# Building Unbiased Comment Toxicity Classification Model

Louise Qianying Huang, and Mi Jeremy Yu

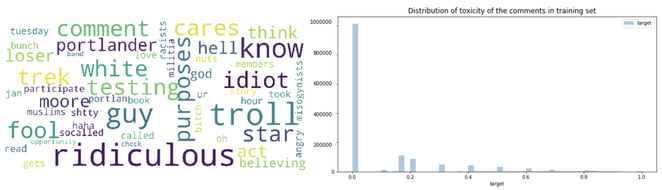
Department of Statistics, Stanford University
{qyhuang, miyu1996}@stanford.edu

## Task Introduction

Machine learning and deep learning techniques have achieved state-of-the-art results in many natural language processing tasks, such as Question Answering and Neural Machine Translation. However, natural language often contains bias and stereotyping, and models that train on such corpus will unconsciously learn and intensify those biases. In the previous iteration of Jigsaw toxicity classification competition, the data contain unintended 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. In this project, we aim to build an unbiased classification model that takes in raw online comments as input and to classify toxicity of those comments, using both non-neural and neural techniques.

## Data

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. Additionally, each example is labelled with a variety of identity attributes, representing whether the identities that are mentioned in the comment. Since the original test set does not have the identity attributes, for faster testing, we held out 5% of the original training set as our new test set. Therefore, the new training dataset size is 1715241, and the new test dataset has 89633 comments. The left graph shows some common words associated with toxicity greater than 0.5. The left graph shows the distribution of toxicity of the training set.

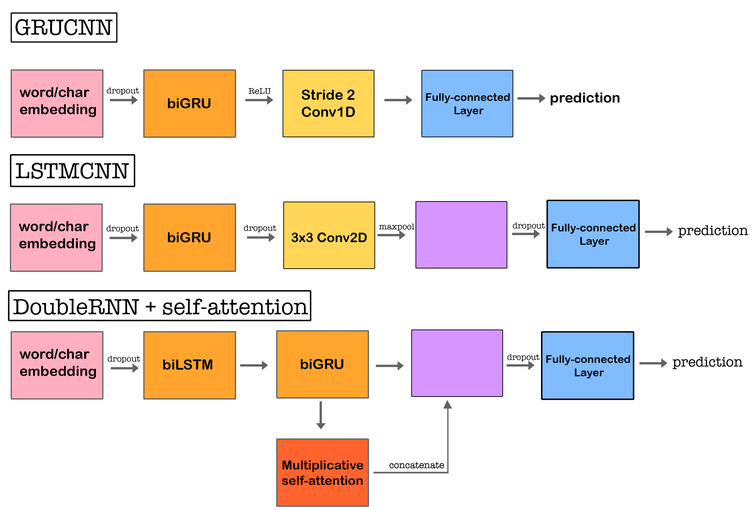



## Non-Neural Methods

- **Feature Extraction.** We first used 1-gram and 2-gram word tokens and 1-gram up to 6-gram character tokens to tokenize the corpus. Then we counted the occurrence of each tokens with sublinear scaled frequency to smooth the influence of count. In the end we normalized the frequency by inverse document frequency. We selected the top 20000 word features and 30000 character features.
- **Logistic Regression Baseline.** The logistic regression classifier has the form $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, which returns the probability of a comment being toxic.
- **Naive Bayes Classifier Baseline.** Multinomial event model is under the assumption that $x^{(i)}$ are conditionally independent given y. The likelihood for multinomial event model is $l(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^{n}(\prod_{j=1}^{d_i} p(x_j^{(i)}|y; \phi_k|y = 0, \phi_{k|y=1}))p(y^{(i)}; \phi_y)$. After estimating the parameters, the corresponding probability can be calculated using the Bayes rule.

## Neural Methods

- **Embeddings.** We used various pretrained word embeddings on their own or combined them together to mitigate the potential biases in any embeddings. We experimented with GloVe 300D trained on Common Crawl [3] and fastText 300D trained on Common Crawl [2]. We also implemented character-level embedding introduced in [1].

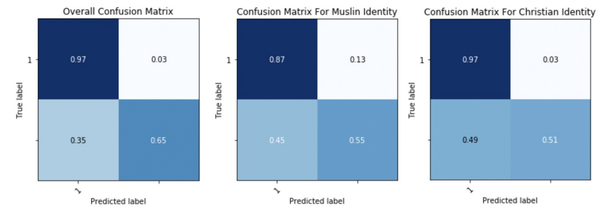



## Results

The evaluation metric is generalized AUC.

| 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 | 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 |

## Discussion and Conclusion




To conclude, pre-trained embeddings are crucial and variations of network architectures have minor influences on the performance, and models with simpler structure are preferred. Seq2seq self-attention does help increase the performance of the model. For future work, we think incorporating pre-trained BERT and ELMo as the embedding layer may drastically increase the performance. Also, if time permitted, training separate models for different subcategories of toxicity may also boost the performance.

## References

[1] Yoon Kim et al. "Character-Aware Neural Language Models". In: *CoRR* abs/1508.06615 (2015). arXiv: 1508.06615. URL: http://arxiv.org/abs/1508.06615.

[2] Tomas Mikolov et al. "Advances in Pre-Training Distributed Word Representations". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/