Building Unbiased Comment Toxicity Classification Model with Natural Language Processing

Louise Qianying Huang, Mi Jeremy Yu
Department of Statistics
Stanford University
{qyhuang, miyu1996}@stanford.edu

Abstract

In this project, we tackle the Jigsaw Unintended Bias in Toxicity Classification Challenge. We built an unbiased classification model to identify toxicity in online comments using non-neural methods, i.e. Logistic Regression and Naive Bayes, and neural methods, i.e. RNN and CNN. Our best-performing model achieves 0.7785 generalized AUC on the test set.

1 Introduction

Machine learning and deep learning techniques have achieved state-of-the-art results in many natural language processing tasks, such as Question Answering(1) and Neural Machine Translation(2). However, natural language often contains biases and stereotypes, and models that are trained on such corpus will inevitably learn and intensify those biases. Jigsaw Unintended Bias in Toxicity Classification Challenge was proposed originally as a Kaggle competition[1]. In the previous iteration of the challenge[2], the data contain unconscious biases as certain identities are overwhelmingly referred to in offensive ways. Unfortunately, the model internalizes such languages biases and learns to predict toxicity based on whether these identities are mentioned.

Our project aims to build an unbiased classification model to identify toxicity in online comments using both non-neural and neural techniques. We not only aim at building a model that can accurately classifies whether a comment is toxic or not, but also seek to combat the unintended language bias with respect to mentions of identities in the training data.

2 Related Works

Early literature in automatic online abusive speech detection often involves feature-based classical machine learning techniques such as Support Vector Machine(3). Simple surface features such as Bag of Words, token and character n-grams are heavily used (4)(5). Linguistic features such as length of comment in tokens, average length of word, and number of punctuations are often used to explicitly look for inflammatory words (6). Although these methods are proved to be highly predictive, extracting knowledge-based features can be an ad hoc labor-intensive procedure, which weakens the generalizability power of these classical ML models(7).

End-to-end deep learning models which utilize deep neural networks are proved to outperform the aforementioned classical methods, without then necessity of extensive feature engineering(8). Popular DL architecture that is applicable to toxic comment detection includes Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). Pavlopoulos et. al found in (9) that attention mechanism is also beneficial, as it can highlight suspicious words for free, without including highlighted words in the training data.

3 Dataset and Attributes

The data was collected from the Civil Comments platform and annotated by the Jigsaw/Conversation AI team. The original training set has 1804874 examples of public comments. The toxicity of the comment is quantified as toxicity label ranging from 0 to 1, where 0 represents non-toxic at all and 1 represents severely toxic. Each example has the overall toxicity label, as well as five toxicity sub-type labels, i.e. severe toxicity, obscene, identity attack, insult, and threat. Additionally, each example is labelled with a variety of identity attributes, representing the identities that are mentioned in the comment. [3]

[1] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[3] To avoid confusion, we use attributes to refer to the columns of the original dataset, and we use features to refer to the extracted features in our non-neural methods.

Since the original test set does not have the identity attributes to calculate generalized , for the purpose of fast testing, we held out 5% of the original training set as our new test set. Therefore, the new training dataset has 1715241 comments, and the new test dataset has 89633 comments. Also, 10% of the training set was held out as the validation set for the purposes of hyperparameters tuning and model checking.

4 Method

4.1 Non-Neural Methods

4.1.1 Feature Extraction

We utilized the Bag of Words to extract numerical features from the raw comment texts. The feature extraction process consists of tokenizing, counting, and normalizing, with the scikitlearn package. First, we tokenized the strings and assigned an integer id to each possible token, with white-spaces and punctuation as token separators. We combined 1-gram and 2-gram word tokens, and 1-gram up to 6-gram character tokens. Character tokens are particularly informative for toxicity detection (10). We counted the occurrence of the tokens in each document. We used sublinear scaled frequency defined as following:

$$wf_{t,d} = \begin{cases} 1 + logtf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $tf_{t,d}$ is the raw count of a term in a comment. This normalization is used because it seems unlikely that the number of occurrences of a term truly carry the same amount of the significance, and this is a common modification that is smoother than raw count. In the end, we normalized and weighted with diminishing importance tokens that occur in the majority of samples by multiply the inverse document frequency. The inverse document frequency is defined as:

$$idf(t, D) = log\frac{N}{|\{d \in D : t \in d\}|}$$

Since our corpus is enormous, we selected the top 20000 word features and the top 30000 character features as our final features.

4.1.2 Logistic Regression

Logistic regression classifier has the form

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

which gives us the probability of the comment being toxic. We used L2 regularization and chose SAG solver as the optimizer.

