Toxic Comment Classification and Unintended Bias

Naga Santhosh Kartheek Karnati
karnati.n@northeastern.edu

Shubhanshu Gupta
gupta.shubh@northeastern.edu

Abstract

In this project we worked on Jigsaw Unintended Bias in Toxicity Classification Challenge on Kaggle. We built an unbiased classification model to identify toxicity in online comments using machine learning models such as Logistic Regression and Naive Bayes, and deep learning models mostly making use of RNN and LSTM. Our best-performing model achieves 0.809 generalized AUC on the test set.

1 Summary

There has been an exponential rise in the number of people who use social media every day. With more and more people gaining access to social media sites, there is an increase in the number of posts, comments and interactions posted every day. With increase in number of posts, there is also a huge increase in the amount of toxicity that is being put every day on these sites. Toxicity can be defined as an insult, threat, identity attack or just pure obscenity. These kinds of comments are known to be more targeted towards certain groups of people or identities. Social media sometimes fail to moderate such content especially in cases where there is bias against some community for example, moderation models might associate a non-toxic comment like “as a gay man” and a toxic comment like “why are you acting gay?” together which creates a bias against people from the LGBTQIA community. Our objective is to develop moderation models that can find toxic comments or posts on social media sites and identify bias against people from certain communities. Also, we aim to find out the communities which face the highest amount of bias.

2 Dataset Description

We used Kaggle dataset [1] which was made available in a competition hosted by Google and was taken from Civil Comments platform. The dataset has around 2 million comments and has around 40 columns. Our focus in the project is to find toxicity and so we use columns such as ‘target’ and ‘comment_text’ and some other columns which relate to specific identities. We have taken only those identities which have at least more than 500 comments in our data. The ‘target’ and other identity columns have fractional values in the range of 0 to 1. These values were decided by raters who rated the comment’s toxicity on a scale of 0 to 1 with 0 being not toxic and 1 being most toxic. Any comments with ‘target’ greater than 0.5 is regarded as toxic and others are non-toxic. For the scope of our project, we have taken toxic class as the positive class and non-toxic class is the negative class. Some of the identities that we are looking at are muslim, jewish, black, white, male, female, homosexual_gay_or_lesbian, psychiatric_or_mental_illness etc.

3 Data Preprocessing

Data was collected from several social media websites by the platform and hence was not clean. We had to perform several text preprocessing steps to make our comment texts clean to perform modelling. Some of the processing steps we took were to remove web addresses and hyperlinks from text, replacing unknown punctuations from the corpus. Emojis play a huge

part in explaining context of social media posts, so we extracted the meaning of emoji and put it in our text. Also, a lot of users on social platform's use bad words by putting in stars such as "damn" becomes "d*mn" so we created a dictionary to match those starred words to actual words. Apart from that we removed contractions to their actual words and removed digits.

4 Topic Modelling via Latent Dirichlet Allocation

Latent Dirichlet allocation is one of the most common algorithms for topic modeling. It is guided by two main principles –

- Every document is a mixture of topics
- Every topic is a mixture of words

LDA is a mathematical method for estimating both at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document.

We fit an initial LDA topic model (no. of topics = 10) on comment text and were able to visualize the clusters of topics determined by the model. Figure 1 defines the clusters we found. Inter-topic separation or distance between clusters is good for topics 1, 2, 3 and 4 but there is significant overlap in topics 5, 6, 7, 8, 9 and 10.

