A Term Paper on Toxic Content Detection on Social Media

Mithil Dave
North Carolina State University
mdave2@ncsu.edu

Krishna Prasanna Bhamidipati
North Carolina State University
kbhamid2@ncsu.edu

ABSTRACT

Social media caters to populations from different ethnicities, social backgrounds, age groups and sexual orientations. Billions of people interact through these platforms through video, audio and textual medium. In these interactions, words are often exchanged between parties which may not hold any significance to one party but may inflict trauma on the other party, therefore restricting a constructive exchange of thoughts and ideas.

Studies now show that online toxicity such as trolling, badgering, personal attacks, abuse and bullying have a significantly deteriorating effect on mental health - particularly on the younger population. Hence it has become necessary to monitor and control the prevalence of toxic content on social media, and create a safe space for all people to participate without feeling bullied or marginalized.

1 PROBLEM STATEMENT

Manually identifying toxic content and flagging/removing them is a tedious and challenging task. Such exercise is time consuming, inefficient and not scalable.

Hence, in this project, we seek to detect toxic content in textual conversations sourced from social media platforms using natural language processing techniques, and perform a multi-label classification of the content into different sub-categories of toxicity.

Through our work we aim to understand the patterns that define toxic texts and detect the groups that they are most directed to, by looking at the sentiment polarity and specific set of curse-words associated with each category.

2 HYPOTHESIS

2.1 Profanity with positive sentiment

: We hypothesize that use of profanity in a sentence does not always infer a negative sentiment. For example, “Shit, this is so good!”. Here, although there is a profanity present in the sentence, the overall sentiment of the sentence is positive. To handle these types of sentences, we will check the frequency of occurrence of such profane words with a positive sentiment in our labelled dataset. This will entail classification of sentences as positive or negative using sentiment analysis techniques. Since sentiment classification in the context of this project is a relatively minor task, we seek to implement standard pre-trained models to accomplish it and assign sentiment tags to our sentences.

2.2 Classifier Performance

: We hypothesize that pre-trained context-based classifier fine-tuned to our toxicity classification task offers a better performance for selected evaluation metric when compared to traditional context-agnostic classifier. We seek to statistically establish whether this stands to be the case using hypothesis testing. Since the comparison may not be completely fair (based on "documented" superiority of context-based models over traditional), we seek to normalize our performance scores against execution times of both models and comment on this trade-off in the context of evaluation scores.

3 DATASET

This project is inspired from Kaggle’s Toxic Comment Classification challenge and Conversation AI team which aims to protect voices in conversation. The dataset being used has been created from Wikipedia comments released under CC0 license and have been labeled by human raters for toxic behavior.

The original dataset is class-imbalanced consisting 159,571 comments with 6 associated labels indicating six levels of toxicity such as toxic, severe-toxic, obscene, threat, insult and identity hate. These labels take a value of 0 or 1 and this dataset supports multi-label classification.

We have reduced the size of our dataset to 20000 comments, half of which are labeled as toxic and other half are non-toxic. The reason for size reduction in data is two-fold:

(1) Downstream sentence embedding task is extremely computationally expensive and totaled to more than 6 hours of training, prompting us to explore smaller dataset.
(2) Class-imbalance of data has been major challenge, with less than 12.5% of data having actual class labels.

4 APPROACH

For this task, we implemented two types of models: A traditional, context-independent model and a transformer-architecture based,context-dependent model.

In the following subsections we provide a detailed explanation of data pre-processing, exploratory analysis, traditional and context-based model constructions and implementation.

4.1 Pre-Processing

4.1.1 ***Data-Cleaning:*** Data cleaning is required to boost the performance of the prediction models.[3] By examining the top 100 comments the following features were identified for removal from the text using regular expression matches:

- /n characters: There are many new line symbols present in the text, in some instances where they were not warranted and add no semantic value.
- Image: There are special references to images, along with image URLs and png extensions in the text. In terms of actual language, they don’t add any semantic value.
- IP addresses, http links: There are several links present in the data referring to specific users, which are difficult for NLP tools to parse.

