Comparison of Classification Models on Unintended Bias in Toxicity Classification

Mijeong Ban[1] and Nathaniel Burgdorfer[2]

[1]Stevens Institute of Technology , mban1@stevens.edu
[2]Stevens Institute of Technology , nburgdor@stevens.edu

Abstract

The goal of this work is to approach the problem of text classification in a comparative manner. To the best of our knowledge, there is a general consensus that the relative state-of-the-art methods in NLP for representing and encoding words is through vectorization and word embeddings using methods such as Word2Vec, GloVe, or BERT [3]; however, there is no real consensus as to which learning method is best for the task of text classification, especially with regards to model bias. There are many models that can be used for the task of text classification, each with their benefits and drawbacks. We look to draw a comparison between different learning models on the basis of specifically measuring unintended model bias as well as prediction accuracy. We look to add this extra dimension to evaluation as it is an important part of ensuring models are as accurate and as appropriate as they should be, even when the data may be skewed or biased in one way or another.

1 Introduction

When discussing text classification and understanding, it sometimes may not be enough to just measure the accuracy of the model evaluated on a given dataset. Most often in the process of data collection, there can exist some skew or some bias in the data being collected and in the layout or distribution of the collected data. When the goal of machine learning and classification tasks is most often to recover the true distribution of the data involved in the task, it is of utmost importance that the specific sampled dataset that we use in this process does not affect the outcome of the model and lead to a distribution far from the true distribution. In this work, we look to analyze and compare a convolutional neural network model and a recurrent network model for the task of text classification on the toxicity in online comments. For this task, it is important that we not only compare traditional classification accuracy but also take into account a bias metric.

2 Background and Related work

This task started from one of the Kaggle competitions, called Jigsaw Unintended Bias in Toxicity Classification [1]. With the given dataset which includes comments with some of them being toxic(a rude, disrespectful, or unreasonable comment that is somewhat likely to make someone leave a discussion) comments in online conversation, the Conversation AI team, a research initiative founded by Jigsaw and Google, built toxicity models which recognize toxicity in comments. They found that their models incorrectly learned to associate certain subgroups with certain labels. Based on existent bias in the collected data, whether this be due to a high volume of certain classes in the data or due to repeated samples of certain pairings (e.g. comments including the subgroup 'white' having the label 'toxic'), a model may learn to just associate the existence of a subgroup in a comment with a label of 'toxic' or 'not toxic' instead of learning to classify the entire context of the comment. In this work, we look to compare the results of different machine learning models with a focused evaluation on error resulting from model bias as well as overall prediction accuracy. We will be referencing some current state-of-the-art architectures for text classification and semantic analysis involving CNNs [4][5] and RNNs [6][7]. We look to experimentally analyze how each architecture responds and learns from the data and to compare how each may avoid biased word association and instead focus on the context of comments.

3 Approach

Our main goal is to use the competition as a basis of our investigation of different classification models. We look to make a comparison between different architectures, staring with base CNN and LSTM models. We chose these models as they are some of the most common models when it come to classification tasks. We will look to see how changes and additions to the model affect the performance with respect to binary cross-entropy loss, bias accuracy, model complexity, and training time per iteration. Additionally, we chose to use a bidirectional LSTM models as our more advanced models due to the benefits these types of models have shown in text classification tasks. They contain the ability to receive the input sentences in both forward order as well as reverse order. The idea is to possibly obtain a better understanding of the context of the sentence by passing the input through the forward and backward layers to obtain the output. Our efforts will be focused on building these models with the goal of classification and bias accuracy in mind. Following a detailed comparison, we look to draw conclusions as to which advancements and changes between architectures affected the overall results and effectiveness of our models. We also chose to use the GloVe word embeddings over classical TF/IDF or other keyword-based approaches for this classification task. Pre-trained word embeddings have shown to be more powerful and robust, allowing to take into account context from surrounding words in a sentence. We will further explore our experimental design choices in the following section.

4 Experimental design

In this work, we used Python 3.8 and executed our program locally in the hardware setting of AMD Ryzen 12-Core, 64GB RAM, GeForce RTX 2080 Super.

