How to reduce toxicity on the internet?

ABSTRACT

In this project we want to work on the complex problem of reducing toxicity on internet. Specifically, we want to focus on identifying toxic comments from diverse internet sources. With such a model of identifying toxic comments, platforms can give users an option of hiding sensitive and toxic comments. It can also better educate users before such remarks are made by combining our model with products such as Grammarly.

We propose to treat identifying toxicity as a classification problem and implement different algorithms to compare performances. The dataset we plan to use is published by Jigsaw which is Google's subsidiary. The algorithms we will implement and compare on this data set are RNN, LSTM, GRU, Bi-directional RNN, Encoder-decoder (Seq2Seq) models and BERT.

1 Introduction

Toxicity is defined as a remark or comment which is identified defined as anything rude, disrespectful, or otherwise likely to make someone leave a discussion. Examples of such remarks could be insults, verbal sexual harassment, obscenities, discriminatory remarks, etc. When constrained to an online environment bulk of such remarks are covered in the form of comments on tweets, posts, blogs, writeups and other forms of online comments.

As the internet accessibility has increased, so has toxicity in online engagements. The 2017 Youth Risk Behavior Surveillance System (Centers for Disease Control and Prevention) estimated that 14.9% of high school students were electronically bullied in the 12 months, prior to the survey. Toxicity discourages users from engaging in discussions and indirectly hampers growth ofsustainable online community. Consequently, besides healthy interactions on online forums, reducing toxicity also has significantmonetary benefits.

WOODSTOCK'18, June, 2018, El Paso, Texas USA

https://doi.org/10.1145/1234567890

There have been various initiatives in this direction by both research and industry in recent years. For the purpose of our work, we propose to implement and compare various models from Natural Language Processing to classify an unstructured text in comments as toxic or not.

Figure 1: toxicity on internet has caused emotional traumas to not only individuals but also those holding public offices.

For our work we will use the dataset released annually by Jigsaw as subsidiary of Google. We intend to implement and compare the following models from literature for performance:

1. Recurrent Neural Network (RNN) [1]
2. Long Short-Term Memory (LSTM) [2]
3. Gated recurrent units (GRUs) [3]
4. Bi-directional Long Short-Term Memory (Bi-LSTM) [4]
5. Bi-directional Encoder Representations from (BERT) [5]

We pick these models based on relevance from the papers and literature we found and believe a comparative study would result in findings that will be helpful in identifying and building a better solution as an extension of this project. We hypothesize, such models could be used as an Open Source tool by app developers in support of anti- bullying efforts.

A challenge presents when comments are multi-lingual in same forum. While we will test the models with different languages through similar sentiment analysis but to train and compare, our prime focus will be on English language first. This is to limit scope

of project work and improvements on multi-lingual scenarios is proposed as future work.

Next, we discuss the related work that we discovered duringprob-lem research.

2 Related work

There is significant momentum in the corporations that host social media to promote healthy conversations on their platform. CEO of Twitter, Jack Dorsey, committed company's efforts for reducing toxicity on Twitter through AI/ML techniques.

Figure 2: public commitment of Twitter's CEO.

Such commitments have also been made by other platforms, such as Instagram, Facebook, Google, etc. Interest in this problem space has resulted in annual collection of database and competitions spawning off. The only solution of tackling this identification problem at scale can be through machine learning models. Below we discuss two papers from academia.

Julian Risch et. al. [7] analyze the comments sections of online news platforms because they are an essential space to express opinions and discuss important political topics. They claim that misuse by spammers, haters, and trolls makes costly content moderation necessary. Through sentiment analysis the authors proposemoder-ation and understanding the dynamics of online discussions. A subtask of content moderation proposed is the identification of toxic comments. To this end, the authors describe the concept of toxicity and characterize its subclasses. Further, they present vari-ous deep learning approaches, including datasets and architec-tures, tailored to sentiment analysis in online discussions. The au-thors treat the problem as fine grained instead of binary classifica-tion and propose to augment training data by using transfer learn-ing.

Sara Zaheri et. Al [8] propose application of Natural Language Processing techniques to classify unstructured text into toxic and non-toxic categories. The authors use LSTM as prime model and analysis of the results showed that LSTM has a 20% higher true positive rate than well-known Naive Bayes method. Their results indicated that smart use of data science can form a healthier envi-ronment for virtual societies.

Our work differentiates from the above discussions as it is more exhaustive in coverage and will also compare well performing models on other languages in limited capacity. While we are not striving to build a new model at this stage, we hope the findings from our analysis can be instrumental in further improvements of such toxicity identification.

3 Proposed work

We propose to implement and compare several models to create benchmarks for future studies and find commonalities on what kind of parameters effect such classifications. The models planned for start are RNN, LSTM, GRUs, Bi-LSTM, Encoder-decoder, BERT models. We may add some more models as our work progresses and findings refine.

The dataset used is described next.

