Detecting Toxicity in Online Forums

Amy Chen, Farzaan Kaiyom, Kristen Anderson
Stanford University (CS 230: Deep Learning)
amyjchen, farzaank, ander50n

1 Abstract

We implemented NLP classifiers to detect online English toxicity messages, which could be used to minimize hate speech and support inclusive online communities. Our data was taken from the Jigsaw Unintended Bias in Toxicity Classification competition, and we worked within the competition's satisficing constraints and an ROC-AUC combined with the generalized mean of Bias AUCs. We pursued several models as well as optimizing and bias-reducing methods. Our most effective model was a rank ensemble between BERT, refined on our dataset, and an LSTM with embeddings, auxiliary labels, and data augmentation. Additionally, we implemented attention and LSTM hyperparameter tuning, but found these to be less effective within our constraints.

2 Introduction

The proliferation of hate speech in social media spaces has created a dilemma for tech companies. Thus, we are creating a classifier that detects toxicity in English online messages. The input to our classifier is an online chat forum comment and the output is whether this comment is toxic (ie. offensive or discriminatory against an identity group). Our training and testing are centered on a Kaggle competition, "Unintended Bias in Toxicity." Our classifiers take in comment text originating from an online political discussion platform and output whether or not that text is toxic. In our work, we compare and combine LSTM, Attention, and BERT models with various bias reducing and optimization techniques (error analysis, data augmentation, auxiliary labels, ensemble averaging, hyperparameter tuning, dropout, AUC score) to pursue an accurate and less biased model. Possible applications of this classifier are detecting online harassment, trolling, and other negative behavior on forums, groups, and other forms of social media. A main challenge is ignoring words commonly misused as insults (e.g. failing to account for the distinction between calling someone gay as an insult and someone simply identifying oneself as gay), which out-of-the-box algorithms fail to fully account for. (Lucas Dixon, 2018).

3 Related Work

In "Reducing Gender Bias in Abusive Language Detection" by Ji Ho Park, they found that due to imbalances in samples, abusive language detection models struggle to avoid ethnic, sexuality, religious, and gender bias, being more likely to interpret "You are a good woman" as sexist than "You are a good man" due to the frequency with which woman is negatively used within training set comments. (Park, 2018). Thus, state of the art pre-trained models such as Google's BERT may be vulnerable to unintentional discrimination when evaluating toxicity, despite having higher accuracy scores than less refined models. Model bias is important to address to ensure that we are not unintentionally discriminating against people associated with these identity labels by being more likely to censure their comments (Lucas Dixon, 2018). In "Nuanced metrics for measuring unintended bias with real data for text classification" by Jigsaw, they defined a metric for measuring the equality gap: the difference of true positive rates of a subgroup and a background threshold (Daniel Borkan).

With respect to models, we researched the best pretrained NLP models as well as different optimization techniques, and pursued LSTMs, Attention (Lucas Dixon, 2018), Bert (Devlin et al., 2018), Rank Ensemble Averaging(Xu et al., 2016; Kanakaraj and Guddeti, 2015), Auxiliary Labels (Nguyen et al., 2011), and Data Augmentation (Lucas Dixon, 2018). For clarity and to reduce redundancy, we will refer to our references findings as these topics are mentioned in the following pages.

4 Data and Features

Our data comes from the a Kaggle competition, "Jigsaw Unintended Bias in Toxicity Classification"(Jigsaw, 2019).

The data was initially released by Civil Comments, a platform for online discussions. It contains the following features:

- Text (comment_text): A string from the internet (forums, comments, etc).

- Target: Labels for types of toxicity in the text (e.g. obscene, identity attack, insult, threat, etc)
- Identity Labels: Labels for identities mentioned in the text (e.g. asian, atheist, bisexual, etc race, religion, sexual orientation, gender, disability, and other protected classes)
- Metadata: Labels for ratings, reacts, date created, publication and article ids, annotator counts.

Our training set was 1,805,383 samples, with all the above features. Our test set is 97,320 samples and only contains the comment text. Its true values were obscured to us and used by Kaggle to evaluate the competition. The comments were typically remarks on a specific political issue.