4.1 Dataset

Our dataset was obtained from the Kaggle competition, Jigsaw Unintended Bias in Toxicity Classification [1]. As the Civil Comments platform made their $\sim 2m$ public comments available to help researchers understand and improve civility in online conversations, Jigsaw helped to extend annotation of this data by asking human raters to rate the toxicity of each comment. The datasets contain train data, test data and other extra datasets for additional research. Train data includes the text of the individual comment in the *comment_text* column and a toxicity label in the *target* column. The *target* values are fractional values which represent the fraction of 10 human raters who believed the label fit the comment with $target >= 0.5$ being considered to be in the positive class (*very toxic / toxic*), and $target < 0.5$ being considered to be in the negative class (*hard to say / not toxic*). Test data has only the comment text values so that models can predict the target toxicity. The train data also has additional toxicity sub-type attributes for research, such as *severe_toxicity, obscene, threat, insult, identity_attack, sexual_explicit*. Furthermore, a subset of comments have identity attributes, *male, female, homosexual_gay_or_lesbian, etc*, which shows what identities that were mentioned in the comment. In this experiment, we randomly sampled 15% of train data for validation set, and labeled as $true$ for samples where its target value is greater than 0.5 and $false$ otherwise. Additionally, we performed the data analysis focusing on identity attribute. Table 1 and Figure 1 show the number of toxic comments, non-toxic comments and weighted toxicity for each identity group. On Figure 1a, it shows the proportion of toxic and non-toxic comments for each identity group and we found out there are many comments about christian, female and male identity groups and a lot of them are not toxic comments. Figure 1b shows the weighted toxicity on each identity group, and the top three of mentioned identities are white, black, homosexual.

4.2 Classification Models

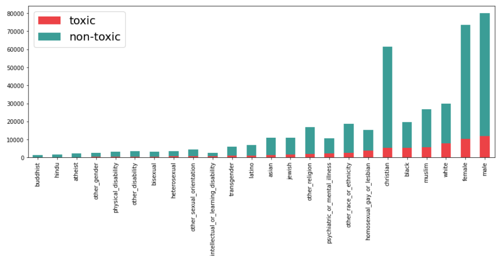
We used GloVe model with 42 billion tokens, 1.9 million vocabulary and 300 dimensional vectors as the input embedding for different text classification models. For algorithms that we used for this experiment, we started from the basic architecture of CNN and LSTM model using Keras and built their variants models.

4.2.1 CNN with Spatial Dropout and Batch Normalization

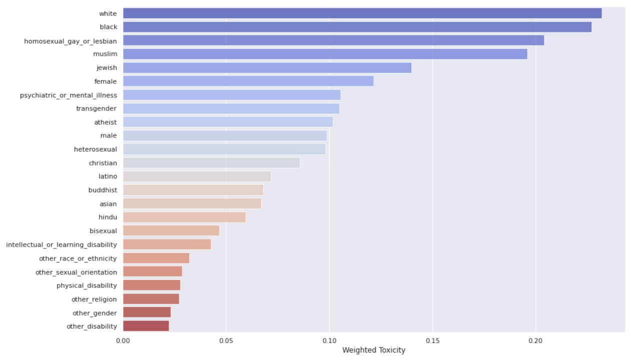
On the top of the basic CNN model with four convolutional layers and max pooling layers, we built its variant model by adding four layers of batch normalization and a spatial dropout layer. Specifically, we chose to use batch normalization for feature standardization to the hidden layers so that it converges faster and makes networks stable, and spatial dropout for better generalization.

Identity Group	Toxic	Non-toxic	Weighted
Male	11797	68382	0.098888
Female	10426	63264	0.121557
Transgender	1082	5038	0.104907
Other gender	427	2296	0.023291
Heterosexual	718	2735	0.098140
Homosexual gay or lesbian	3848	11459	0.204114
Bisexual	530	2800	0.046763
Other sexual orientation	811	3697	0.028722
Christian	5445	55915	0.085739
Jewish	1615	9290	0.139758
Muslim	5643	21007	0.195955
Hindu	196	1361	0.059533
Buddhist	162	1204	0.067936
Atheist	279	1974	0.101816
Other religion	2022	14710	0.027155
Black	5466	14097	0.227006
White	7813	22135	0.231967
Asian	1229	9746	0.067025
Latino	1123	5813	0.071814
Other race or ethnicity	2698	16169	0.032047
Physical disability	448	2779	0.027687
Intellectual or learning disability	825	1823	0.042502
Psychiatric or mental illness	2412	8253	0.105469
Other disability	457	3088	0.022368

Table 1: The number of toxic/non-toxic comments on each identity group



(a) Analysis of toxicity within each subgroup.