3.1 Dataset used

The Jigsaw as subsidiary of Google releases annual dataset [9] that they generate. It contains both labelled and unlabeled data for training (223,549 rows), testing (63,812 rows) and validation (8,000 rows). The dataset contains comments with each row con-taining a different comment with following details in schema:

1. ID – the identifier of the comment
2. Comment text
3. Toxicity class, which can be one of the following:
    a. toxic
    b. severe_toxic
    c. obscene
    d. threat
    e. insult
    f. identity_hate

While the class list of toxicity is described as a multi-class in the dataset, we propose to start with a binary classification first using the different models. This is due to the skewness in dataset amongst different classes. Once a binary classification model is done, we plan to extend the work on different category for completeness.

4 Evaluation

To evaluate our models, we will use a k-fold validation on the training data and then validate manually on unlabeled data of our dataset. We will vary k as 50%, 70% and 90% to see the impact of training volume on models validating on the remaining labeled data. The validation on unlabeled data will be more qualitative in nature. We will primarily use AUC scores for benchmarking in this exercise.

The runtime consists of Google Collab Pro with 8 TPUs and total of 36GB RAM on the machine for training and testing. Any lower setup would bring challenges as most of the deep neural networks require significant RAM and compute resources (GPU and TPUs) to run and complete. This is not just to save time, but some algo-rithms would fail to run on lower amount of resources, particular-ly the RAM and Cache per TPU/GPU.

For completeness we briefly discuss the models to evaluate below. All the models are variants of artificial neural networks which are known to work well on text and language processing applications.

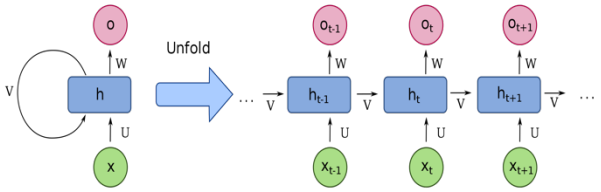
Just like our brains are connected with thousands of neurons to process any information and respond to it automatically, an artificial neural network is somewhat a similar structure. It is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria. It does need retraining for new data. Most neural networks are data hungry and hence emerge as a new category of Deep learning algorithms.

Next we discuss the different kinds of neural networks we are going to use for our project and their brief overview. The definitions and figures are taken from different sources online such as Wikipedia.

4.1. RNN or Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. In other words, in traditional neural networks, all the inputs, and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words (as the next word will depend on your previous input). Example you watch a movie and in between you stop to predict the ending, it will depend on how much you have seen it already and what context has come up yet. This is where RNN is useful as it remembers everything. It solves this problem of traditional neural network with a hidden layer.

This is the simplest model we have considered to benchmark. Below is a diagram (source Wikipedia) that represents an RNN structure.

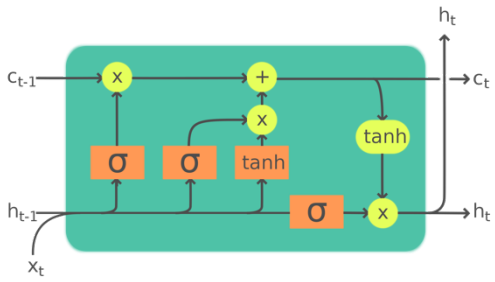


4.2. LSTM or Long Short-Term Memory

Long short-term memory (LSTM) is an variant of the artificial recurrent neural network (RNN) architecture. It is primarily used in the field of deep learning and unlike standard feedforward neural networks such as RNNs, LSTMs have feedback connections.

It can process not only single data points (such as images), but also entire sequences of data (such as speech or video) which make it useful for processing sequence of texts or even timeseries.

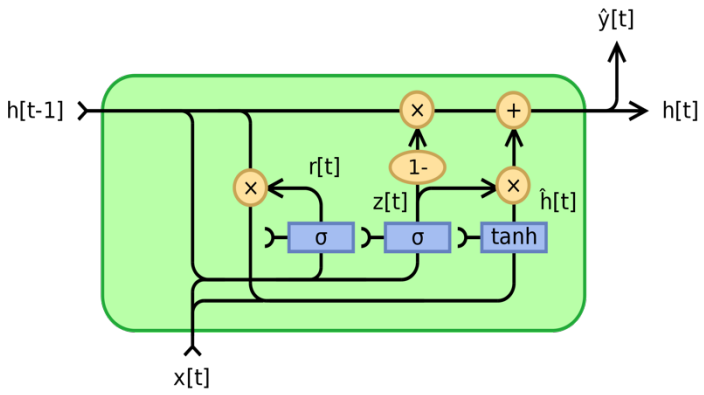
A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.



4.3 Gated Recurrent Units or GRUs

Gated recurrent units (GRUs) are also a kind of RNN with a gating mechanism in recurrent neural networks, introduced in 2014. The GRU is also similar to a long short-term memory (LSTM) but with an additional *forget* gate. It also has fewer parameters than LSTM, as it lacks an output gate.

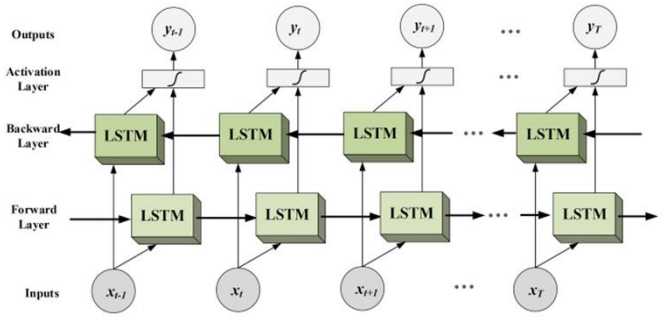
GRU's performance on certain tasks of phonetic modeling for speech recognition, speech signal modeling and natural language processing was found to be similar to that of LSTM but in some cases GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets. So it is a good candidate for our benchmarking exercise. Below is an example of a fully gated version of GRU from Wikipedia.



