Table of Contents

1. Introduction

1.1 Background

The problem of Toxicity in Human Language and its Prevalence in the Internet originates from multitude of online forums, in-which people participate by sharing opinions, facts and interests in the form of comments. These comments might be abusive, insulting or even hateful, thus sometimes hinder effective online discussion and hurt sentiments of individuals or groups. Toxic comments online might lead to spread of hatred and racism, attack on individuals, abuse of freedom of speech or even tension in communities.

73% of adults have witnessed online harassment, and 40% have experienced it personally as indicated in the 2014 Pew Report[1]. More often platforms struggle to effectively facilitate a conversational platform which requires appropriate moderation to limit or completely remove toxic user comments based on the nature of the comments. Thus, maintaining decorum on social media platform is the need of the hour.

1.2 Problem Statement

Toxic comment identification and prediction was published as a Kaggle competition, which seeks to use Natural Language Processing (NLP) to build a multi-headed model that's capable of evaluating comments and classify their toxicity. Machine learning models such as Convolution Neural Network (CNN) and Recurrent Neural Networks (RNN) can then generate a score of the perceived impact a comment might have on a conversation. Developers and publishers can use this score to give real time feedback to commenters, help moderators do their job efficiently, and allow readers to find relevant information more easily and effectively.

1.3 Dataset

1.3.1 Source of the dataset

Source data was downloaded from Kaggle.com "Toxic Comment Classification Challenge", which is an initiative of Conversation AI to tackle the challenge. Comments were collected from the English Wikipedia and are mostly written in English with some outliers, e.g., in Arabic, Chinese or German language[2]. The target labels for training Dataset were human rated.

1.3.2 Basic Statistics of the dataset

- The dataset consists of two splits (Training and Test) in comma delimited format.
- Number of Observation in both Train and Test dataset is approximately 150.
- Number of Variable in Train Dataset - 8 columns
- Number of Variable in Test Dataset – 2 columns
- Target – Multi-Label (6) with two class per labels ('0' and '1')
- Comments can be associated with multiple classes at once, which frame this task as a multi-label classification problem.

[1]Maeve Duggan. 2014. Online harassment. Pew Research Center.
[2]The dataset is under CC0, with the underlying comment text being governed by Wikipedia's CC-SA-3.0.

Toxic	Severe Toxic	Obscene
Threat	Insult	Identity Hate

2. Data Cleaning and Preprocessing

2.1 Data Exploration

Figure 1 shows that different classes in toxic comment are correlated. The correlation between toxic class and obscene class is 0.68 and the correlation between toxic insult and obscene class is 0.71. Severe toxic is a subset of toxic as shown in figure 2. There are 1639 severe toxic comments and they are all part of the 13699 toxic comments. Words in comments are highly similar as shown in figure 3 which again indicates that this is a multi-class classification problem.

Figure 1: Category Correlation Analysis

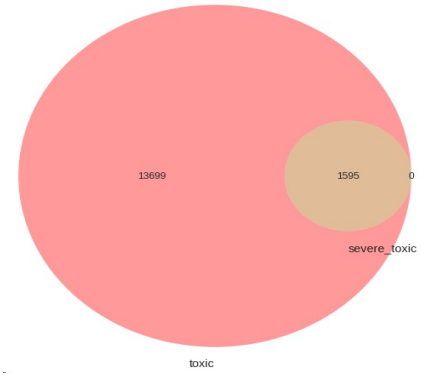

Figure 2: Category Relationship

Figure 3: Vocabulary Analysis Under Each Comment Type

Data Cleaning	Feature Engineering
○ Translate ○ Remove special characters/symbols ○ Remove numbers and punctuations ○ Convert to lower case ○ Remove stop words ○ Remove trailing and excess white space ○ Remove username and email address	○ Length of comment ○ Number of capital characters ○ Ratio of capital characters/length ○ Number of words ○ Number of unique words ○ Ratio of unique words vs words

3.Baseline Model

3.1 Undersampling imbalanced dataset

From Figure 4 below, we can see that the number of comments without label is 127,723, while the number of comments with label is 15,924, which indicate a very imbalanced dataset.