4.1.3 Naive Bayes

Multinomial event model is under the assumption that $x^{(i)}$ are conditionally independent given y. The likelihood for multinomial event model is

$$l(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^{n}(\prod_{j=1}^{d_i} p(x_j^{(i)}|y; \phi_k|y = 0, \phi_{k|y=1}))p(y^{(i)}; \phi_y)$$

where here $x^{(i)}$ are the features we extracted. We can estimate the parameters by using maximum likelihood estimators.

4.2 Neural Methods

4.2.1 Embedding Layer

The task of the embedding layer is to map each word token into a more dense representaion using character level and word level features. For this project, we investigated word embeddings GloVe300D (11) and fastText300D (12), where both are pretrained on Common Crawl. We also implemented character level embedding layer introduced in the paper (13), using 1D-CNN and a two-layer Highway Network(14). The output of this layer will be $E \in R^{T \times d}$, where T was the length of the sentence, and d was either 300 or 600, depending on whether we concatenated representations with different embeddings.

4.2.2 Gated Recurrent Neural Networks

The two basic building recurrent blocks we used are Long-Short-Term-Memory Unit (LSTM) and Gated Recurrent Unit (GRU). The benefit of using the below RNN structure is to memorize the information from the early stage while dealing with long sequential input.

GRU

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned}
$$

LSTM

$$
\begin{aligned}
\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} &= \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot \tanh(c_t)
\end{aligned}
$$

Figure 1: Mathematical Representation of GRU and LSTM.

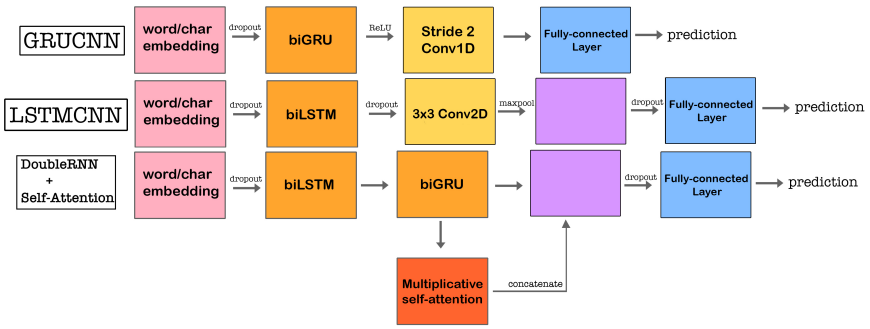

Figure 2: Architecture of Neural Models

4.3 GRUCNN & LSTMCNN

GRUCNN will utilize a bidirectional GRU and a Conv1d over the sequence length. We have the following structure:

$$
\begin{aligned}
H_t &= BiGRU(E_t) \\
C &= Conv1d(H_t) \\
C_{max} &= Maxpool(C) \\
O &= Linear(C_{max})
\end{aligned}
$$

where $H_t \in R^{2\times h}$. Conv1D over sequence length gives us $C \in R^{t\times h}$, where $t = \lceil T/2 \rceil + 1$. $C_{max} \in R^h$. We hope the CNN will capture the local structure of consecutive words in our text. The fully-connected layer collects all the information and transform the output into the label shape.

LSTMCNN was purposed by Zhou et al. in (15). It follows the similar idea as the previous structure. Here we replaced bidirectional GRU with bidirectional LSTM, and used Conv2D over sequence length and hidden size of the output of biLSTM, followed by a dropout layer.

4.4 doubleRNN

DoubleRNN uses two back-to-back bidirectional RNN layers to process the sequential information of a comment. Let H_t denote the output and H_t^{last} denote the last returned state of biGRU.

$$
\begin{aligned}
H_t^{max} &= Maxpool(H_t) \\
H_t^{avg} &= Avgpool(H_t) \\
C &= Concat(H_t^{last}, H_t^{max}, H_t^{avg}) \\
\tilde{C} &= Dropout(C) \\
O &= Linear(\tilde{C})
\end{aligned}
$$

where the average pooling and max pooling are over the sentence length.

4.5 Multiplicative Self-attention

Inspired by R-Net(16), we passed the output of the CNN layer or the output of the second RNN layer to a multiplicative self-attention layer, to extract and add importance to informative words that might be relevant to classification. We

have the following structure:

$$
\begin{aligned}
c_t &= Softmax(H_t^T W_h H_t) \\
A_t &= c_t^T H_t \\
O &= Concat(H_t, A_t, H_t * A_t) \\
C &= ReLU(WO + b)
\end{aligned}
$$

5 Experiments and Discussion

5.1 Data Preprocessing

Unlike traditional spam classification, comments are often written in casual English with a lot of out-of-dictionary vocabularies, spellings, and also typos. We decided to remove all the spaces, special characters, and numerical symbols as we assume that these are not going to have significant impacts on the toxicity of the comment. We also changed all characters to the lower case. We also removed all stop words and changed abbreviations to the full forms. We decided to keep the words in their original form instead of changing to their stem words, since such changes might lose information about the toxicity of a word. We also corrected obvious typos in the corpus.