4.4 Bi-directional Long Short-Term Memory (Bi-LSTM)

Bidirectional LSTMs are also a variant of RNNs/LSTMs and and are really just putting two independent LSTMs put together. This structure allows the networks to have both backward and forward information about the sequence at every time step creating a better feedback loop. Using bidirectional will run inputs in two ways, one from past to future and one from future to past. It differs in approach from unidirectional LSTM as it runs backward, preserves information from the future and using the two hidden states combining it is feasible in any point in time to preserve information from both past and future. In other words they preserve the context very well and a more complex NLP application can benefit from it.

A simplified structure of the BiLSTM is shown below (source https://www.i2tutorials.com/deep-dive-into-bidirectional-lstm/).



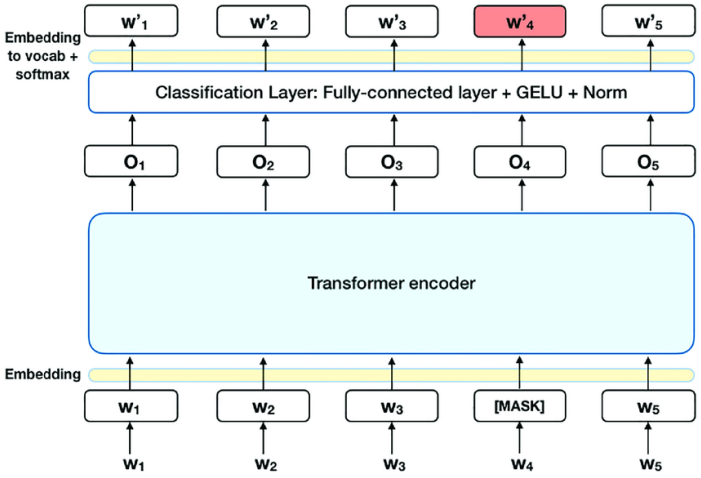
4.5. Bidirectional Encoder Representations from Transformations or BERT

BERT has its origins from pre-training contextual representations including Semi-supervised Sequence Learning and Generative Pre-Training. Unlike previous models, BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. In 2019, Google announced that it had begun leveraging BERT in its search engine, and by late 2020 it was using BERT in almost every English-language query. A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model. Indeed, BERT is the most in news model for NLP classifications these years.

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its simplest form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

Further, as opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

A simplified structure of BERT network is presented below for understanding.



4.6 Different layers for model implementations

To do proper benchmarking of the kernel of each model, we will use a consistent layer strategy for designing the models.

We will use TensorFlow for designing each model and use Keras on top of it for easier designing. Each model design follows the following strategy:

1. **Input layer:** consists of the *Embedding* class from Keras which consists of all the tokens from the training data set. These tokens are generated from the comments text of all the training rows. The output of the embeddings layer will be 300 dimensions and fed into the next layer.
2. **Middle Layers:** consists of the different layers of the model under consideration. Each of the models we are implementing are available to experiment and design in TensorFlow/Keras APIs. The middle layers will emit output of 100 sized vector for simpler RNNs / LSTMs and 300 sized vector complex GRUs/ BiLSTMs/ BERT.
3. **Output Layer:** The final layer is a sigmoid Dense model from TensorFlow which receives the output from the last layer and then emits a 0 or 1 based on if the comment is toxic or not.

The training happens on the data with epoch value of 5 and then we compute the AOC for each model to compare the performance.

Each model is also trained on TPUs as during experimentation we found CPUs and GPUs were not sufficient to train on the size of data we have at hand. Even with the extra resources we trimmed down the original dataset to 1/5 size for meaningful experiment results.

5. Results

A lot of energy has been given to data exploration and absorbing its parameters. With the size of the data, it was critical step before we implemented any models for classification purposes. Some of our findings are below:

5.1 Data asymmetry and classes

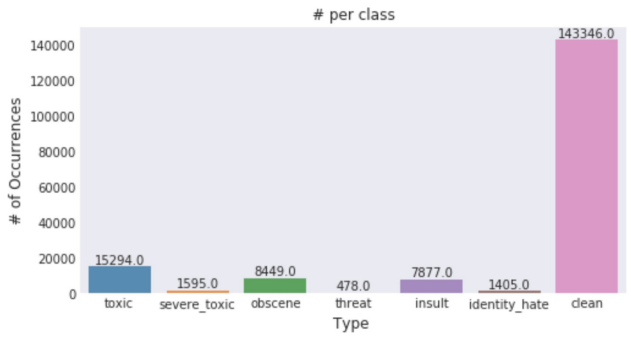


Figure 3: comments per class in the data and the skewness

The data set consists of total 159,571comments, but out of these 143,346 are clean and only 35,098 have tags. The Figure 3 shows the more granular distribution of comments tagged in classes. The toxicity is not evenly spread out across classes and is expected to be a problem that we are analyzing ways to deal with.