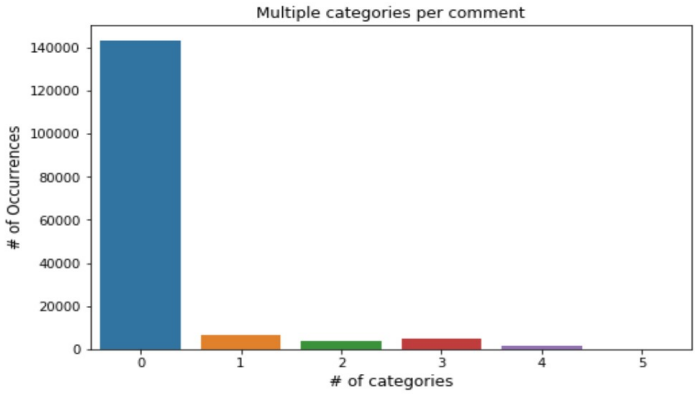

Figure 4. Imbalanced datasets before random sampling

With imbalanced data sets, the model might not get the necessary information about the minority class to make an accurate prediction. Random undersampling method is applied to randomly chooses observations from majority class which are eliminated until the data set gets balanced. In this case, comments without label are down sampled to 15,924, which is now the same as comments with labels.

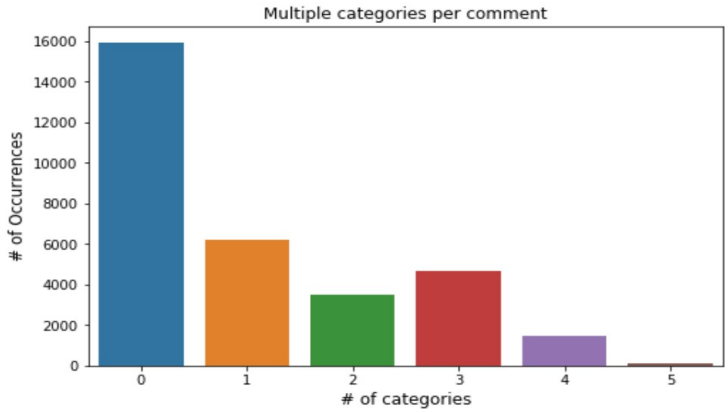

Figure 5. Balanced datasets after random sampling

3.2 TF-IDF

Term Frequency - Inverse Document Frequency (TFIDF) matrix created from comment text are used as features for logistic models. TF-IDF captures the importance of a word to the document in a corpus, and each unique word is considered as a feature. By using Scipy, TFIDF matrix is combined with new features created from feature engineering process.

3.3 Logistic Regression

Logistic regression is chosen to be the baseline model. Logistic regression allows the use of rich feature representation which can be highly interdependent, and it also provides probability estimates required.

After fitting features into the logistic model with 5-fold cross validation, the trained model predicts the probability of each type of toxicity for each comment. Each label is treated as a separate single classification problem.

Comment	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**
"*Hello everyone I'm just here to tell you that you're all freaks*"	**100%**	**1%**	**9%**	**1%**	**20%**	**3%**

3.4 Performance Analysis

Area Under Curve (AUC) is chosen to evaluate performance of baseline model. For binary classification problem, AUC gives information about how your classifier is as compared to random guesses. It is a better measure of classifier performance than accuracy because it does not bias on size of test or evaluation data. Accuracy measures the proportion of true positives and negatives in the whole data set, while AUC measures how true positive rate (recall) and false positive rate trade-off.

Label	**toxic**	**severe_toxic**	**obscene**	**threat**	**insult**	**identity_hate**
Train	0.6215	0.9505	0.7354	0.9849	0.7624	0.9553
Test	0.6237	0.9472	0.7390	0.9847	0.7590	0.9571

4.Deep Learning Model

4.1 Word Embeddings

Word embedding is the collective name for a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Word embeddings take as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space

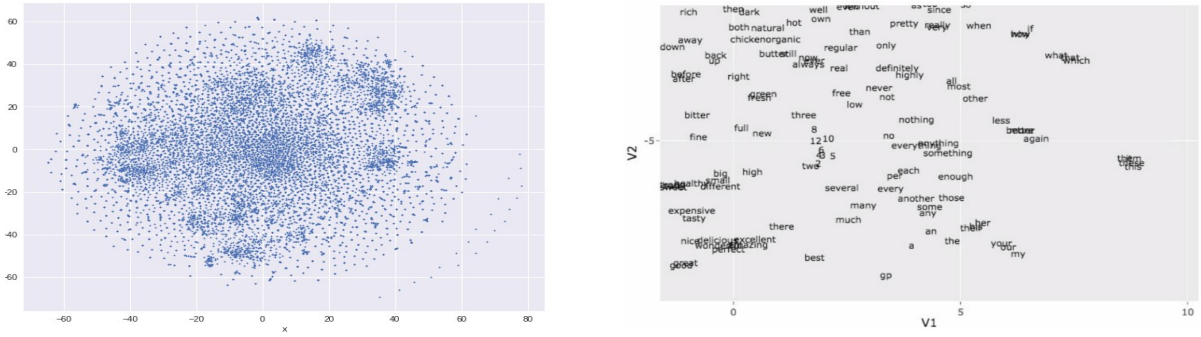

Figure 6. Visualization for Google Word2Vec embedding for top 30,000 words in the vocabulary