All labels are in the range 0.0 - 1.0, where 1 is positive. Annotators' labels were averaged to result in the final label. We round anything greater than or equal to 0.5 to 1 (True) and anything else to 0 (False). We found that 6% of the examples in our training set are toxic. Only 29% of the dataset had been annotated for identities.

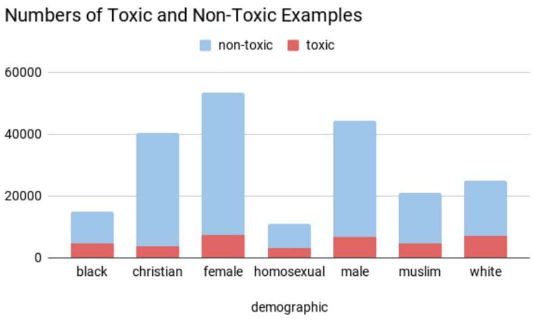

Figure 1: Of annotated data, number of samples by demographic.

4.1 Word Embeddings

We preproccessed all comment text by tokenizing and vectorizing them by word. We used pretrained GloVe embeddings to supplement our model's vocabulary. There are three main sources of GloVe embeddings: Wikipedia, Common Crawl, and Twitter. We used the Common Crawl embeddings, which is more reflective of the internet at large, and which best reflect the disussion-based vocabulary in our samples. (Pennington et al., 2014)

4.2 Data Augmentation

"Measuring and Mitigating Unintended Bias In Text Classification" noted the success of using augmenting data with non-toxic examples of identity terms to reduce bias. In the aforementioned paper, they pulled data from Wikipedia (Lucas Dixon, 2018). It occurred to our team that the difference in sentence length and formality between chat forums and websites such as Wikipedia may skew our data. In addition to extracting data from Wikipedia, we extracted these examples from experience-based articles, editorials, and news, as well as example sentences and encyclopedia articles for a more neutral tone. To scope our search to our project time-line, we primarily looked for examples for the most affected demographics from our error analysis. We wrote a script that labeled each sample based on the presence of different identity keywords. This script also varied the sample length, sometimes splitting text into sentences to keep our model from being biased between longer and shorter texts. We generated a total of 4,580 samples.

5 Methods

5.1 Evaluating Potential Models

We initially explored combining Naive Bayes with a TfidfVectorizer component so as to scale down the impact of frequent word tokens that offer little informative value, such as "the" or "a" (Trick, 2018; Jigsaw, 2018b). We ultimately decided against continuing with this model on account of the naive bayes assumption that words within a sentence are independent of each other(sta, 2009). Following this, we experimented with the GPT-2 model before determining its lack of performance was because it works better on text generation than text classification (H).

We then worked on a simple two-layer NN, an LSTM, a CNN, and a CNN-LSTM, tuning hyper-parameters along the way to evaluate which one we should pursue further. We focused in particular on dropout as well as numbers of filters and/or units. We evaluated each of these by accuracy, although our later models would pursue an equity-based metric.

Model	Train Acc	Val Acc	Test Acc
Two-layer NN	0.9335	0.9316	0.9089
LSTM	0.9554	0.9494	0.9505
CNN	0.9474	0.9467	0.9249
CNN-LSTM	0.9537	0.9501	0.9507

Table 1: Results from our initial models.
Hyperparameters: Two-layer NN:1 32-unit layer, relu-activation, sigmoid activation. LSTM: 50 units, dropout 0.2, sigmoid activation. CNN: 3 CONV-POOL layers(128 filters, dropout 0.35, maxpool size: 5). CNN-LSTM: 3 CONV-POOL layers (size 5, 128 filters; pool sz 2), 1 LSTM (50 units, dropout 0.5), sigmoid activation.

5.2 Error Analysis

Because our focus is on bias, we wanted to understand which demographics were most adversely affected. By finding the percentage of incorrectly classified samples out of total samples for that demographic, we discovered that three identity labels are particularly adversely effected by bias: Black (4.4%), Muslim (2.9%), and homosexual (2.7%). Other labels had error in the 1.0 - 1.5 % range.