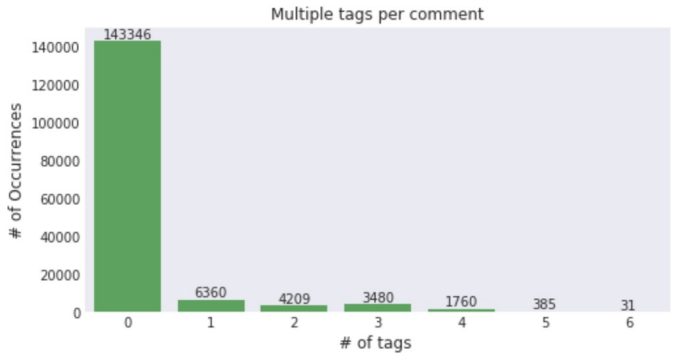


Figure 4: number of tags on comments show many having more than 1 tags per comment.

Additionally, each comment is tagged with possibly multiple tags and their distribution is shown in Figure 4. This will be managed by merging classes into 2: 'toxic' and 'not toxic'. Even with the merging of classes we expect the labelled data to be significantly skewed with respect to non-toxic comments. Part of the reason for the asymmetry is the challenge of labelling data. However, we will take normalization strategies into our models to tackle this asymmetry.

An additional analysis we did was comparing the distribution of comments tags correlating classes. To do this we constructed the crosstab/confusion matrix of toxic comments label with the other classes. In Figure 5. One interesting observations from the table is that a severe toxic comment is always toxic and other classes also seem to be a subset of toxic barring a few exceptions. To further dig deeper we plotted correlation matrix to find which tags

	severe_toxic		obscene		threat		insult		identity_hate	
severe_toxic	0	1	0	1	0	1	0	1	0	1
toxic										
0	144277	0	143754	523	144248	29	143744	533	144174	103
1	13699	1595	7368	7926	14845	449	7950	7344	13992	1302

Figure 5: crosstab/confusion matrix of toxic comments label with the other classes

go together in the Figure 6. This confirms our approach to be valid forgrouping multiple tags for classification purposes.

Figure 6: correlation matrix of tags overlaps on comments.

5.3 Data cleaning and feature engineering

The feature engineering can be classified into 3 broad categories as described below. We will select some of these and try to build classifiers on top of them.

1. Direct features:
    a. Word frequency features:
        i. Counting features
        ii. Bigrams
        iii. Trigrams
    b. Vector distancing mapping of words (E.g. Word2Vec)

c. Sentiment scores

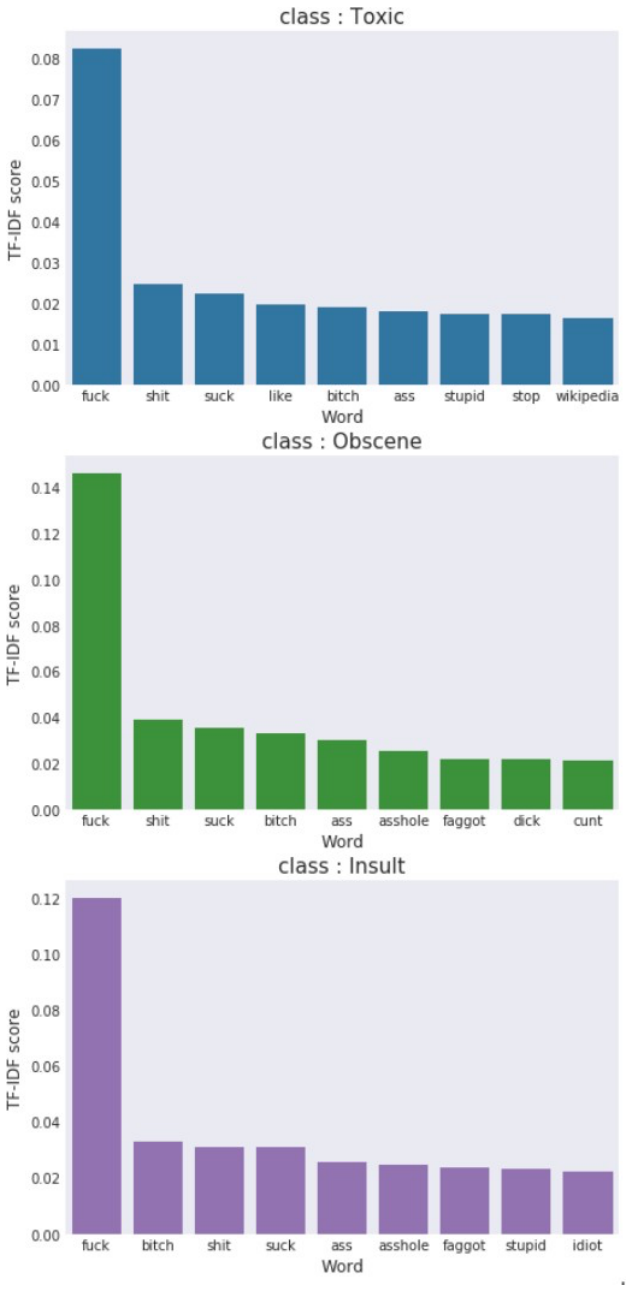


Figure 7: Unigram of classes toxic, obscene and insult.