Data Cleaning	Word Embedding & Dimension
○ Tokenize text ○ Convert word to index ○ Pad the sequence – fixed size ○ Replace Index with Pre-trained vectors	○ Google Word2Vec – 300d ○ Stanford Glove – 300d ○ Facebook FastText – 300d

4.2 Recurrent Neural Network (RNN) with Bi-LSTM/GRU

The RNN type model used here is the Bi-LSTM and GRU model. This model is attractive because the individual cell states in the model have the ability to remove or add information to the cell state through gate layers. This is useful in practice because it allows the model to remember insights derived from words throughout the comment. Dropout was used during model training. The model will produce an output of 6 probabilities specifying whether the sentence is a toxic comment. Using a fully-connected layer, multiple neurons will be reduced to 6 outputs. A sigmoid activation is used in the layer to form a probability $\in [0,1]$. We have tried a few variations of this model using different word embeddings and architecture.

One observation from this experiment was that it didn't have much difference in accuracy whether it was LSTM or GRU units being used for the sequence layer. However, different embeddings had a noticeable difference. We tried with fastText (crawl, 300d, 2M word vectors) and GloVe (Crawl, 300d, 2.2M vocab vectors) and Word2Vec. FastText embeddings worked slightly better in this case. Below we can find the Model architecture and performance:

Model	Embed	Pooling	Activation	Optimizer	Epoch	Train Accuracy	Validation Accuracy
BI-LSTM	Word2Vec	MaxPool	Sigmoid	Adam	2	0.9814	0.9822
Bi-LSTM	**FastText**	**MaxPool**	**Sigmoid**	**Adam**	**2**	**0.9852**	**0.9844**
Bi-LSTM	Glove	MaxPool	Sigmoid	Adam	2	0.9842	0.9838
Bi-GRU	Word2Vec	Avg/Max	Sigmoid	Adam	2	0.9821	0.9826

POOLED GRU	FastText	Avg/Max	Sigmoid	Adam	2	0.9845	0.9836

We also wanted to try out on how well attention layer works out. We used an attention layer and a secondary LSTM layer. Strangely, it couldn't surpass the first two models without attention. Probably, it could have with more time spent on tuning it, but the training time needed was higher than other models.

Model	Embed	Pooling	Activation	Optimizer	Epoch	Train Accuracy	Validation Accuracy
BI-LSTM + ATTENTION	FastText	NA	Sigmoid	Adam	2	0.9835	0.9831

4.3 Convolution ID Model

The CNN model used consists of one 1-dimensional convolutional layer across the FastText word embeddings for each input comment. The convolutional layer has 128 filters with a kernel size of 5 so that each convolution will consider a window of 5-word embeddings. The next layer is a fully connected layer with 50 units which is then followed by the output layer. We also included Global MaxPool layer to get the maximum of each filter output. The intuition behind this is that with a good model, each filter can capture something from the n-grams. And the maximum value would be representative of when a neuron would be activated based on a certain input

Model	Embed	Pooling	Activation	Optimizer	Epoch	Train Accuracy	Validation Accuracy
CNN	FastText	Avg / MaxPool	Sigmoid	Adam	2	0.9846	0.9835

4.4 Stacked Model (Pooled GRU + CNN +Bi-LSTM)

One of the other architectures we tried combines convolutional with Long Term Short Term (LSTM) and PooledGRU layers, which is a special type of Recurrent Neural Networks. The promise of Bi-LSTM that it handles long sequences in a way that the network learns what to keep and what to forget.

This model was an attempt at combining sequence models Bi-LSTM and PooledGRU with convolutional neural networks (CNNs). At the beginning we tried a convolution layer passing signals to the sequence layer. But it seems swapping these layers — embeddings feeding to GRU first and then using a CNN on each LSTM unit's state — and pooling to an output layer yield better results.