- Wiki links: There are links to Wikipedia pages or articles which are not of any use.
- Date related fields: Date and time format cause trouble during the sentence embedding. We have observed that most date mentions refer to the date on which comment was posted or edited or referenced in another comment.
- Unnecessary punctuation: Exaggerated use of punctuation was observed in dataset for emphasis or due to formatting issues. Symbols and characters such as '.', '?', ':', '(', '!' etc were removed for efficient modeling.

4.1.2 ***Spelling correction:*** After the preliminary data cleaning is done, spelling correction was performed on the text.

- Peter Norvig's Google inspired spelling correction technique was implemented. This technique uses text from project Gutenberg's ebooks corpus and calculates the probability of each word in the Gutenberg text.
- It then performs spelling correction of words in our comments by searching for a set of words from this corpus that are within 1 or 2 edit distances of our 'incorrect' word by considering all permutations (insertions, deletions, replacements, and transpositions). This function uses a 'Levenshtein Distance' algorithm to find permutations.
- It then picks the word that has highest probability as calculated from Gutenberg corpus.

4.1.3 ***Lower-casing and stop-word removal:*** Data cleaning typically also includes removing the stop-words and handling of the upper case letters. Here, these tasks HAVE NOT BEEN performed for reasons stated below:

(1) Stop words have NOT been removed, because we are using a context-based embedding technique which relies on stop words to build proper context.
(2) Words have NOT been converted to lower-case because we believe that some places, where capital words have been used, reflect emphasis that the speaker want to convey. For example 'BAD!!' symbolizes more negative sentence than 'bad'.

4.2 Exploratory Data Analysis

4.2.1 ***Correlation between label occurrences.*** : Correlation scores between different labels was calculated and Figure 1 shows the correlation matrix for the dataset labels.

By looking at the correlation matrix in figure 1, we see a strong positive correlation (>0.6) the following pairs of labels:

- Insult and Toxic
- Obscene and Toxic
- Obscene and Insult

4.2.2 ***Label-wise distributions:*** The label-wise count of toxic comment instances in the dataset of 20000 comments was determined and Figure 2 shows the distribution of label counts:

The data-set contains a low percentage of 'severe_toxic', 'identity_hate' and 'threat' type comments. These labels will be challenging to predict correctly given their lower representation.

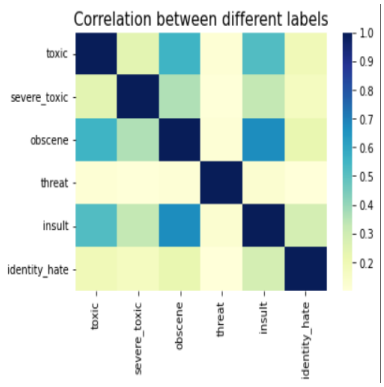

Figure 1: Correlation Heatmap - Pearson Coefficient

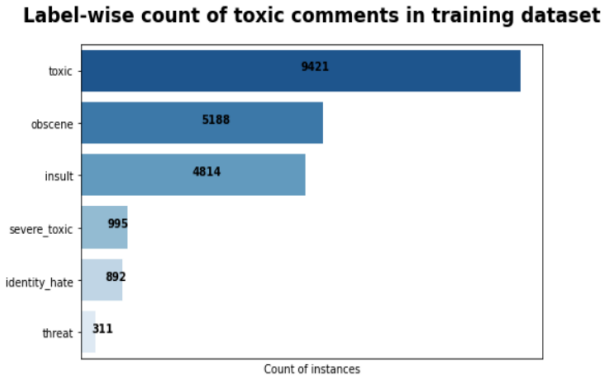

Figure 2: Label-wise distribution of toxic comments

4.2.3 ***Sentiment Analysis.*** To determine the sentiment polarity of the comments and find out the correlation between this polarity strength and the labels, sentiment analyzer VADER was used.[4] VADER predicts the positive or negative sentiment attached to text in terms of a sentiment score. Figure 3 shows the polarity score distribution for each label. Figure 4 shows the over-all sentiment polarity distribution for toxic vs non-toxic comments, toxic meaning that the sentence belongs to at-least one of the 6 class labels.