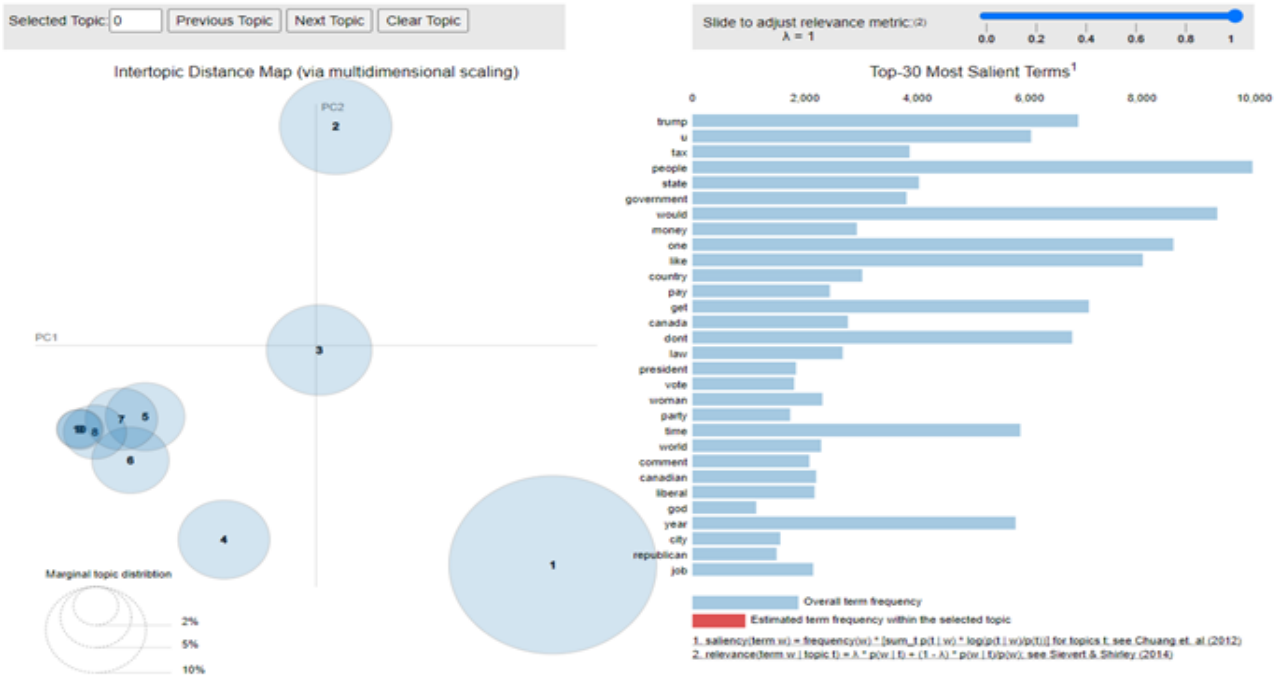

Figure 1: Clusters of topics in initial LDA model

To reduce this overlap, we determined the optimum number of topics for this corpus by performing a grid search over number of topics ranging from 1 to 14. The results showed us that the optimum number of topics is 2, since topic coherence is very high at this level and the average topic overlap is lower when compared to other levels. The results are shown in Figure 2.

When number of topics is equal to two, we achieved no overlap between topics but the results seemed counterintuitive for the scope of our project. We wanted to examine what's behind the scenes in comment text so we stuck with our initial topic model where number of topics is ten for encapsulating a wider range of topics while also offering decent cluster separation for few topics. The results for words in our clusters are shown in Figure 3.