Model	Embed	Pooling	Activation	Optimizer	Epoch	Train Accuracy	Validation Accuracy
Stacked (GRU + CNN +LSTM)	FastText	Avg/Max	Sigmoid	Adam	2	0.9835	0.9839

5. Analysis of Models and Bias in ML System due to Input Data

5.1 Spread of Result for Different Models

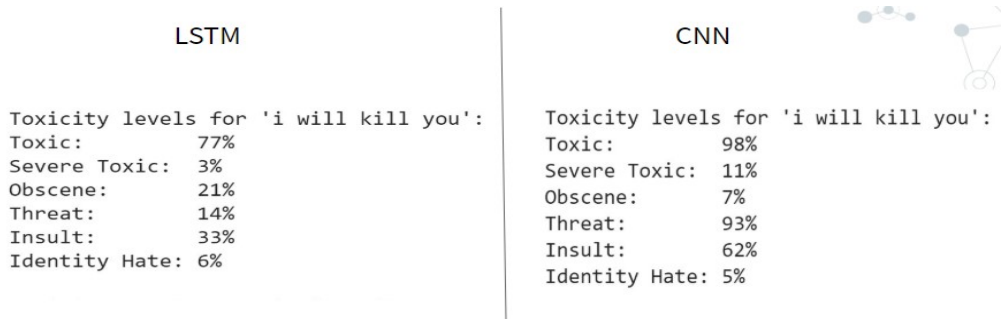

Figure 7. Class Probabilities of LSTM and CNN Model

As we can see in the above comparison, LSTM and CNN have very different output for a random twitter comment “I will kill you”. Even though, LSTM model accuracy is higher than CNN model by a small margin CNN is doing a better job at identifying non-dominant classes such as threat because it might be considering “kill you” as one unit while LSTM might not.

5.2 Bias in Input Data

This led us to believe that even the best models are not immune to characteristics of input data. We wanted to further analyze this phenomenon, hence we plotted a embedding space visualization to identify words that might not be toxic as individual units but can be regarded as toxic due to association with other words in the input data. After some exploration it was shown that the word “Life” is one such. As you can see in the below visualization life is surrounded by mostly words with negative connotation.

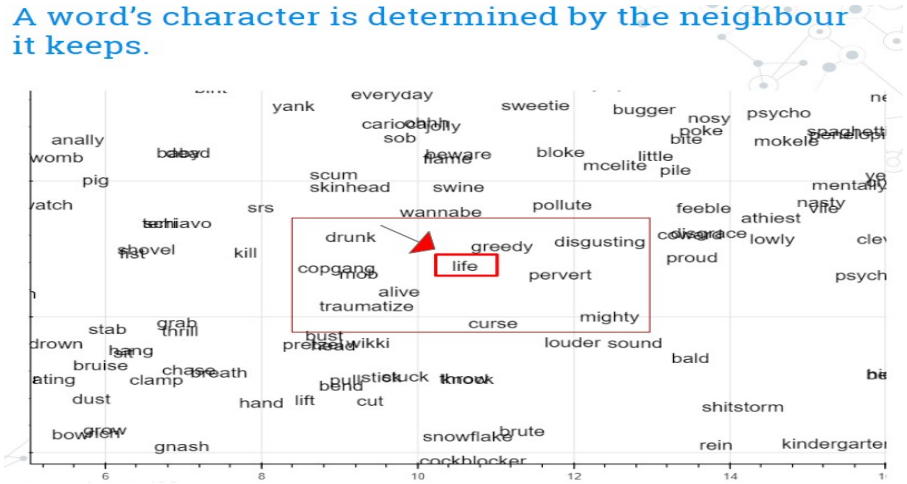

Figure 8. Embedding space **t-sne** visualization of space surrounding word “**Life**”

5.3 Explainable AI: Local Interpretable Explanation

We tested above hypothesis using one of the recent break-through in the world of Machine Learning known as “Explainable AI”. To explain our models better, we chose “Local Interpretable Model Agnostic Explanation” or LIME. We fabricated two input comments, one toxic and another non-toxic, that use the word “Life” and used Lime’s text explanation feature to explain the impact of individual words on the prediction outcome. Below are the results:

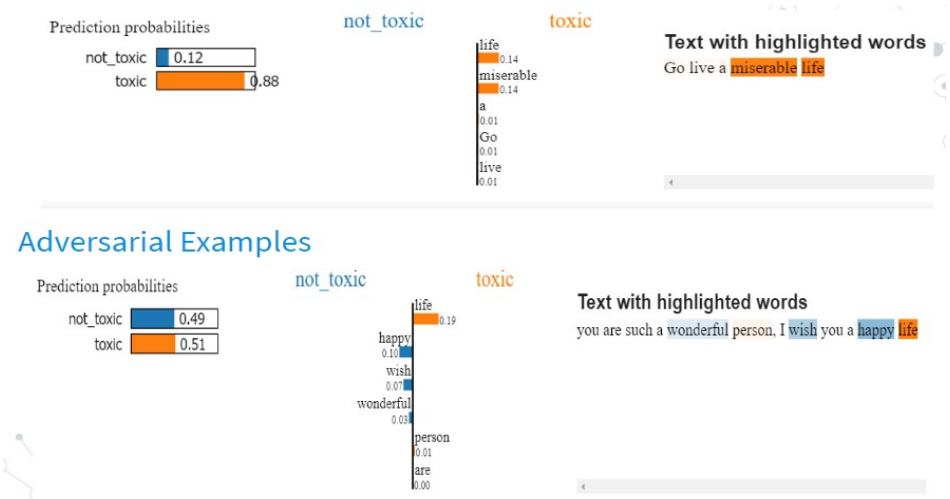

Figure 9. Model agnostic explanation of two input examples containing word "**Life**"