5.2 Data Augmentation

Data augmentation was proved to be effective as a form of regularization to reduce overfitting. We augmented our training data by translating a comment to intermediate language and translate it back to English, with the TextBlob library which works using Google Translate API(17). Therefore, we obtained three extending training dataset, where the intermediate languages are Spanish, German, and French.

5.3 Hyperparameters

We tune the hyperparameters with our held-out validation set. All the neural methods are trained using the Adam optimizer(18), with learning rate 1e-3, betas (0.9, 0.98), eps 1e-9, weight decay 5e-5, and gradient clipping at with maximum gradient norm 5. Learning rate was decayed by a factor of 0.2 when the trainig loss was stuck at plateau for 2 epochs. Batch size is 640 for training stability and most models were trained for 15 epochs. We used a dropout probability of 0.3 for the embedding layer, and 0.2 for all other dropout layers.

5.4 Result

The evaluation metric is generalized AUC as detailed in (19). Generalized AUC combines overall AUC, Subgroup AUC, Background Positive Subgroup Negative AUC, and Background Negative Subgroup Positive AUC to balance overall performance with various aspects of unintended bias. Generalized AUC is defined as the following in this task:

$$
GeneralizedAUC = w_0 AUC_{overall} + \sum_{a=1}^{A} w_a (\frac{1}{N} \sum_{s=1}^{N} m_{s,a})^{\frac{1}{5}}
$$

where $m_{s,a}$ is the sub-metric a for the subgroups.

Model	**Embedding**	**Dev GAUC**	**Test GAUC**
Single Models			
Logistic Regression Baseline	N/A	N/A	0.6930
Naive Bayes	N/A	N/A	0.6402
GRUCNN	GloVe	0.7296	0.7738
GRUCNN	fastText	0.7330	0.7766
GRUCNN + self-attention	fastText	0.7483	0.7785
GRUCNN	GloVe + Character Embedding	0.7366	0.7545
GRUCNN	GloVe + fastText concatenate	0.7301	0.7660
GRUCNN	GloVe + fastText average	0.7151	0.7496
LSTMCNN	GloVe	0.7050	0.7621
LSTMCNN	GloVe + fastText concatenate	0.7404	0.7633
doubleRNN	GloVe + fastText average	0.7295	0.7604
doubleRNN	fastText	0.7430	0.7632
doubleRNN + self-attention	GloVe + fastText average	0.7524	0.7682
Ensemble Models			
doubleRNN + self-attention ensemble with translated training texts augmentation	GloVe + fastText average	N/A	0.7649
Best 5 models ensemble	N/A	N/A	0.7660

For non-neural methods, logistic regression achieves a better performance than naive bayes. Comparing to logistic regression, naive bayes has a strong assumption on the conditional independence among features. However, this assumption could be severely violated in natural language, as features are very likely to co-occur in a toxic comment.

For neural methods, the best performing model is GRUCNN + self-attention using fastText word embedding. Concatenating different word embeddings and using augmented data during training do not help improve the performance on testing. Also, adding character-level embedding makes the model more vulnerable to overfitting and weakens the performance. However, self-attention does help increase the generalized AUC by a small margin.

5.5 Quantitative Analysis

In order to determine the performance of our best model, we pick the top 5 most frequently mentioned identities, and compare the confusion matrix of them to the overall confusion matrix. We are more concerned about false positive rate