2. Indirect features include count of:
    a. Sentences
    b. Words
    c. Unique words
    d. Letters
    e. Punctuations
    f. Stop words
    g. Average length of word in sentence
3. Leaky features:

a. These are features such as usernames, IP, other metadata that is not directly part of language and can result in overfitting.

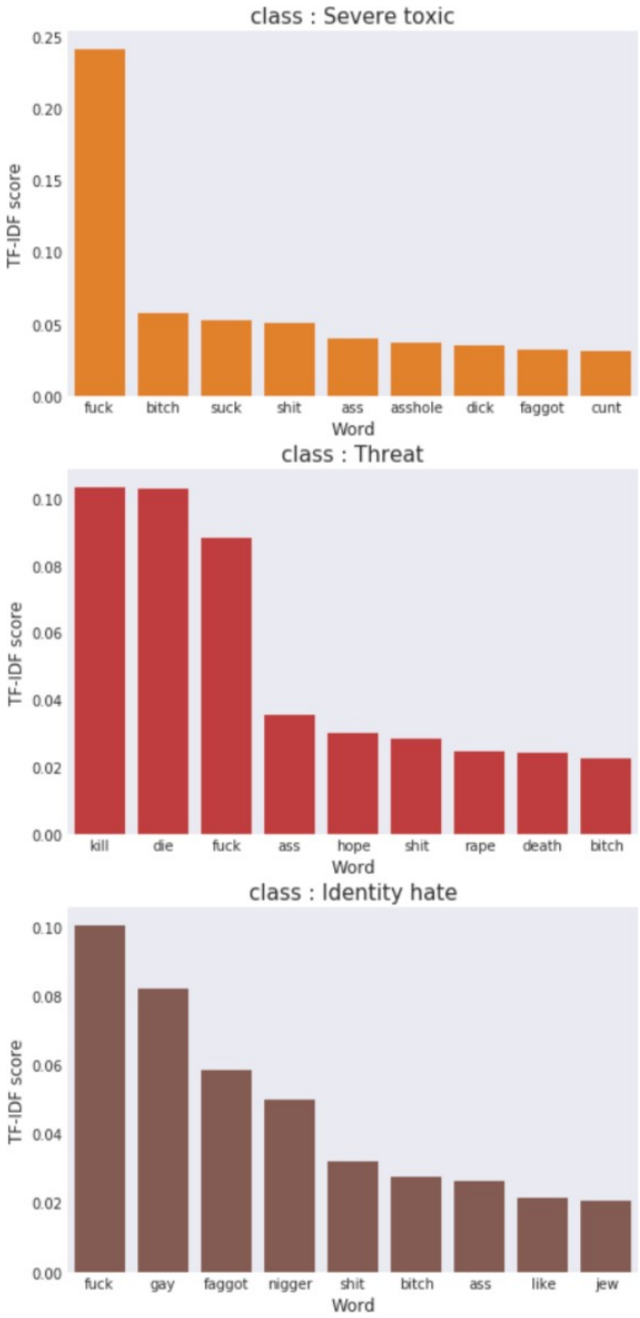


Figure 8: Unigram of classes severe toxic, identity hate and threat.

The comment texts have been scrapped and collected which make it necessary for cleaning the comments text before feature extraction and tokenization can take place. We convert all comments text to lower case, remove characters like '\n' and leaky features like IP, user id, usernames quoted. We also convert apostrophe containing words such as 'aren't' to their full expanded form that is 'are not'

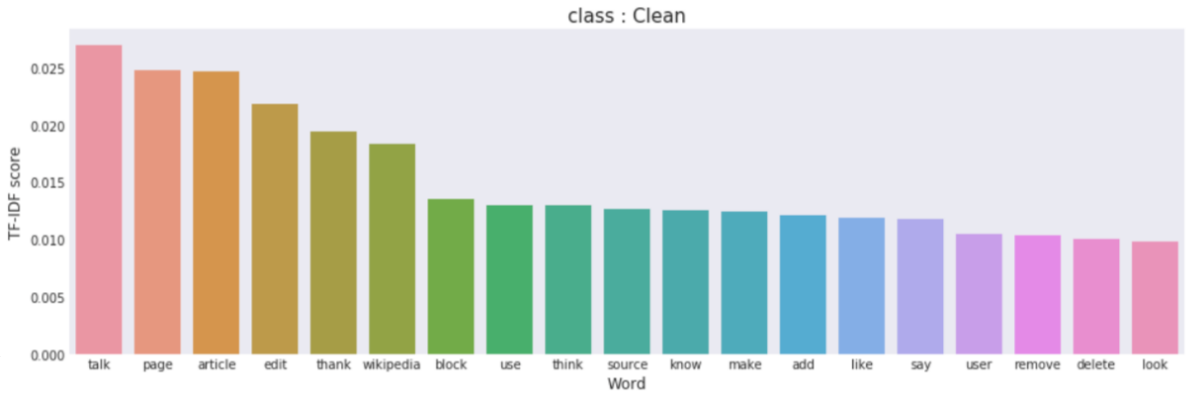


Figure 9: Unigram of clean comments in the data set.