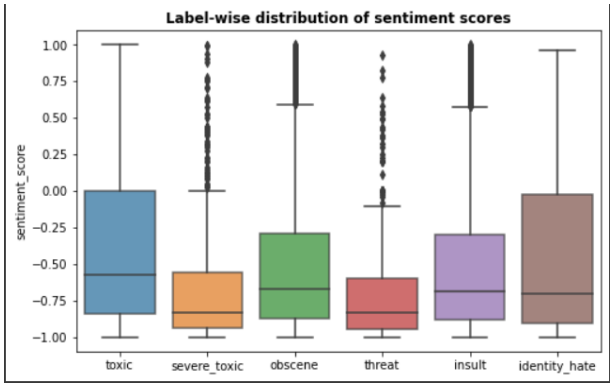

Figure 3: Label wise distribution of the sentiment score

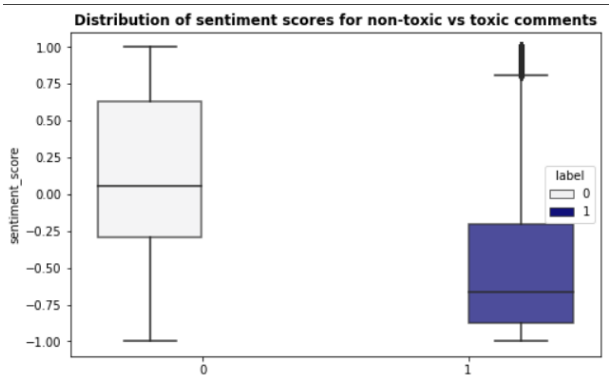

Figure 4: Sentiment score distribution for toxic vs non-toxic comments

Sentences which are labeled as non-toxic have a comparatively positive average sentiment (0 or more), in comparison to those which take on at-least one of the six labels. Such sentences have an average negative sentiment polarity of -0.75.

4.2.4 ***Curse Word Analysis.*** Google's Profanity Words list (link: curse_word_list) was used to check if the text contains any curse-words. The below scatter plot shows the proportion of sentences belonging to one of the 6 categories having curse words.

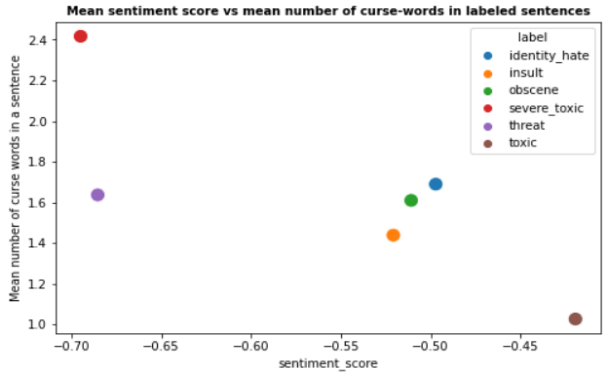

Figure 5: Sentiment score vs percentage of labeled sentences with curse-words across each category

We observe that higher the negative polarity, higher the probability that the sentence has curse word. This pattern is mostly consistent with all categories, except for "threat" class label, which has low proportion of curse words despite its high negative sentiment polarity.

4.2.5 ***Word-Cloud.*** : A very quick way to check the frequently occurring words in the dataset is through word-cloud. We used python's WordCloud library to find out most common words corresponding to every label in our dataset. The WordCloud for our "Threat label" is shown in the figure 6.

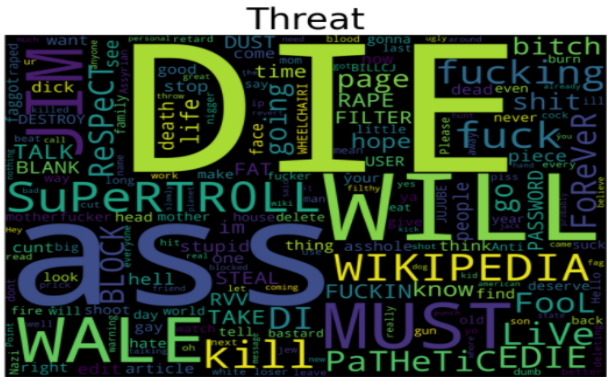

Figure 6: WordCloud for the "Threat" labeled comments

4.3 Context-independent Classification Model

Our context-independent classification model comprises of NLTK word tokenizer, GloVe based-sentence embeddings, OneVsRest classifier and classifier Chain for multi-label classification.[7][2]