(b) Weighted analysis of most frequent identities

Figure 1: Data analysis of subgroups and toxicity ratios.

4.2.2 LSTM with Attention

To make changes to the LSTM model, we added one more LSTM layer and attention layer. The attention layer that we used was a basic Generalized Attention layer using additive attention. This takes in as input the output from the LSTM. This takes in all the hidden states from the previous layer, applying a weight and bias to each state, as well as applying a hyperbolic tangent and exponential function to the input. There is the option of using an attention mask, but this was not included in our implementation. This was added in an attempt to see how the attention layer would perform for our given experiments. Attention is know to be beneficial to many tasks of natural language processing. Mainly in translation tasks and text generation tasks. We wanted to see how attention would aid in the task of text classification, specifically if attention would help with bias accuracy in our model.

4.2.3 Bidirectional LSTM

For our more advanced model, we chose to use a Bidirectional LSTM architecture. These types of models are typically very good at text classification stemming from the fact that the input is processed forward and backwards, hopefully helping the model to learn the context of each input. The construction of our Bidirectional LSTM model is our most complex, starting with embedding and spacial dropout, followed by two bidirectional LSTM layers. Following these beginning layers, we added a concatenation of pooling layers, as well as a cascading structure of additive fully connected layers. As we attempt to advance this model, we added the same attention layer as the LSTM model to our bidirectional model following the bidirectional LSTM layers. Using these architectures, we look to compare their results and performance.

4.3 Evaluation

Once we construct several different CNN and RNN models, training each and evaluating them on their prediction accuracy and bias. The comparison of these models will include their respective time requirements (training time), complexity (model parameters), and model parameter values (learning rate, optimization, loss, etc). However, the comparison will be more focused on the evaluation method that the competition provided, ROC-AUC [2]. This evaluation method is a newly developed metrics which contains three different sub-metrics(Subgroup AUC, BPSN and BNSP), so that we can properly evaluate the overall performance with respect to overall classification accuracy and unintended bias evaluation. The goal is to restrict the data samples during each sub-metric evaluation in order to extract different bias evaluations from the model. In the first sub-metric, we look at each specific subgroup and measure the accuracy score within the subgroup. This tells us how well the model performs at classifying comments within a subgroup. The BPSN (Background Positive Subgroup Negative) sub-metric focuses on negative (non-toxic) samples within the subgroup and positive (toxic) samples outside of the subgroup. This sub-metric is a measure of the false-positive rate of the model. The lower this score, the more biased the model is in classifying a comment containing a subgroup as toxic when it has a true value of non-toxic. Furthermore, the BNSP (Background Negative Subgroup Positive) sub-metric focuses on positive (toxic) samples withing the subgroup and negative (non-toxic) samples outside the subgroup. This sub-metric is a measure of the false-negative rate of the model. The lower the BNSP score, the more biased the model is in classifying comments containing a subgroup as non-toxic when it has a true value of toxic. The latter two sub-metrics also include the 'Background' data in the metric to incorporate the accuracy of the model's predictions of the incorrect label for the subgroup.

The overall score is calculated as follows:

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

where,

$$\begin{aligned} M_p &= \text{the } p^{th} \text{ power-mean function} \\ m_s &= \text{the bias metric } m \text{ calculated for subgroup } s \\ N &= \text{the number of identity subgroups} \end{aligned}$$

The above equation is the per-identity bias AUC score. This paper combines this score with the overall AUC score to obtain the final evaluation score,

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

where,

$$\begin{aligned} A &= \text{the number of sub-metrics (3)} \\ m_{s,a} &= \text{the bias metric for subgroup } s \text{ using sub-metric } a \\ w_a &= \text{a weighting for the relative importance of each metric; (all four are set to 0.25 for this evaluation)} \end{aligned}$$

For the weights (w_0, w_a), we wanted to stay consistent with the competition to be able to make a comparison between our results and the results show by the competition; therefore, each weight has the same value of 0.25. This could be changed to weights having different values based on which sub-metrics we want to highlight. For example, we could highlight BPSN sub-metric if we look to evaluate more on false positive rates; however, staying consistent with the competition and weighing all sub-metrics the same, as well as the overall accuracy score, we produce our overall bias score for this paper.

