meaningful. It is trained on aggregated global word-word co-occurrence statistics from a corpus. Our project uses the GloVe word embeddings, as well as those of fastText, developed by Facebook Research [4].

Research in fairness of algorithms has proposed a range of definitions for metrics of its evaluation. Menon A. and Williamson R. studied the trade off between accuracy and fairness in binary classification [5]. Alex Beutel, Jilin Chen, Zhe Zhao, Ed H. Chi proposed nuanced metrics for measuring unintended bias, that measure the difference between the true positive rates between the subgroup and the overall background [6]. Borkan D., Dixon L. et al expanded on these methods and applied them to text classification. In their paper, they introduced 3 AUC-based metrics: Subgroup AUC, BPSN (background positive, subgroup negative) AUC and BNSP AUC [7].

3 Dataset and Features

3.1 Data

Our data source is a Kaggle dataset [8] that consists of $1,804,874$ comments collected in 2017 from the Civil Comments platform upon its closure. Each comment is scored by multiple graders from "not toxic" to "very toxic" and the results are aggregated to form a classification label from 0.0 to 1.0. Each comment was seen by at least 10 annotations, and some many more (up to thousands). For the purposes of our project, a score of 0.5 or above is automatically labeled as toxic. Each comment also contains a binarized hand label if the comment contains a mention of certain identity groups (e.g. `intellectual_learning_ disability`). We only focused on identities with more than

500 examples in the test set: `'male'`, `'female'`, `'homosexual_gay_or_lesbian'`, `'christian'`, `'jewish'`, `'muslim'`, `'black'`, `'white'`, and `'psychiatric_or_mental_illness'`.

The data preprocessing consisted of text tokenization and normalization which varied slightly when processing features for each model. The quality of the comments was low as expected of internet comments. The raw vocabulary size was $\sim 15,000$ compared to the $\sim 10,000$ estimated vocabulary size of "highly educated native English speakers" [9]. This differential may have been due to the diverse nature of specific conversations, but also internet specific vocabulary and typos. To deal with this, we took the $10,000$ most common words in the comments and only extracted features on this smaller vocabulary. Also, for our CNN, we padded all the comments they all contained 250 words in order for all the vectors to have the same length. The embeddings vectors were padded with 0's.

For the training process of each model, we split the data for training and validation in the ratio 80:20, so that we had $1,443,899$ training points and $360,975$ validation points.

3.2 Features

We looked at a variety of language processing features to create vector representations and them applied them to our models. We had vocabulary $V = \{w_i : 1 \leq i \leq 10,000\}$ where w_i are the top $10,000$ unique words. We also had a corpus of comments $C = \{c_i : 1 \leq i \leq 1,443,899\}$ where each comment $c_i = w_{c_i[1]} w_{c_i[2]} w_{c_i[3]} \ldots$ is represented as a string of words in order.

- **N-gram bag-of-words.** This method vectorized each comment by creating a "vector space" of the entire vocabulary V where each dimension $v_i \in V$ had an associated word v_i. Thus, a sample comment $c = w_1 w_{250} w_{3405} w_3 w_1 \ldots$ was represented as $c = (2, 0, 1, \ldots)$ where each w_i entry is the term frequency of the word w_i in the comment. We found this with the unigram and the bigram models where each bigram $(w_i, w_j) \in V \times V$. Note that the unigram model does not preserve the order of the words in the original sentences.

- **Tf-idf.** We quickly found that term frequencies were not the best representation for the text since extremely common words like "the", "a", "and" were the most represented. Since having a high raw count does not necessarily mean that the corresponding word is more important in the classification task, we used tf-idf. This embedding score increased each v_i proportionally to the number of times the word w_i appears in the comment c_i and is offset by the number of comments in C that contain the w_i, which helps to adjust for the fact that some words appear more frequently in general. Note that this embedding also does not preserver order.

- **GloVe/fastText.** These are pretrained embeddings that maps words to the $\mathbb{R}^d$ for $d = 50, 100, 200, 300$. These vectors are much more compact than the term frequency vectors (which has $d = 10,000$) and the spatial mapping captures relation of words. The two embeddings used were Stanford's GloVe and Facebook's fastText [3] [4]. Both these embeddings were created from neural networks and trained on internet text. Thus, the embedding for a comment would be a 1D vector where the first d entries are the pretrained embeddings for the first word in the comment, the second d entries are the pretrained embeddings for the second word in the comment, and so on.

We also appended the binarized identity mention data to the end of the vector representations.

4 Preliminary Models

4.1 Naive Bayes

We applied the multinomial Bernoulli event model with the n-gram bag-of-words count features. Thus, we classified each validation comment as $\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$ where we MAP

estimated $P(y)$ and $P(x_i \mid y)$. We did this for both $n = 1, 2$ and found the results similar, but the $n = 2$ case much more computationally expensive. Laplace smoothing was used in the model.