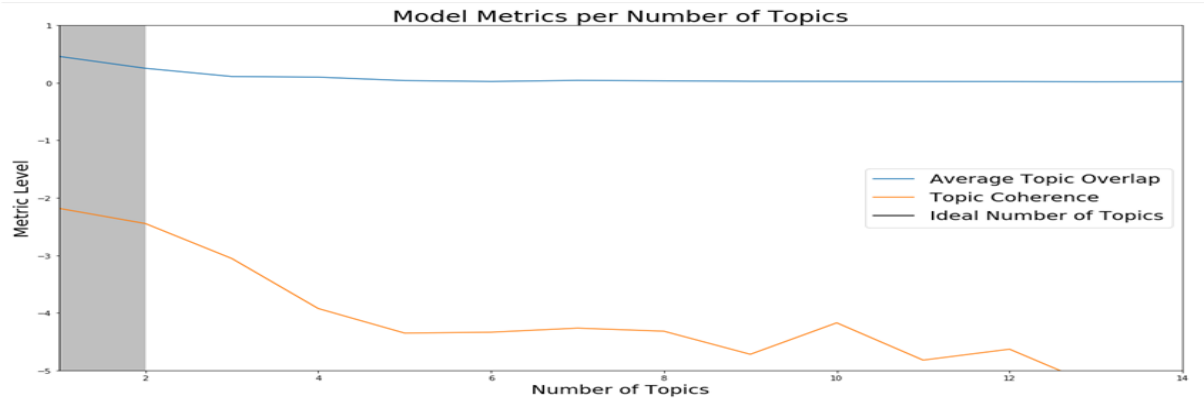

Figure 2: Topic coherence and average topic overlap values of the grid search

```
------ Topic 0 ------
alaska fund control hawaii coming alaskan sex john funny native

------ Topic 1 ------
god common sexual break global ignore gave heart language limit

------ Topic 2 ------
tax money pay care business cost million system company service

------ Topic 3 ------
law woman comment believe man find read post white must

------ Topic 4 ------
trump president news clinton medium lie hillary russian campaign evidence

------ Topic 5 ------
immigration wage worker sale demand wanted percent park healthcare province

------ Topic 6 ------
city oil using area industry thank ive trade water food

------ Topic 7 ------
vote party republican election conservative political member voter democrat attack

------ Topic 8 ------
u state government country canada world canadian liberal job american

------ Topic 9 ------
people would one like get dont time year need know
```

Figure 3: Important words from each topic in the initial topic model

From Figure 3, you can see the important words from each topic in the initial topic model. This LDA topic model gives us better insights into the comment text than the regular bigram analysis. You can see topic 0 might be related to native people of Alaska and Hawaii, topic 2 might be related to money and taxes that go into companies and businesses and topic 3 might be related to the 2016 elections and the alleged Russian interference. We can have similar interpretations for the rest of the topics.

5 Models and Metrics used for Inference

For classifying toxicity of comment text, we used common classification models like Logistic Regression with and without sampling, Naive Bayes with sampling and more complex models like LSTM with CNN and Glove embedding and Bidirectional LSTM (RNN). We hypothesized that LSTM with CNN layers or the Bidirectional model would give best results i.e. the best accuracy, precision, recall and f1 scores. Getting a model with best evaluation metric scores would give us the best classifier but it's not very helpful in detecting unintended bias towards people belonging to certain identities and communities.

To get a model that best identifies unintended bias, there are no industry standard metrics. But a lot of research is done on this and we followed evaluation metric which is

generalized AUC score as detailed in [2]. 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. Brief description of all three AUC's used –

Subgroup AUC: We restrict the data set to only the examples that mention the specific identity subgroup. A low value in this metric means the model does a poor job of distinguishing between toxic and non-toxic comments that mention the identity.

BPSN (Background Positive, Subgroup Negative) AUC: We restrict the test set to the non-toxic examples that mention the identity and the toxic examples that do not. A low value in this metric means that the model confuses non-toxic examples that mention the identity with toxic examples that do not, likely meaning that the model predicts higher toxicity scores than it should for non-toxic examples mentioning the identity.

BNSP (Background Negative, Subgroup Positive) AUC: Here, we restrict the test set to the toxic examples that mention the identity and the non-toxic examples that do not. A low value here means that the model confuses toxic examples that mention the identity with non-toxic examples that do not, likely meaning that the model predicts lower toxicity scores than it should for toxic examples mentioning the identity.

Generalized AUC score is defined as the following in this task:

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

In the above formula **A** is the number of submetrics which is three in our case. **ms,a** denotes the bias metric for identity subgroup s using submetric a and **wa** is the weighting for relative importance for each submetric. **Mp** is the pth power-mean function which is used to combine the per-identity Bias AUCs into one overall measure which is called the generalized mean. The formula for **Mp** is below and p is five in our case:

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

6 Description of Models and Methods

Since our dataset is very large, we created a subset of the dataset using stratified sampling as it was getting very difficult to train deep learning models with limited resources. We took around 5 million datapoints and then divided that into training and testing sets in 80:20 ratio. Also, we had a very imbalanced dataset with only 8% toxic comments, so we used sampling techniques for Non-Neural methods to improve the ratio of classes. We first performed random under sampling for non-toxic class to make the toxic class 20% of the total of non-toxic comments. Then we performed SMOTE analysis to create equal number of records for toxic and non-toxic classes.

6.1 Non-Neural Models

6.1.1 Feature Extraction with TF-IDF

We utilized term frequency-inverse document frequency to extract numerical features from the raw comment texts. We used white-spaces as token separators and combined 1-gram and 2-gram word tokens and used sublinear scaled frequency for normalization of term frequency [3]. It is given as:

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

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. Since our corpus is enormous, we selected the top 20000 words to be treated as our final features from tf-idf model.