4.3.1 ***Tokenization:*** We use NLTK's word_tokenizer method to split a sentence into word tokens. NLTK (Natural Language ToolKit) is a library written in Python for symbolic and statistical Natural Language Processing. We noticed that this word tokenizer considers punctuation as tokens which might create trouble while performing the word embeddings. Thus we cleaned the text before feeding it to the tokenizer.

4.3.2 ***Stop Words:*** In a contextual sentence, stopwords usually present an important role in formulating the meaning of the text. But In this context-independent model, we do not consider context of the sentences during the embeddings. Thus, stop words do not hold any meaning in the context-free approach. We used NLTK's stop word list to remove stop words from the dataset and we extended this list to contain other stopwords seen in our dataset, for example "wikipedia", "link", "page" etc.

4.3.3 ***Sentence Embeddings:*** GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The underlying aspect of this algorithm is that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. Hence the GloVe model is trained over the non zero values of the global word-word co-occurrence matrix.

The effectiveness of these embedding depends upon the choice of the training dataset. For example, a pre-trained model on a Wikipedia dump will not perform well for Reddit because the lingo is quite different. Here we use Glove's twitter dataset (glove-twitter-25) which contains 2B tweets, 27B tokens, and 1.2M vocabularies for training. This dataset is quite similar to our dataset in terms of structure and language.

4.3.4 ***OneVsRest Classifer:*** Traditional multi class problems can be cast into multi-label ones by restricting each instance to have only one label. These problems do not consider correlation of the labels while classifying. OneVsRest classifer has a strategy of fitting

one classifier per class to do multi-class and multi-label classification. The aim is to solve multi-label problem by decomposing it into multiple independent binary classification problems.[2]

In "OneVsRest" approach, there are multiple independent classifiers doing the labeling. For a new unseen test data point, it chooses the class for which the confidence is maximum. Here, the assumption is that the labels are mutually exclusive. We do not consider any correlation between the labels in this method. In our case, the data labels seem to have very less correlation. But for a highly unlabeled dataset, this classifier may fail due to extensive overfitting.

4.3.5 ***Classifier chain:*** The binary relevance method of multi-label classification is scalable and quite efficient for the large datasets. But they do not consider the correlation of the labels during predictions. We used chain classifiers which can model label correlations while maintaining acceptable computational complexity.

In this method, a chain of binary classifiers C0, C1, ... Cn is constructed. The classifiers are linked along a chain where each classifier deals with the binary relevance problem associated with a single label L[j] L. The feature space of each link in the chain is extended with the predicted label associations of all previous classifiers, i.e., the classifier chain links together all binary classifiers in a chain structure, such that class label predictions from previous steps is used as features for the next classification step.[7] [6]

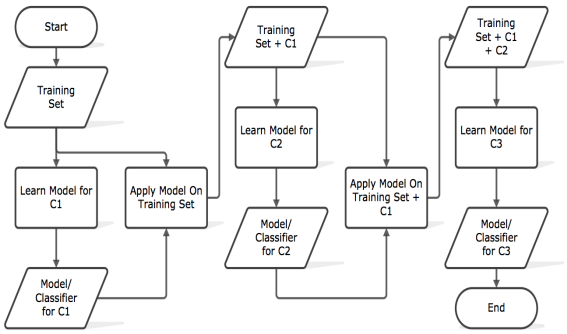

Figure 7: Architecture of a Chain Classifier (n=3)

In this way, it passes the label information between the classifiers. It allows us to take into account label correlations and thus overcome the problem of label independence. The chaining methodology indicates that multiple classifiers are not parallelly executable.

The order of the classifiers in the chain affects the output. To find the best possible order of the chain, different ensemble methods can be used.

Serial execution allow CC to process a single binary problem at a time and retain the advantages of binary relevance method of having low memory and low run-time complexity.