4.2 Logistic Regression

We implemented a logistic regression with the tf-idf features next. We aimed to minimize cross-entropy with L2 regularization. With regularization parameter $\lambda/2n$ and predicted probability p we had loss function

$$\mathcal{L} = -\frac{1}{n} \underbrace{\sum_i \left[y_i \log p + (1 - y_i) \log (1 - p) \right]}_{\text{cross-entropy}} + \underbrace{\frac{\lambda}{2n} \sum_\theta]\theta^2}_{\text{L2 reg}} .$$

5 Convolutional Neural Networks

We used Keras with a TensorFlow backend to implement a one-dimensional convolutional neural network [10] [11]. The general architecture of the networks we built are as follows:

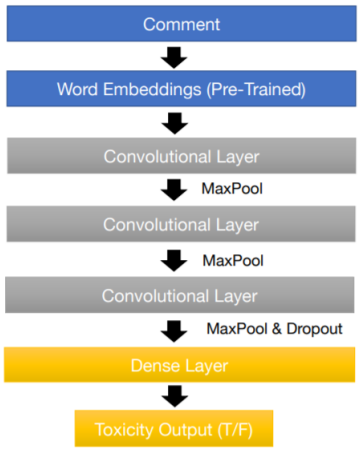

Figure 1: Architecture of our CNN.

We first fed the comment into the pretrained word embeddings from the GloVe 50, 100, and 200 dimension vectors as well as the fastText 300 dimensional vectors (each a separatae trial). We tried concatenating the two embeddings and running this through out model, but it proved much too computationally expensive with trivial increase in accuracy. Note that this is there 90% of the model complexity resides.

We then fed these word embeddings into a three-layer CNN. We experimented with various architectures including adding and taking out layers, varying the filter size, and different max pooling dimensions. Our final model was constructed as three one-dimensional convolutional layers interlaced with max pooling layers. The MaxPooling serves to reduce dimension, speed up run time, and mitigate overfitting. Between the layers, we used ReLU activation over sigmoid to cut down on execution time since the vanishing gradient was computationally expensive. Our final max pooled convolutional layer employs dropout into the dense layer in order to regularize and further mitigate overfitting [12].

We chose the categorical cross-entropy function for our loss, also known as softmax loss [13]. If C is the number of classes and x_i denotes the current training example, then the loss function we

aimed to minimize is given as

$$\mathcal{L} = -\sum_{i}^{C} x_i \log\left(f(s)_i\right)$$

where

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}.$$

The final output from the dense layer was a toxicity score given from 0.0 to 1.0 with the goal of predicting the toxicity scores provided in the dataset. The results were immediately binarized so 0.5 or greater indicated a `toxic` comment.

6 Results and Discussion

We ran our CNNs for 7 epochs each with a batch size of 128. We found that the accuracy started to converge in the thousanths place around epoch 5, and decided to cut our model off at 7 epochs to mitigate overfitting. We used a batch size of 128, as we found that larger batch sizes made results more stable.

Model	Embeddings	Accuracy (validation)
Naïve Bayes	(Bag-of-words)	0.9202
Logistic Reg	(TF-IDF)	0.9474
CNN	GloVe 50d	0.9390
CNN	GloVe 100d	0.9450
CNN	fastText 300d	0.9484

Figure 2: The prediction accuracies for the models we used.

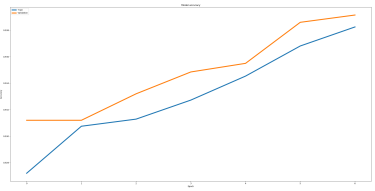

Figure 3: CNN accuracy on train (blue) and validation (orange) sets for 7 epochs.

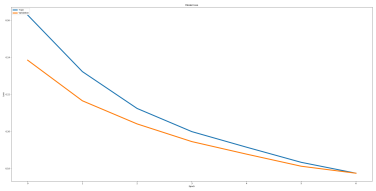

Figure 4: CNN loss on train (blue) and validation (orange) sets for 7 epochs.

Our best model was the fastText CNN, which reached a prediction accuracy of 94.84%. This model beat out the others narrowly, and while the model did improve upon Naive Bayes considerably, the runtime was around 20 minutes per epoch (as compared to only 7 minutes per epoch for the 100-dimensional GloVe embeddings). There was no significant difference between the two sets of pre-trained embeddings, and because the time increase was linear in the number of dimensions, it is not worth it to increase the dimensionality of the word embeddings. Thus, we did a decent job of preventing overfitting through our MaxPooling layers and dropout, but improvements in accuracy were not very significant over the simpler models.

Logistic regression performed very well on the data, reaching a higher predictive accuracy than a few of our CNNs. Logistic regression also ran about 10 times faster than the 50-dimensional GloVe

CNN, which was our fastest neural network model. This may have been due to the fact that logistic regression performs incredibly well when the data has a clear separation. In inspecting out data, comments often had a toxicity score of 0.0 or a score close to 1.0 since many of the graders either deemed a comment "not toxic" or "toxic" (compared to "hard to say" and other middle ground labels). Thus, logistic regression was able to find a comparable linear separation in our data set to the CNN classification.