6.1.2 Logistic Regression

We implemented a logistic regression with the tf-idf features and sampled dataset which was sampled by techniques as explained above. Logistic regression is very effective in giving baseline accuracies for two class classification problems, so we decided to go with it. We used L2 regularization and used 'saga' solver from scikit-learn to run the model.

6.1.2 Multinomial Naïve Bayes

We applied the multinomial naïve bayes model with tf-idf features. Multinomial Naive Bayes classifier assumes conditional independence of each of the features in the model and generally works well with feature vectors that we created using tf-idf.

6.2 Neural Models

For training all our neural networks, we used Adam Optimizer with learning rate 1e-2. Learning rate was decayed by a factor of 0.2 when the training loss was stuck at plateau for 3 epochs. We used batch size is 256 for training stability and models were trained for 30 epochs. We used a dropout probability of 0.2 for both embedding layer and other dropout layers. We also used CNN1D with MaxPooling1D layers as we hoped that CNN will capture the local structure of consecutive words in our text. All the places where we used CNN, we used 64 filters with kernel size of 5. For Maxpooling layer, we used pooling size of 4 and for the last layer, since it is binary classification problem, we used Sigmoid activation function and in all other places we used RELU activation function. Also, since the data was imbalanced, so we explicitly derived class weights for both classes and used them as parameters for training the models in optimization step.

6.2.1 Glove Embedding Layer

The task of the embedding layer is to map each word token into a denser representation using word level features. For this project, we investigated word embeddings GloVe300D (4) which is pretrained on Common Crawl. GloVe or "Global Vectors" is a word vector technique that captures both global statistics and local statistics of a corpus, in order to come up with word vectors. There are a lot of options for embedding layers such as word2vec, fasttext etc. but we investigated only one option here.

6.2.2 LSTM

Long short-term memory networks (LSTMs) are a special kind of RNNs designed to be capable of learning long term dependencies. LSTMs are a variation of Vanilla RNNs which is used to solve the problem of vanishing gradients in long sequences using Vanilla RNNs by introducing a hidden cell and replacing the simple update rule of the vanilla RNN with a gating mechanism. We used two approaches to train LSTM models. For the first approach, we trained two layers of LSTM using keras with each layer having 128 hidden units. We also

used another architecture proposed in [5] where we used CNN layer after the LSTM layers to get the local structure of text to our fully connected layers. The architecture is shown:

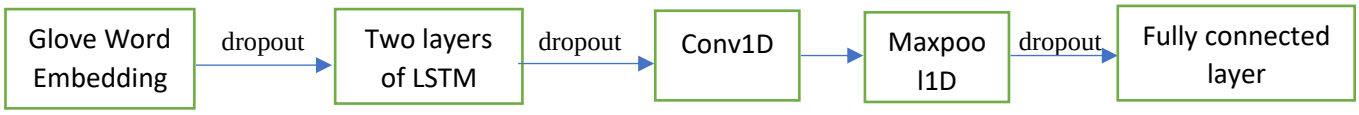

Figure 4: LSTMCNN architecture

6.2.3 GRU

Gated recurrent units (GRUs) are another form of gating mechanism in recurrent neural networks but have fewer parameters than LSTM as it has less gates. So, training GRU is faster than LSTM and in our case it gave almost similar results when compared with LSTM. We used similar approaches as LSTM on GRU networks as well.

6.2.4 Bidirectional LSTM & GRU

The dependencies within sequential data, like sentences, are not just in one direction, but may be observed in both directions. Therefore, we used BiLSTMs and BiGRUs, in which, bidirectional layers exploit the forward and backward dependencies by combining the features obtained going in both directions simultaneously. The architecture we used for bidirectional LSTMs are very similar to the original architecture which is shown below. We also tried adding a CNN layer which had very small improvement in our score.