4.4 Context-based Classification Model

The main challenge with traditional sentence embedding techniques is their limited ability to model a sequenced representation of words and prioritizing context-defining words.

Context-based classification models leverage context-driven embeddings which can then be fed into a down-stream task. These context-driven embedding techniques may be unidirectional or bi-directional. Unidirectional models have shown sub-optimal performance for sentence-level tasks.

4.4.1 ***BERT:.*** Bi-directional models have a masked-language-model (MLM) pre-training objective, which enables the representation to fuse the left and the right context. BERT is one such state-of-the-art encoder-based, deep-bidirectional transformer model developed by Google whose pre-trained model can be fine-tuned using user's data for a wide range of down-stream tasks.

BERT out-performs on 11 NLP tasks including pushing the GLUE score to 80.5%,MultiNLI accuracy to 86.7%, SQuAD v1.1 question answering Test F1 to 93.2 and SQuAD v2.0 Test F1 to 83.1

4.4.2 ***distilBERT:.*** distilBERT, developed by the Hugging Face team, is a smaller, faster, cheaper and lighter version of BERT transformer architecture. It has only 3% degradation with respect to BERT's performance and pre-trained model consumes only a fourth of BERT's computational resources. [5]

4.4.3 ***Tokenization:*** The default tokenizer used by distilBERT models is a sub-word tokenizer called WordPiece. Working of sub-word tokenization has been illustrated in Figure 8. Hugging Face allows us to load the custom tokenizer specially designed for pre-trained distilBERT model from the transformer's class.[9]

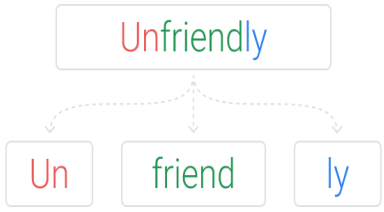

Figure 8: An example of sub-word tokenization

This tokenizer is applied on each comment sentence, and it generates two fields - An input ID which corresponds to token's ID as defined for pre-trained model, and an attention mask which is set to 0 for the positions where sentence is padded.

Figure 9 shows the distribution of token counts for comments in our dataset.

We set the maximum sequence length of tokenizer to 256 words based on above graph, while truncating those comments which exceed this token length and padding those which fall short.

4.4.4 ***Sentence Embeddings.*** : For generating sentence embeddings we use distilbert-base-cased model from transformers library which is capable of handling case-sensitive text.

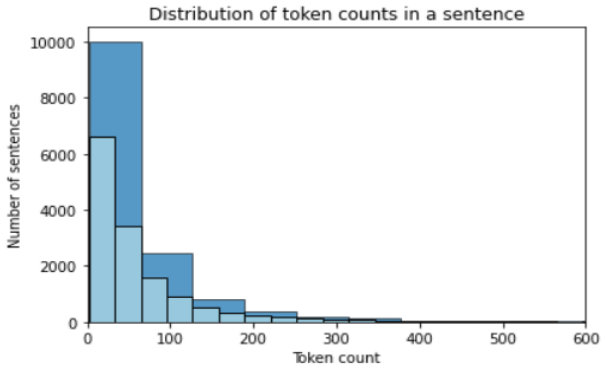

Figure 9: Distribution of token counts in each sentence

Since the dataset is small and still contains many misspelled words, trainable layers of this embedding model are set to false. This means that we are using the pre-trained model as is, but not fine-tuning it for the classification task. This is an important design decision which will change if the misspelled words are corrected without compromising the context of sentence, and if size of data increases.

4.4.5 ***Model Architecture***. : Figure 10 shows the actual architecture of our multi-label classification model, picked up from the Hugging Face repository:

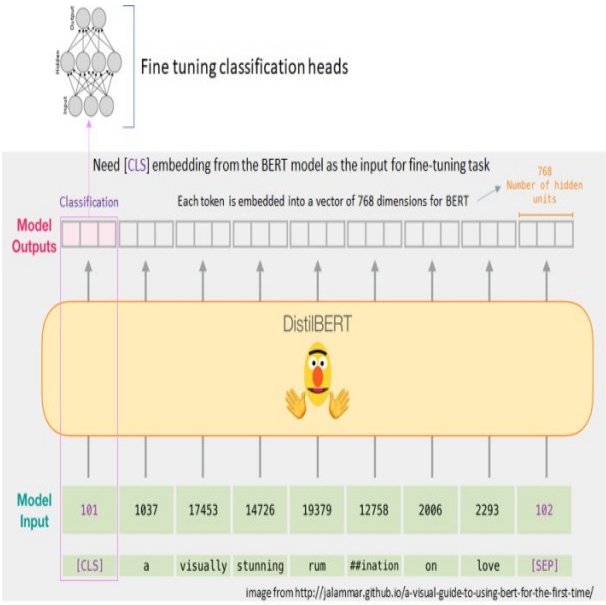

Figure 10: distilBERT based classifier architecture

- A Keras based classification model is built with following layers:
  (1) The distilBERT sentence embedding layer that takes tokenizer's results as input. Dimensionality of the encoder layers and the pooler layer of distilBERT is 768, so each sentence is encoded into a vector of dimension 768.
  (2) Dense layer with 128 nodes
  (3) Dense layer with 32 nodes
  (4) Final dense output layer with 6 nodes that uses sigmoid classifier to independently yield values between 0 and 1 for each label.
- Adam optimizer is used as a standard choice, but in Keras, AMSgrad property is set to False. AMSGrad is an extension to the Adam version of gradient descent that attempts to improve the convergence properties of the algorithm, avoiding large abrupt changes in the learning rate for each input variable. But due to a small learning rate of 5e-5 and unlikely chance of abrupt changes, we set the AMSgrad to False.
- Binary cross entropy is used to compute loss, and accuracy is used as an evaluation metric for the model.

4.5 Model Implementation and Evaluation

4.5.1 ***Train,Test and Validation Datasets***. : 20000 comments with labels are split into 3 parts - 14400 comments are used for training the model, 10% of rows or 2000 comments are used for constructing validation dataset and model's final performance is evaluated on 3600 comments of test dataset.

4.5.2 ***distilBERT-based Model Execution***. : The model is implemented for 3 epochs, considering a batch-size of 50. The input to this model is tokenizer's results and the output is a 3600x6 array of numeric arrays of for 3600 comments, and 6 label values between 0 and 1.

4.5.3 ***Evaluation Metrics***. : Inspired from the Kaggle's Toxic Comment Classification challenges, model's performance is evaluated on ROC curve, AUC score in addition to accuracy.

5 RESULTS AND DISCUSSION

The results table below in Figure 11 shows the results as obtained from both classification techniques, by setting classification threshold at 0.1 for all individual classifiers. Note that the Classifier Chain doesn't have single accuracies, but rather an overall subset accuracy which is characteristic to Classifier Chain.

Model	Accuracies across each label and over-all accuracy						
	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity Hate	Overall
OneVsRest	82%	95%	81%	98%	81%	96%	89%
Classifier Chain	-	-	-	-	-	-	57%
distilBERT (Threshold 0.1)	63%	89%	60%	98%	59%	93%	77%

Figure 11: Results for different classifiers

5.1 Results for context-independent model

- For OnevsRest Classifier, the ROC curve in Figure 12 shows that the AUC values are lower in some categories and higher in other. For categories like toxic and obscene, the algorithm has predicted output with better accuracy. The AUC values for 'toxic' label is 0.82 while for 'threat' label it is 0.50.

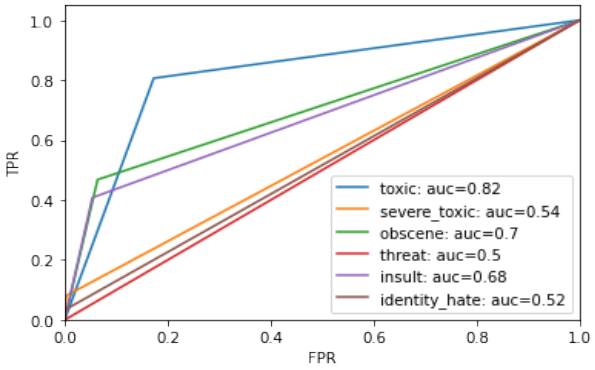

Figure 12: ROC curve for OneVsRest Classifier

- For OnevsRest Classifier, the accuracies for different categories are shown in the Figure 13 below. Note that these are individual class accuracies but not a single accuracy across six labels.