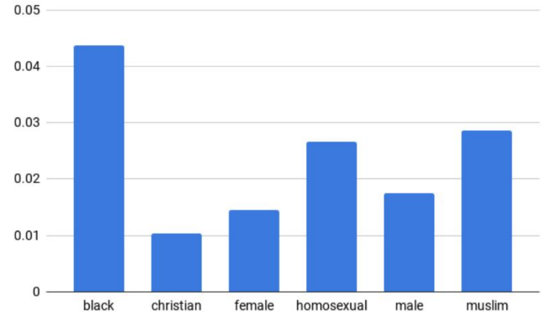

Figure 2: Percent incorrectly classified out of number of samples, for all demographics with larger than 10,000 samples.

Running a script to identify commonly mis-classified keywords also helped us identify specific issues, such as the disproportionate use of the words ”crazy”, ”silly”, and ”insane”: words that also marginalize those with mental health disorders.

5.3 Selecting Models

We used the CNN-LSTM from our initial results as our baseline, given its accuracy as well as its faster runtime than a pure LSTM. We looked into others' public implementations of different models and techniques to see how others tried to solve the bias and accuracy trade-off.

With respect to models themselves, we (1) implemented and refined BERT, a pretrained model for NLP, which applies the bidirectional training of Transformers to language modeling, (Devlin et al., 2018) (2) worked with a better LSTM implementation and did hyperparameter tuning on it for learning rates and epochs, (3) evaluated an LSTM and attention model, and (4) implemented two types of ensemble averaging, linear and rank.

5.4 Limitations

Our methods were limited by a few features of the Kaggle competition: run time limits and access to our test set. This posed a problem for us because if our submission ran longer than 2 hours on a Kaggle GPU, we wouldn't be able to submit to the competition. This meant that we couldn't get our final score (AUC) if training our models ran over two hours. Our lack of access to the ground truth predictions also meant that we couldn't get raw accuracy, precision, or recall for our test set. Kaggle also instills a 13 GB RAM limit, which also became an issue. We've noted when this was an issue in the following sections. While these were frustrating at times, we got to work with satisficing and optimizing constraints.

5.5 BERT

We were drawn into BERT: it has an enormous size of 24 Transformer blocks, 1024 hidden layers, and 340M parameters , and is pre-trained on 40 epochs over a 3.3 billion word corpus (Devlin et al., 2018). We chose to work with BERT's uncased version to refrain from distinguishing words at the beginning of sentences from words within a sentence's interior. To implement and tune BERT, we trained it on our training dataset. Given BERT's size, this took a long time. However, we were able to train the model on the majority of the data and tune hyper parameters such as dropout based on those results.

5.6 LSTM

In contrast to conventional RNN and feed-forward neural networks, LSTMs can selectively remember important patterns within long-term dependencies (Srivastava). This is advantageous when we consider longer chunks of comment text.

5.7 Adding Attention

We adopted an LSTM-Attention model to better represent sentence meaning when evaluating comments. Attention does so by identifying word relevance within comments and aggregating highly-relevant words to create sentence vectors. This process can similarly be performed on a sentence level, determining which sentences have high value in determining toxicity (Lucas Dixon, 2018).

5.8 Ensemble Averaging

Ensemble Averaging is implementing multiple models and combining their results. We decided to focus on averaging BERT and LSTM. We based this decision on The University of Melbourne's ensemble of neural networks and word2vec based models for sentiment classification, which demonstrated that combined models perform better than any single classifier (Xu et al., 2016). This improvement is achieved because errors are averaged between models, leading to both models compensating for the each others' weaknesses while combining strengths. We chose the LSTM and BERT models because those were our best performing models.