There is clear contrast to the unigram of clean class of comments in Figure 9.

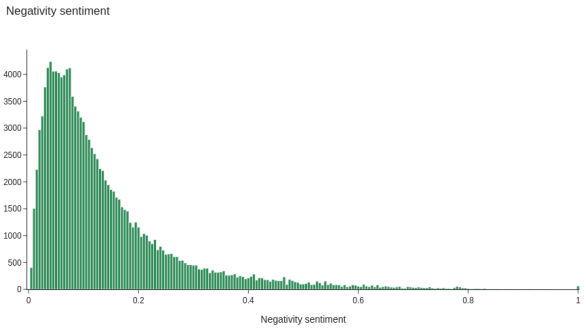
In terms of classifier models implementation which was part of our 3rd milestone we have started with some implementation of RNN models, but further work is in progress along with implementation of other models to compare and benchmark.

5 Exploratory Data Analysis.

In this section, we will be performing EDA on the dataset. We will be using the NLTK library to gather the sentiment profiles of the comments and check how they relate with the toxicity of the comment.

5.1 Negative Sentiment.

Negative sentiment refers to negative or pessimistic emotions. It is a score between 0 and 1; the greater the score, the more negative the subject is.



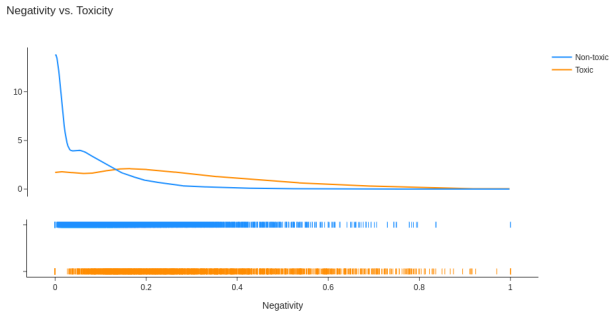
From the above plot, we can see that negative sentiment has a positive skew, indicating that negativity is usually on the lower side.

This suggests that most comments are not toxic or negative.

5.1.1 Negativity vs. Toxicity

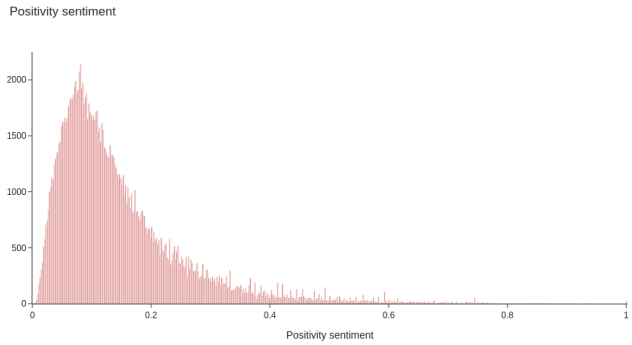
We can clearly see that toxic comments have a significantly greater negative sentiment than non-toxic comments (on average).

The probability density of negativity peaks at around 0 for non-toxic comments, while the negativity for toxic comments are minimum at this point. This suggests that a comment is very likely to be non-toxic if it has a negativity of 0.

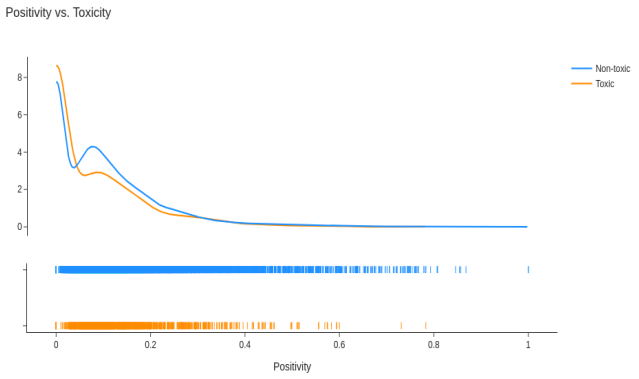


5.2 Positive Sentiment

Positive sentiment refers to positive or optimistic emotions. It is a score between 0 and 1; the greater the score, the more positive the subject is.
From the below plot, we can see that positive sentiment has a



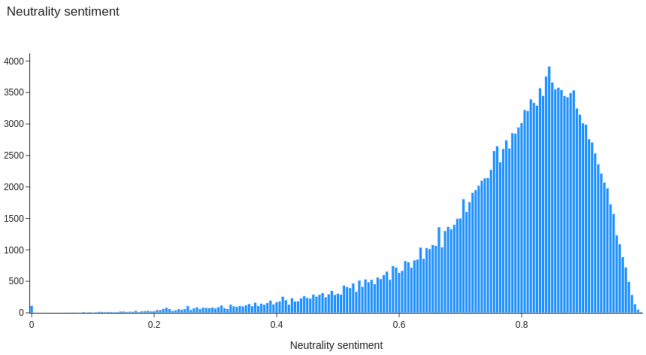
positive skew, indicating that positivity is usually on the lower side. This suggests that most comments do not express positivity explicitly.