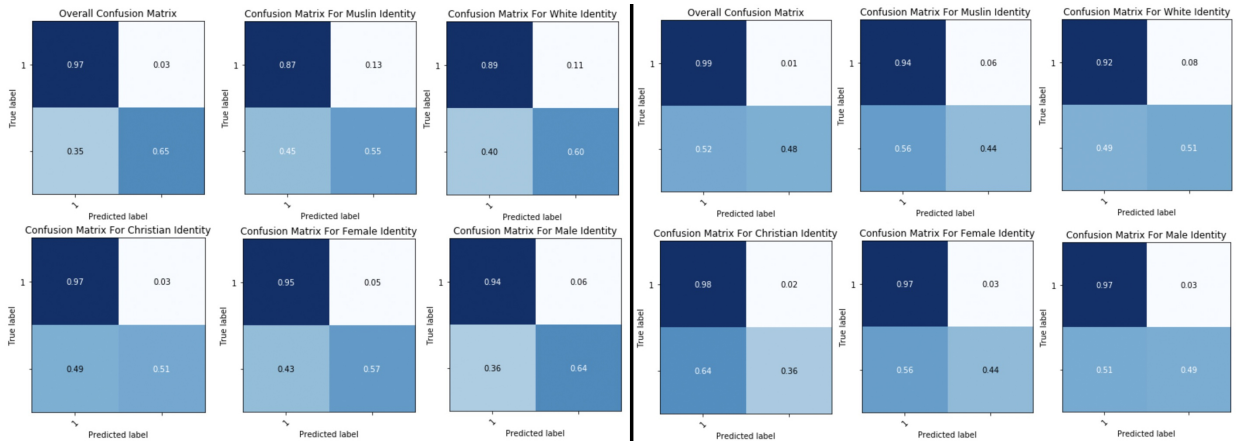

Figure 3: Confusion Matrix of GRUCNN + self-attention

because we do not want to classify toxic comments solely based on words associated with these identities. Figure 3 shows that for all 5 identities, the false positive rates were higher than the overall false positive rates. The false positive rate for Muslim and Christian are extremely high. The false negative rate for Muslim and White are extremely high. This also shows despite using huge amount of training data that are related to Muslim identity, our model cannot perform well on comments related to Muslim. Comparing to the baseline model, our neural models have significantly decreased the false positive rate and successfully reduced the bias in the toxicity classification.

5.6 Qualitative Analysis

We will examine two false positive examples of the comments related to Christian and Muslim.

a dozen muslim countries publicly behead homosexuals and hillary clinton took oos of millions of from them all

For the Muslim case, we found that comment containing identities related to Muslim also tends to include other identities that could be easily misclassified. It mentions Muslim as well as 'behead homosexuals', which in regular context can be considered as toxic.

For the Christian case, we noticed the similar pattern. Especially, many sentences that mentioned the Christian identity also mentioned the Muslim identity. For example,

what does being a christian have to do with it ? are christians supposed to willingly allow their countries to be invaded by muslims who outbreed them : ?

The comment will be classified as toxic almost all the time when Muslim and Christian are mentioned together.

6 Conclusion and Future Work

To conclude, our end-to-end neural methods outperform non-neural methods in a large margin. For neural methods, models that have simple architecture such as GRUCNN are preferred, and self-attention does help highlight important words that are informative to classification. Our best performing neural model achieves 0.7785 generalized AUC on the test set. From the result and error analysis, we found that our model can effectively classify toxicity, but still struggles when there are multiple frequently attacked identities simultaneously being mentioned. Further work includes incorporating pretrained BERT and ELMo as the embedding layer, and training separate models for subcategories of toxicity and identities.

7 Contribution

- Jeremy: data cleaning and preprocessing, non-neural methods, character embedding + GRUCNN, Error Analysis.
- Louise: EDA, training set data augmentation, neural methods, hyperparameter tuning.

8 Code

Please refer to the follow github repo: https://github.com/MiYu1996/CS229Project

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1412–1421, Association for Computational Linguistics, Sept. 2015.

[3] D. Yin, B. D. Davison, Z. Xue, L. Hong, A. Kontostathis, and L. Edwards, "Detection of harassment on web 2.0," 2009.

[4] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, (Stroudsburg, PA, USA), pp. 656–666, Association for Computational Linguistics, 2012.

[5] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," in *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, SOCIALCOM-PASSAT '12, (Washington, DC, USA), pp. 71–80, IEEE Computer Society, 2012.

[6] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, (Republic and Canton of Geneva, Switzerland), pp. 145–153, International World Wide Web Conferences Steering Committee, 2016.

[7] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, (Valencia, Spain), pp. 1–10, Association for Computational Linguistics, Apr. 2017.

[8] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos, "Convolutional neural networks for toxic comment classification," in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN '18, (New York, NY, USA), pp. 35:1–35:6, ACM, 2018.

[9] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos, "Deeper attention to abusive user content moderation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 1125–1135, Association for Computational Linguistics, Sept. 2017.

[10] Y. Mehdad and J. Tetreault, "Do characters abuse more than words?," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, (Los Angeles), pp. 299–303, Association for Computational Linguistics, Sept. 2016.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[12] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[13] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *CoRR*, vol. abs/1508.06615, 2015.

[14] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015.

[15] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," *CoRR*, vol. abs/1611.06639, 2016.

[16] M. R. A. Natural Language Computing Group, "R-net: Machine reading comprehension with self-matching networks,"

[17] P. Ostyakov, "Data augmentation through translation," 2018.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[19] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman, "Nuanced metrics for measuring unintended bias with real data for text classification," *CoRR*, vol. abs/1903.04561, 2019.