Ensemble averaging performed especially well in sentiment classification for data from Twitter (Kanakaraj and Guddeti, 2015) so we felt that it would be ideal for our social platform-based data set. We considered various methods, including linear averaging (equal or static weights between models) and rank averaging (models weighted based on their validation scores). We ultimately selected these two because they were the most intuitive for our task. The general equation for linear averaging is shown below:

$$\hat{y} = \sum_{i=1}^{N} \omega_i y_i(\mathbf{x})$$

where $\hat{y}$ is the final prediction and ω is the probabilistic weight of each model's output y_i out of N models (Hashem, 1997). Our final model uses rank averaging, where the weights are proportional to the validation scores of each model, due to its better performance.

5.9 Auxiliary Labels

In a paper titled "Learning Classification with Auxiliary Probabilistic Information", they state that providing labels with probabilistic weights could increase the accuracy of classifiers (Nguyen et al., 2011). Many public kernels at Kaggle attempted this as well. Intuitively, this should improve performance because it teaches the model to recognize and differentiate between your auxiliary labels. By adding auxiliary identity labels, our model would then be able to classify toxicity along this additional axis of understanding.

5.10 Hyperparameter Tuning

We optimized the LSTM hyperparameters by using the fastai library to train the same Learner and NeuralNet model used within the LSTM over multiple possible learning rates and epochs, so as to compare the loss results of using different hyperparameters. The ideal hyperparameters are 400 epochs with a learning rate value of 0.01, however, this violated our time constraints. As such, we constrained our epoch number to 4 and experimented with learning rates of 0.01 and 0.001.

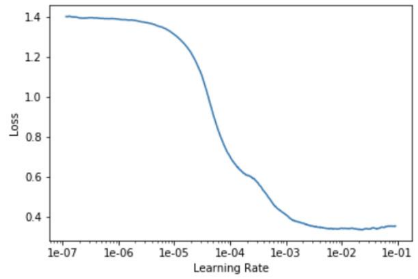

Figure 3: Impact of Epoch Number and Learning Rate on Loss: This is a graph of learning rate versus loss with an epoch number of 1000. An almost exact minimum loss value can be achieved through an epoch number of 400. Other epoch numbers we experimented with included 4, 6, 8, 40, 100, 200, and 500. The lowest loss result for epoch number 100 was 0.9 in comparison to 0.4 for epoch number 400, showing clear improvements in a substantial rise in epoch number.

6 Results

6.1 Metrics: Bias-based Area Under Curve

Our data analysis is based on measuring the Area Under the Curve (AUC). The Kaggle competition we entered provided a bias-based AUC equation, which we used to evaluate both our train and test set. ROC-AUC is a metric that optimizes for correct classifications of both positive and negative results. AUC is the area under the ROC curve, the "probability that a model ranks a random positive example more than a random negative example."(Google).In the context of our problem, AUC evaluates the probability that a randomly selected toxic example will receive a higher toxicity score than a randomly selected neutral example (Daniel Borkan).

Our metric also involved Bias AUCs, the generalized mean between subgroups of AUCs of restricted datasets. These were restricted as follows: **(1) Subgroup AUC:** examples that mention the specific identity subgroup, to evaluate confusion within an identity, **(2) BPSN (Background Positive, Subgroup Negative) AUC**: non-toxic examples that mention the identity and the toxic examples that do not, to evaluate if toxicity predictions are higher than they should be, and **(3) BNSP (Background Negative, Subgroup Positive) AUC**: toxic examples that mention the identity and the non-toxic examples that do not, to evaluate if non-toxic predicitions are higher than they should be. (Jigsaw, 2019)

$$M_p(m_s) = \left(\frac{1}{N}\sum_{s=1}^{N} m_s^p\right)^{\frac{1}{p}}$$

Figure 4: Equation for AUC. We used a generalized mean of bias AUC to consider per-identity BIAS AUCs in one comprehensive measurement. m^p = the p^{th} power-mean function. m_s = the bias metric m calculated for subgroup s. N = number of identity subgroups (Jigsaw, 2018a).