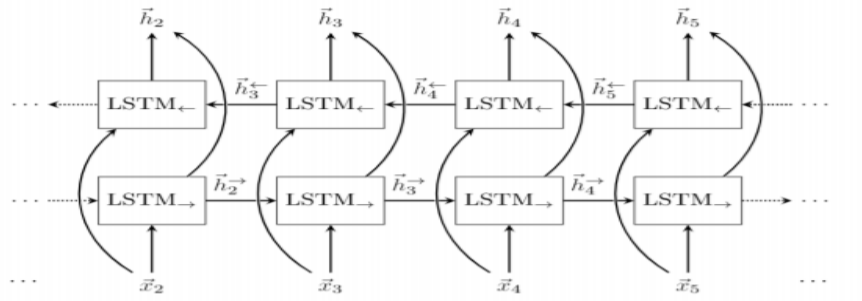

Figure 5: Bidirectional LSTM Architecture

7. Results

For non-neural methods, logistic regression achieves a better performance than multinomial naive bayes model. Naive bayes has a strong assumption of 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. Hence, Logistic regression performed better than naïve bayes which we then took as our baseline model. We also realized that sampling in non-neural models and assigning class weights in neural methods have an impact on our final score. Below table shows the results of models we used for our project.

Model	Embeddings	Generalized AUC score
Logistic Regression with TF-IDF	N/A	0.66
Logistic Regression with TF-IDF and sampling	N/A	0.69
Naive Bayes with TF-IDF and sampling	N/A	0.67
LSTM with manual Embedding	N/A	0.72
LSTMCNN	Glove	0.76
Bidirectional LSTM	Glove	0.79
Bidirectional LSTM + CNN	Glove	0.809
Bidirectional GRU + CNN	Glove	0.802

Tab. 1 Table of models and embeddings used in them along with their generalized AUC scores

For our neural methods, we saw that training embeddings from our corpus with LSTM layers did not give good result and we could see an impact of using pre-trained Glove word embeddings. Our best model is Bidirectional LSTM + CNN which gives us generalized AUC of around 0.81.

8 Analysis of Results

We analyzed the unintended bias in our model for different groups and identities. The results are shown in below Figure 6.

	SUB	BPSN	BNSP
psychiatric_or_mental_illness	0.802439	0.825355	0.849469
jewish	0.763617	0.767056	0.869096
homosexual_gay_or_lesbian	0.693333	0.648492	0.917032
black	0.656029	0.655234	0.874198
muslim	0.71714	0.699106	0.891277
white	0.695257	0.686311	0.88288
christian	0.831138	0.85446	0.849728
female	0.804615	0.821587	0.857497
male	0.806559	0.820566	0.859955

Figure 6: Bias metric scores from our best model

We see particularly bad conflation in the subgroup scores for ’black’, ’white’, ’homosexual gay or lesbian’ and ‘muslim’, partially since these labels comprise the majority of the toxic comments, which diminishes model accuracy considerably. We could also see that our model achieved relatively high BNSP scores showing that the model did a good job in identifying true toxic comments for most of these categories. Our BPSN scores were lower than our BNSP scores showing that our model predicted higher toxicity for non-toxic comments thus

increasing false negatives in the model.

9 Conclusion and Future Work

In conclusion, our neural methods outperform non-neural methods by a huge margin showing the impact that deep learning had on natural language processing. For our neural methods, although Bidirectional LSTM gave a slightly better accuracy than Bidirectional GRU but we will still want to go with GRU as we found it is much faster to train as compared to LSTM. Our error analysis shows that there is still room for improvement in our models especially for few groups which have large instances of data. So, to do that in the future, we would like to incorporate attention mechanism as it helps to highlight important words in classification tasks. We would also like to incorporate transformer based pretrained embedding layers such as BERT and see the difference in overall performance of model.

10 Contribution

- Naga Santhosh Kartheek Karnati: Exploratory Data Analysis, Topic Modelling, data preprocessing and non-neural methods.
- Shubhanshu Gupta: Data cleaning and preprocessing, Exploratory Data Analysis, Modelling for neural and non-neural methods, Error Analysis.

11 Code Repository

You can find the code for this project at: https://github.com/kartheekkarnati30/Toxic-Comment-Classification-and-Unintended-Bias

References

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

[2] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, Lucy Vasserman (2019) Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification

[3] Sublinear tf scaling - https://nlp.stanford.edu/IR-book

[4] GloVe Embedding - https://nlp.stanford.edu/projects/glove/

[5] 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.

[6] Hamida, C. B., Ge, V., & Miranda, N. (2019). Toxic Comment Classification and Unintended Bias.

[7] Huang, L. Q., & Yu, M. J. Building Unbiased Comment Toxicity Classification Model with Natural Language Processing.

[8] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.