5 Experimental results

To start, we began with our experimentation with the basic instance of a CNN. As the table shows, this model obtains our worst performance. This has the lowest bias accuracy score as well as the highest loss value. The vanilla CNN architecture was the fastest model, however, with the CNN architectures in general performing better with respect to training time. The next model we looked at was the vanilla LSTM model. This model out-performed the base CNN model, improving in both bias accuracy as well as loss. This did result in a higher train time. The next architecture we used was improving upon our existing CNN architecture, adding spatial dropout as well as batch normalization after each convolutional layer. This increased our bias accuracy from our vanilla CNN model, and also reduced our loss, with a slight increase in our training time per iteration. This CNN model still was faster that our base LSTM implementation; however, the CNN architecture still under-performed the LSTM model. The next step up in our LSTM architecture included deepening our network. We added a second LSTM layer, resulting in an increase in bias accuracy and a decrease in loss, with a bump in model complexity and required train time. Another method we tested was adding an attention layer following the two LSTM layers. This was done to empirically test how attention affected our network for the task of text classification. As we can see, the attention layer did improve our loss from the previous LSTM implementation, as well as slightly increasing our bias accuracy score. This did come with a slight increase in complexity, as well as train time. Moving on to some of our more advanced models, we looked to see how the bidirectional LSTM models compared to the base and more advanced CNN and LSTM models. The Bidirectional LSTM model showed improvement in both loss and bias accuracy, but showed a significant increase in model complexity and training time. The Bidirectional models were more involved than the simple LSTM models we had written so these increases are expected. Continuing our exploration, we looked to again test the benefits of an attention layer in our models. We added a generalized attention layer following the two bidirectional LSTM layers. This again only increased our bias accuracy slightly, while decreasing our loss and increasing our model complexity and training time per iteration. Finally, we looked to deepen our previous model and added a 2 more fully connected layers, which gave us our best performing model, with the highest complexity and training time our of the models we had tested.

Model	Loss	Bias Accuracy(AUC)	Parameters	Trainable Parameters	Train Time (s/epoch)
CNN	0.1320	0.9049	123,392,213	378,113	78
LSTM	0.1244	0.9093	123,233,877	219,777	93
CNN+(SD,BN)	0.1247	0.9083	123,394,261	379,137	90
LSTM+(x2)	0.1234	0.9132	123,365,461	351,361	135
LSTM+(x2,ATT)	0.1222	0.9134	123,365,945	351,845	141
Bi-LSTM+(x2)	0.1212	0.9168	124,373,461	1,359,361	268
Bi-LSTM+(ATT)	0.1199	0.9172	125,029,945	2,015,845	276
Bi-LSTM+(ATT,FC)	0.1187	0.9173	126,341,945	3,327,845	285

6 Conclusion and future work

The comparison of the different model architectures we explored led us to a few different conclusions. Generally, the more complex the structure of the models, the higher the bias score. This does come with some conditions. When we increase complexity in our models, we must make sure to incorporate ways to aid the model in generalization. It is important to avoid over-fitting the data and to create a robust and generalized model to produce as little bias towards the training data as possible. We also noticed that the attention layers did not improve the models we had tested significantly. Typically, attention models and layers are suited for sequence-to-sequence tasks, such as translation and text generation. We wanted to explore the effects of incorporating this type of structure into a text classification system. While the attention layer did improve upon the performance of the models, the improvements and the overall difference in the model was minimal. Lastly, we observed the differences between the CNN-based architectures and the LSTM-based architectures. The CNN tended to be faster, yet more complex, as well as less accurate from a bias accuracy standpoint compared to the LSTM models. As we can see from the Figure 2, the CNN models were actually better performing than the base LSTM models in the area of overall accuracy. This leads to the assumption that when inspecting the bias in the models that we have tested, the CNN architectures tended to be more biased, even with an increased accuracy rating than the LSTM models. In future derivations of this work, it may be beneficial to take a look into a few different improvements. The dataset we used included additional categories of classification for each subgroup mentioned in the comment. We could look to leverage these subgroup labels in the classification process in order to possibly reduce the bias towards a given subgroup. We could also test the difference in embeddings used. In our models, we used GloVe, but we could test how BERT performs, as well as possibly an aggregation