First example text "Go live a miserable life" is obviously toxic and identified as such. But the second example text "You are such a wonderful person, I wish you a happy life" is a very positive and uplifting comment that is identified as "toxic" because of the toxic weight associated with the word "Life".

We have already shown above in **Figure-8** that word "Life" is grouped together with many toxic words in the embedding space due to fact that word life occurs in many toxic comments. This proves our hypothesis that there is indeed a significant amount of bias exists in our machine learning solution due to the effect of input data.

6. Battling Bias in Input Data

6.1 Careful Evaluation and Attenuation of Feature Space

There are several steps that can be used to overcome the effect of input data bias and one such step is carefully curating the input data to machine learning system. But this is a very time-consuming process and the data has to evaluated by human raters for many possible biases that might exist.

6.2 Develop Models that Utilize Neighbourhood Context

Several models that can do contextual classification such as LSTM models with Attentions Layers and CNN using filters to look not just current word but also surrounding words as n-grams (2-gram, 3-gram etc..). We have already discussed and shown in **figure-7** how CNN model is able to classify non-dominant classes such as "**Threat**" much more confidently.

6.3 Universal Language Model - BERT

Utilizing new state of the art models that do positional embedding and are able to understand the meaning of word depending on the context it is used in. In fact, we have also modified BERT to do Multi Class Multi Label classification but due to lack of computational resources we were not able to train the decoder layers of such a gigantic network. (Freeze encoder - Unfreeze decoders).

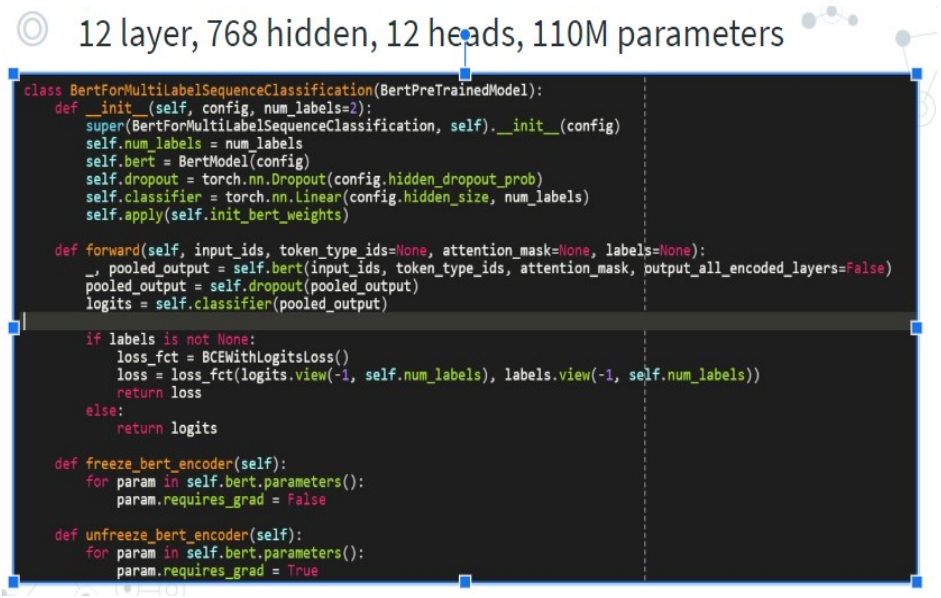

Figure 10. Modifying BERT for Multi Class Multi Label Classification

7. Conclusion

In this project we worked with several different machine learning and deep learning models with four different types of word embeddings. We also exposed how bias in input data affects the prediction of machine learning systems and discussed several possible solutions to overcome these biases. We tried to open the black-box and explain why an example is classified as toxic or not toxic. As models such as these will determine which comments get deleted or which user gets banned from a forum, it is absolutely crucial to be able to explain why a decision was taken instead of simply declaring the model output decided so. From this experience we can safely conclude that CNN model with word embedding is the best model for toxic comments classification. Although its accuracy is slightly (negligible) lower than bidirectional LSTM model, it performs much better at classifying broad spectrum of classes.