The overall ROC-AUC value was then combined with the generalized mean of the Bias AUCs to result in a final metric:

$$score = w_0 AUC_{overall} + \sum_{a=1}^{A} w_a M_p(m_{s,a})$$

Figure 5: Equation for final metric. A = the number of sub-metrics. $m_{s,a}$ = bias metric for identity subgroup s using submetric a. w_a = a weighting for the relative importance of each submetric (Jigsaw, 2018a).

6.2 BERT across modifications

Training BERT would take over 7 hours for each iteration, so our main focus was tuning the fraction of the training set used while keeping our entire runtime within 2 hours. As we trained BERT on larger fractions of the dataset, we observed its AUC scores increasing consistently. We were ultimately able to significantly increase the accuracy of BERT. Training on 1/3 of the dataset recorded an AUC of about .92, while our BERT model, trained on 0.7 of the dataset got well above .93 accuracy. We also found that increasing dropout was useful.

Model	**Hyperparameters**	**Architecture**
LSTM Auxiliary Labels, Embeddings	Lr: 0.001 Epochs : 4 Batch size: 512 maxlen=300	Embedding layer with spatial dropout (0.3) 2 stacked bidirectional LSTM Layers 2 hidden linear layers (relu; input: LSTM layers max and avg pool) 2 relu output (1 for targets, 1 for auxiliary labels)
LSTM / ATTN Auxiliary Labels, Embeddings	Lr: 0.001 Epochs : 4 Batch size: 512 maxlen = 220	Embedding layer with spatial dropout (0.3) 2 stacked bidirectional LSTM Layers 2 attention layers 2 hidden linear layers (relu; input: attention layers, max and avg pool) 2 relu output (1 for targets, 1 for auxiliary labels)
BERT	Lr: 0.00002 Batch size: 32 Epochs : 1	Trained Googles pretrained and uncased BERT model with 69.4% of the data for train and export.
Rank Ensemble BERT + LSTM Auxiliary Labels, Embeddings	Lr: 0.001 Epochs : 4 Batch size: 512 maxlen = 220	0.35 BERT 0.65 LSTM

Table 2: Summary of Hyperparameters and Architecture of our final models

6.3 Auxiliary Labels and Embeddings

Our biggest AUC increase to our LSTMs came from having them predict auxiliary labels and using embeddings, a jump of 3% reaching an AUC comparable to that of BERT. While we didn't run these independently of each other, the intuition is that the combination of a larger vocabulary and teaching our models to distinguish between identities helped them understand what toxicity means with respect to different identities.

6.4 Attention

Our LSTM with Attention performed worse than our LSTM without attention. This is likely because, as we found while tuning, a high max length for our sequences would cause us to run over our 13 GB RAM limit.

6.5 Data Augmentation

Data augmentation on our LSTMs provided mixed results. It improved our baseline and ensemble averaged models but made our pure LSTM perform worse. Our examples may have confused our LSTMs by containing examples of people talking about harassment and other negative experiences in connection to identity. The increased performance of the ensemble averaged model, however, indicates there may be some benefit the LSTM gained from this that our BERT model lacked.

Model	Train	Train DA	Test	Test DA
BL	0.9618	0.9645	**0.9067**	**0.9080**
L	0.8769	0.8929	**0.9364**	**0.9359**
L/A	0.8188	0.8139	**0.9346**	**0.9339**
B	N/A	N/A	**0.9365**	N/A
EA	N/A	N/A	**0.9391**	**0.9392**

Table 3: AUC scores of our final models. ***Abbreviations:*** *DA = Data Aug, BL: Baseline (CNN-LSTM), L = LSTM, A = Attention, B = BERT, EA = Ensemble Average (BERT + LSTM)*

6.6 Ensemble Averaging

The combinations of different models succeeded in increasing our overall accuracy on the test sets, producing our best model. For example, the standalone BERT and LSTM models had around 93 percent accuracy, but by weighting the average outputs of the models we were able to reach around 94 percent accuracy (93.914 to be exact). Linear averaging increased our accuracy too (up to around 93.8), but rank averaging increased it to be where it is right now. We concluded that this meant that our models indeed compensated for each other's errors, but that it was still important to weigh the better model's ”opinion” the most.