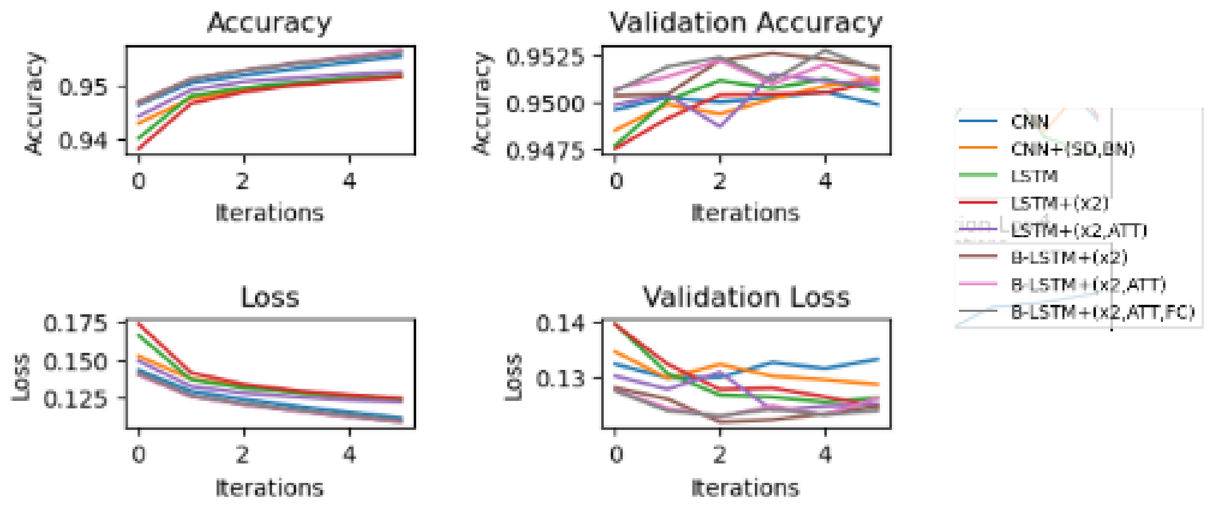


Figure 2: Analysis of training and validation accuracy and loss.

of different embeddings. GloVe provides an available embedding trained on Twitter data that may be useful for toxicity classification. A more ambitious approach to improving our results would be to create a custom loss function to specifically penalize models that are estimating classification in a biased manner. This would include a way to measure error in classification given a specific sample with a specific subgroup label included. This would take a substantial amount of work and ingenuity, but may prove to provide much better results with regards to bias accuracy.

References

[1] Jigsaw/Conversation AI. Jigsaw unintended bias in toxicity classification. URL `https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification`.

[2] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019. URL `http://arxiv.org/abs/1903.04561`.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[4] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. pages 562–570, July 2017. doi: 10.18653/v1/P17-1052. URL `https://www.aclweb.org/anthology/P17-1052`.

[5] Yoon Kim. Convolutional neural networks for sentence classification. pages 1746–1751, October 2014. doi: 10.3115/v1/D14-1181. URL `https://www.aclweb.org/anthology/D14-1181`.

[6] Baoxin Wang. Disconnected recurrent neural networks for text categorization. pages 2311–2320, July 2018. doi: 10.18653/v1/P18-1215. URL `https://www.aclweb.org/anthology/P18-1215`.

[7] Zeping Yu and Gongshen Liu. Sliced recurrent neural networks. *CoRR*, abs/1807.02291, 2018. URL `http://arxiv.org/abs/1807.02291`.