5.2.1 Positivity vs Toxicity

Here we have plotted the distribution of positivity for toxic and non-toxic comments above. We can see that both the distributions are very similar, indicating that positivity is not an accurate indicator of toxicity in comments.

5.3 Neutrality Sentiment.



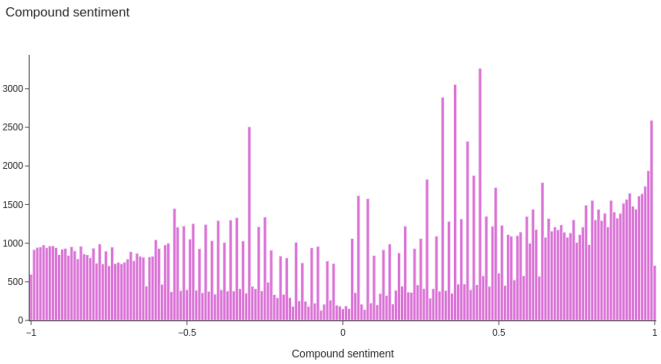
Neutrality sentiment refers to the level of bias or opinion in the text. It is a score between 0 and 1; the greater the score, the more neutral/unbiased the subject is.

From the above plot, we can see that the neutrality sentiment distribution has a negative skew, which is in contrast to the negativity and positivity sentiment distributions. This indicates that the comments tend to be very neutral and unbiased in general.

5.3.1 Neutrality vs. Toxicity

We can see that non-toxic comments tend to have a higher neutrality value than toxic comments on average. The probability density of the non-toxic distribution experiences a sudden jump at 1, and the probability density of the toxic distribution is significantly lower at the same point. This suggests that a comment with neutrality close to 1 is more likely to be non-toxic than toxic.

5.4 Compound Sentiment

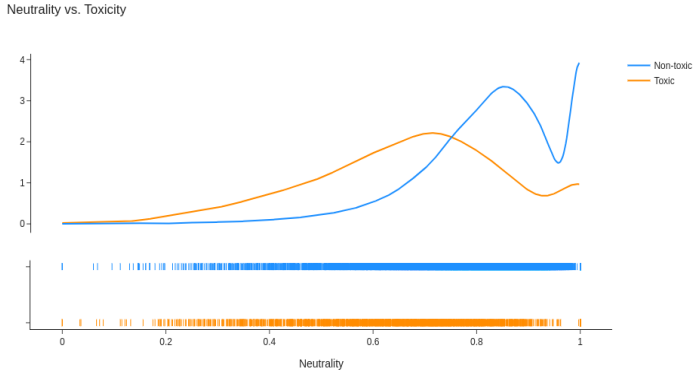


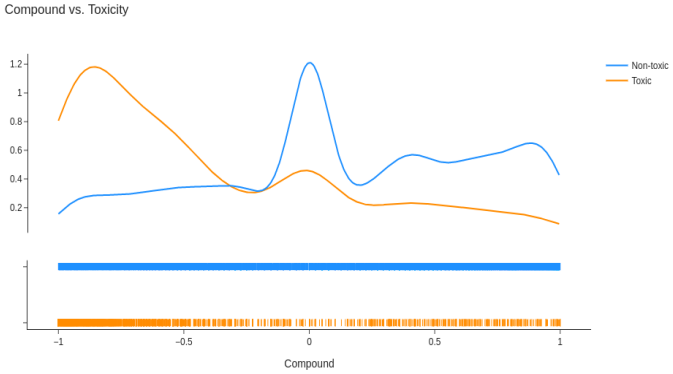
Compound sentiment refers to the total level of sentiment in the sentence. It is a score between -1 and 1; the greater the score, the more emotional the subject is.

From the distribution above, we can see that compound sentiment is evenly distributed across the spectrum (from -1 to 1) with very high variance and random peaks throughout the range.

5.4.1 Compound sentiment vs. Toxicity

We can see that compound sentiment tends to be higher for non-toxic comments as compared to toxic comments. The non-toxic distribution has a negative skew, while the toxic distribution has a positive skew. This indicates that non-toxic comments tend to have a higher compound sentiment than toxic comments on average.





6 Modeling

As we have discussed earlier, we will be treating this problem as a binary classification problem. We will be using Simple RNN, LSTM, GRU, Bi-directional LSTM and Bert as our models and evaluate the performance of different models for classifying toxic comments.

6.1 RNN

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer.

In an RNN we input a sentence word by word. We represent every word as one hot vectors of dimensions: Numbers of words in Vocab +1. The keras Tokenizer takes all the unique words in the corpus, forms a dictionary with words as keys and their number of

Model	AUC Score
Bert	0.9779
GRU	0.9736
Bidirectional LSTM	0.9662
LSTM	0.9616
RNN	0.8262

occurrences as values, it then sorts the dictionary in descending order of counts. It then assigns the first value 1 , second value 2, and so on. So, let's suppose word 'the' occurred the most in the corpus then it will assign index 1 and vector representing 'the' would be a one-hot vector with value 1 at position 1 and rest zeroes.

Our model achieves an accuracy of almost 1 which is just unexpected and indicates we are clearly over fitting. This was the simplest model of all, we can tune a lot of hyper parameters like RNN units, we can do batch normalization, dropouts etc. to get better result. We got an AUC score of *0.8262* without much efforts.