7 Conclusions

We explored several avenues to improve our AUC metric, an indicator of how biased our model is. We found that ensemble averaging, auxiliary labels, and embeddings were incredibly effective, while — in fitting our satisficing constraints — data augmentation, attention, and specific hyperparameter tuning were less effective.

8 Future Work

Error analysis on our ensemble averaged model, would help us further refine it. Using his analysis, we can identify problems in our data augmentation to further supplement and create an even more equitable and effective dataset. Other than that, we would like to train out models outside of competition contraints. We would have liked to train our LSTM for 400 epochs, run our Attention model with longer sequence lengths, and further refine BERT by training it on the aforementioned augmented data. Thorough it all, we would further pursue ensemble averaging.

9 Contributions

As requested, our individual teammate contributions are as follows:

Amy Chen: Baseline Models (2-Layer NN, LSTM, CNN-LSTM), Error Analysis (scripts and analysis), data augmentation (collection, scripts for processing, using augmented data on all our models, and analysis), calculating AUC on all our models, Poster design.

Farzaan Kaiyom: Baseline Model tuning (CNN), BERT implementation and tuning, ensemble averaging model research and implementation (linear and rank with BERT and LSTM)

Kristen Anderson: Digging into and summarizing research papers, BERT (first try), hyperparameter tuning (LSTM), Baseline model (Naive Bayes w/ TfidfVectorizer)

Team Efforts and Other Sources: Finding our final models was a team effort, and most of the credit should go to the creators of those models on and outside of the Kaggle community (LSTM + Attention (Sang-Won), BERT, BERT + LSTM ([ods.ai] Lyalikov Artyom)). We also made use of Pytorch(Paszke et al., 2017), scikit (for metrics) (Pedregosa et al., 2011), and Keras (Chollet et al., 2015). Many thanks to Jigsaw for providing our dataset as well as outlining metrics to follow. (Jigsaw, 2019)

10 Acknowledgements

We would like to thank the CS 230 teaching team, without whom we would not have the means through which to do this project. In particular, we would like to give a special shoutout to Suvadip Paul, our project mentor, for guiding us through our project.

References

2009. Properties of naive bayes. *Cambridge University Press*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Jeffrey Soren Nithum Thain Lucy Vasserman Daniel Borkan, Lucas Dixon. Nuanced metrics for measuring unintended bias with real data for text classification. *Jigsaw*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Google. Classification: Roc curve and auc — machine learning crash course — google developers.

Jim H. Two small experiments on gpt-2.

Sherif Hashem. 1997. Optimal linear combinations of neural networks. *Neural networks*, 10(4):599–614.

Jigsaw. 2018a. Competition evaluation.

Jigsaw. 2018b. Countvectorizer, tfidfvectorizer, predict comments.

Jigsaw. 2019. Jigsaw unintended bias in toxicity classification.

Monisha Kanakaraj and Ram Mohana Reddy Guddeti. 2015. Nlp based sentiment analysis on twitter data using ensemble classifiers. In *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–5. IEEE.

Jeffrey Sorensen Nithum Thain Lucy Vasserman Lucas Dixon, John LI. 2018. Measuring and mitigating unintended bias in text classification. *Jigsaw*.

[ods.ai] Lyalikov Artyom. Bert lstm (simple blender) - 0.93844 lb.

Quang Nguyen, Hamed Valizadegan, and Milos Hauskrecht. 2011. Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*, pages 477–486. IEEE.

Ji Ho Park. 2018. Reducing gender bias in abusive language detection. *Centre for Artificial Intelligence Research*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sang-Won. Simple lstm attention.

Pranjal Srivastava. Essentials of deep learning: Introduction to long short term memory.

Andrew Trick. 2018. Classifying comment toxicity w/ nltk and nb.

Steven Xu, HuiZhi Liang, and Timothy Baldwin. 2016. Unimelb at semeval-2016 tasks 4a and 4b: An ensemble of neural networks and a word2vec based model for sentiment classification. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 183–189.