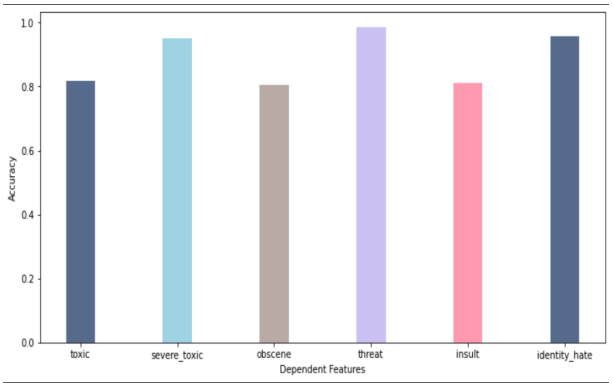

Figure 13: Accuracies for OneVsRest Classifier

- For Classifier Chain, The accuracy is significantly lower. This is because the classifier chain accuracy is computed as a subset accuracy. The subset accuracy is the strictest measure for which the set of labels predicted for a sample must exactly match the corresponding set of labels in testing data. We got subset accuracy of 57% for our dataset. It indicates the percentage of samples that have all their labels classified correctly.

5.2 Results for context-based model

- The model's loss and accuracy on training and validation datasets is shown in figures 14 and 15.

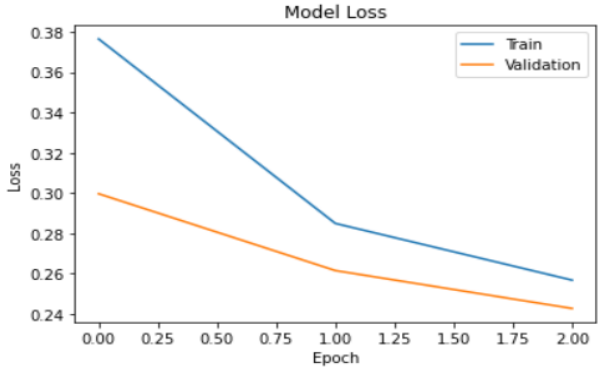

Figure 14: distilBERT based classifier architecture

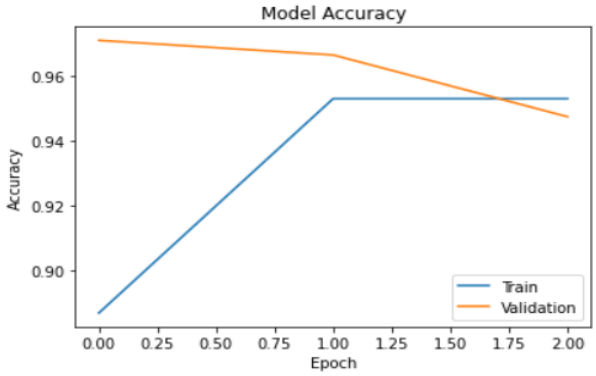

Figure 15: distilBERT based classifier architecture

- The model's ROC curve is shown in Figure 16. The area under the curve comes to 92.79

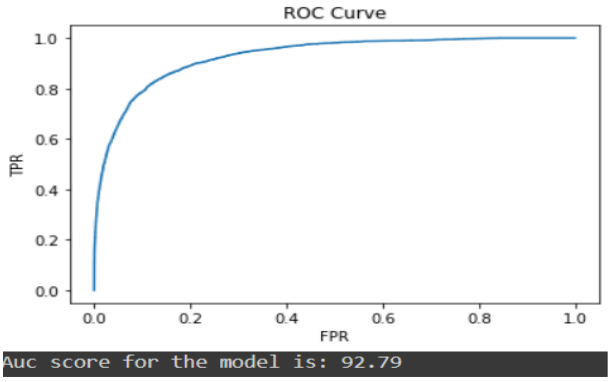

Figure 16: distilBERT based classifier architecture

- The above graph is plotted by taking a difference between raw sigmoid outputs to class labels which have values 0 or 1. Hence the model's AUC is fairly generous. Figure 17 plots model's ROC curve for threshold value of 0.1, i.e. all sigmoid outputs below 0.1 are classified as non-toxic and all those

above 0.1 are classified as toxic. This model's ROC curve looks different as we can see, and area under curve for this model is 82.95.