6.2 LSTM

Simple RNN's were certainly better than classical ML algorithms and gave state of the art results, but it failed to capture long term dependencies that is present in sentences. So, in 1998-99 LSTM's were introduced to counter to these drawbacks. We have already tokenized and padded our text for input to LSTM's.

As a first step we calculate embedding matrix for our vocabulary from the pre-trained GLoVe vectors. Then while building the embedding layer we pass Embedding Matrix as weights to the layer instead of training it over Vocabulary and thus we pass

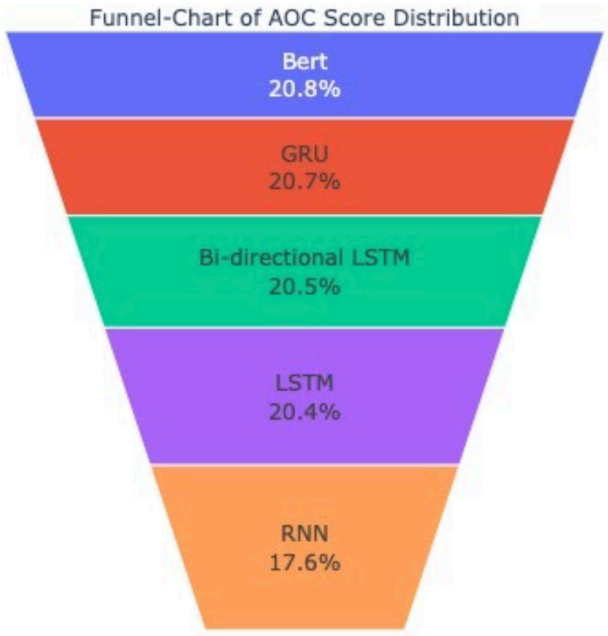
trainable = False. Rest of the model is same as before except we have replaced the Simple RNN by LSTM Units.

We observe that the model is not over fitting and achieves an AUC score of 0.9616 which is quite fair, also we close in on the gap between accuracy and AUC. We see that in this case we used dropout and prevented over fitting the data.

6.3 GRU

Introduced by Cho, et al. in 2014, GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU's are a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results. GRU's were designed to be simpler and faster than LSTM's and in most cases produce equally good results and thus there is no clear winner.

We see that the model achieves an higher AUC score of 0.9736 compared to the previous two models. Using GRU, we can see that with almost same accuracy we achieved a higher AUC when compared with LSTM.



6.4 Bi-directional LSTM

A Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm.

We have achieved a similar accuracy and AUC score as before. We

achieved an AUC of 0.9662 here.

We observed that GRU has achieved a higher AUC score among the models we have used so far. While it is important to note that these results might not translate to multi-class classification models, it is part of our future work.

6.5 Bidirectional Encoder Representations from Transformations or BERT[5]

The last model we attempted implementing and comparing for our benchmarking exercise was BERT. This was fairly challenging and so far the most resource heavy model in this exercise. It required significantly higher end TPUs and also higher amount of time (~3 mins per epoch). However, it did beat the accuracy of all other models.

The AUC Score for BERT model was 0.9779, which marginally better than GRU even in our simple case classification of toxic vs non-toxic. We believe it might be worthy future work exercise to repeat the exercise on multilingual and multi class classification as BERT might fare better in those scenarios.

8. CONCLUSION

We have dealt here with binary classification of toxicity classification of comments text. While there can be different categories of toxicity the data around each kind is very sparse in datasets available and hence converting all toxic classes to one after cleanup was necessary. Next, we used several kinds of known and existing models on the dataset from Jigsaw Inc. The exercise resulted in really good results and established while BERT outperforms other kinds of models, the time, complexity and resources needed may not justify. GRU in contrast was almost comparable to BERT And needed much less resources.

9. FUTURE WORK

The exercise also established for us the need and significance of resources to train efficient models and get results. There is a tradeoff involved in terms of accuracy vs other constraints and the problem space and application can dictate which model to choose. Here we are not claiming any one particular winner and leave the choice on application developers as one might suit better than the other.

In terms of future work, we have left out multilingual classification and multiclass toxicity classification as possible ventures to experiment with. Another future work is how to deal with the sparse classes data across different categories of toxicity.

10. ACKNOWLEDGMENTS

We want to thank Professor Di Wu and our class for the feedback and inputs during our class presentation. The well-defined structure on deliverables was very helpful for us to work efficiently too.

11. REFERENCES

[1] https://www.ibm.com/cloud/learn/recurrent-neural-networks
[2] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
[3] Rahul Dey, Fathi M. Salem, Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks, https://arxiv.org/abs/1701.05923
[4] Schuster, Mike & Paliwal, Kuldip. (1997). Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on. 45. 2673 - 2681. 10.1109/78.650093.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805
[6] https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346
[7] Risch, Julian & Krestel, Ralf. (2020). Toxic Comment Detection in Online Discussions. 10.1007/978-981-15-1216-2_4.
[8] Zaheri, Sara; Leath, Jeff; and Stroud, David (2020) "Toxic CommentClassification," SMU Data Science Review: Vol. 3 : No. 1 , Article 13.
[9] https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification/data