6.1 Unintended Bias Analysis

identity (id)	bnsp_auc	bpsn_auc	id_auc	id_size
black	0.961	0.747	0.801	3025
white	0.961	0.757	0.807	5047
homosexual_gay...	0.957	0.771	0.807	2210
muslim	0.954	0.802	0.830	4204
female	0.936	0.868	0.865	10804
...mental_illness	0.957	0.835	0.869	954
jewish	0.941	0.854	0.871	1570

Figure 5: Bias metric scores for the GloVe 50d CNN.

We used the AUC bias metrics to determine the unwanted bias in our CNN model. The subgroup AUC metric restricts the data set to just those examples that mention a particular subgroup; a low value for this indicates that the model did a poor job differentiating between toxic and non-toxic comments that mention that particular subgroup. The BPSN (background positive, subgroup negative) score measures how likely a model is to predict higher toxicity for nontoxic comments mentioning the identity, and the BNSP (background negative, subgroup positive) measures how likely a model is to fail to predict toxicity for toxic comments mentioning the identity.

These metrics provide insight into why our model did not perform better. While our CNN earned a high BNSP score, showing that it did a good job of accurately identifying true toxic comments, the BPSN scores were low, meaning that our model predicted higher toxicity scores than desired for actual non-toxic examples. We see particularly bad conflation in the subgroup scores for `'black'`, `'white'`, and `'homosexual_gay_or_lesbian'`, partially due to the fact that these labels comprise the majority of the toxic comments, which this diminishes model accuracy considerably.

7 Conclusion and Future

We have constructed a few models that classify toxicity levels in comments as well as measured the unintended bias present through BNSP, BPSN, and Subgroup AUC scores. Our fastText CNN with 300-dimensional word embeddings achieved an accuracy of 94.84% on the validation set, and we hypothesize that the tendency to conflate toxicity with mention of gender identity results in an accuracy drop. This hypothesis was suggested by the BNSP/BPSN scores analyzed above.

In the future, we would like to utilize the unintended bias data we calculated to penalize negative identity biases and improve our hyperparameters. Given more time, we also would have tried to implement translation in our preprocessing. If we were able to translate comments to different languages then back to English, some of the more idiomatic language would have been removed, and the comments would have been more machine understandable. We also would experiment with a completely different neural network architecture, such as using a recurrent neural network with BLSTM (bidirectional long short-term memory) [14].

Link to GitHub Repository

`https://github.com/nmiranda98/toxicity/tree/master`

Contributions

- Chady Ben Hamida worked on preprocessing, wrote the updated code for logistic regression, assisted in writing the CNN, helped with the poster and paper, and commented the code.
- Victoria Ge worked on preprocessing, wrote the updated code for Naive Bayes, assisted in writing the CNN, and helped with the poster and paper.
- Nolan Miranda worked on preprocessing, assisted in writing the CNN, tuned model hyperparameters, and helped with the poster and paper.

References

[1] Barlett, C. P. "Anonymously hurting others online: The effect of anonymity on cyberbullying frequency." *Psychology of Popular Media Culture*, 4(2), 70-79. 2015.

[2] conversationai. "Conversation AI," GitHub repository. 2017. https://conversationai.github.io/

[3] Jeffrey Pennington, Richard Socher and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." *ACL Anthology* 1532–1543. 2014.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov.fastText. *facebook Research.* 2016.

[5] Aditya Krishna Menon, Robert C. Williamson. "The Cost of Fairness in Binary Classification." *Proceedings of Machine Learning Research*, PMLR 81:107-118. 2019.

[6] Alex Beutel, Jilin Chen, Zhe Zhao, Ed H. Chi. "Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations." 2017. `arXiv:1707.00075`

[7] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain and Lucy Vasserman. "Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification." 2019. *arXiv:1903.04561.*

[8] Kaggle. "Jigsaw Unintended Bias in Toxicity Classification." 2019. https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data

[9] Marc Brysbaert, Michael Stevens, Pawel Mandera, and Emmanuel Keuleers. "How Many Words Do We Know? Practical Estimates of Vocabulary Size Dependent on Word Definition, the Degree of Language Input and the Participant's Age." *Frontiers in Psychology.* 2019.

[10] Keras Documentation. keras.io. 2019.

[11] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning." *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16).* 2016.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ilya Sutskever. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research.* 2014.

[13] Ràul Gomez. "Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names," GitHub repository. 2018.

[14] Yu Zhou, Vikram Ramanarayanan, David Suendermann-Oeft, Xinhao Wang, Klaus Zechner, Lei Chen, Jidong Tao, Aliaksei Ivanou, Yao Qian. "Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech." *IEEE Xplore*. 338-345. 2015.