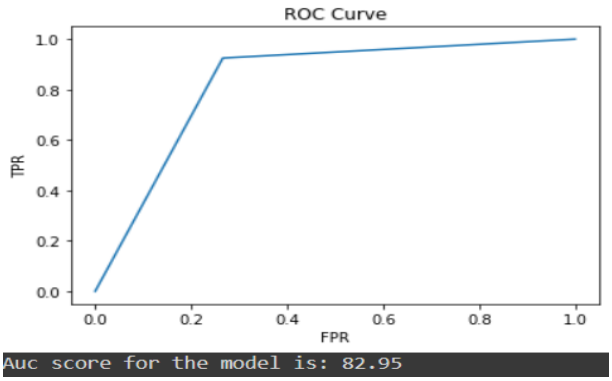

Figure 17: ROC curve for predicted labels vs actual labels at threshold of 0.1

6 FUTURE WORK

Our model faces many challenges in certain areas which degrades or limits its performance. Some of these challenges have been discussed below. And as a part of future work, these challenges can be addressed to build a more nuanced yet robust, scalable model:

6.0.1 Identify vulnerable target groups. Wikipedia's dataset comprising of gender-based, ethnic or racial slurs can be appended to the dataset to improve the model's capability in identifying abuses directed to vulnerable social groups such as minorities, people or colour and race.In our dataset we have observed the racial, ethnic and religious hate words frequently used in the comments marked as 'identity hate'. Even certain body shaming words such as 'FAT' is frequently used in these type of comments.

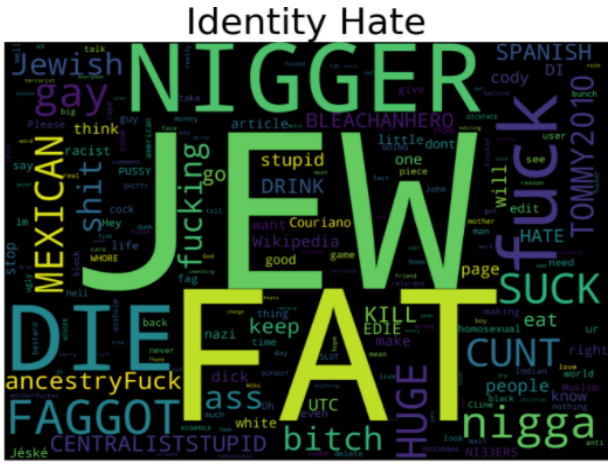

Figure 18: Identity hate word cloud

6.0.2 Noised tokens. We observed instances of abusive words with a minor modification to prevent the classifier from detecting them while retaining the readability for the user. For example, attacker can alter a single character to an asterisk (eg. "Shut up" to "S*ut up") to bypass the toxic content detection filter. A robust model capable of identifying such words can be constructed by training a denoising auto-encoder that takes in noised tokens as input and predicts the denoised version of the token. This implementation would substitute or precede spelling correction task.

6.0.3 Use of acronyms. In today's time, use of acronyms for swear words have been in common use. For example, "bs" is used for "Bull shit". The lexicon approach uses a dictionary of words containing their sentiment features. It also contains phrases (such as "bad ass") and sentiment-laden acronyms (such as "WTF", "BS!"). These words are rated for their polarity and intensity on a scale from "-4" to "+4". The average of their lexical features can be incorporated as the sentiment of the word in our training model.

7 LIMITATIONS AND IMPLEMENTATION CHALLENGES

7.0.1 Problem formulation with available data.

- **Challenge**: The biggest challenge we faced working with this size of data. With over .15 million training data, and class-imbalance on top of that, our computational resources were severely limiting - we could not even go as far building a classification model because tasks such as corpus-based as well as context-based spelling correction exhausted GPU quotas when executed on Google Colab.
- This forced us to construct a class-balanced sample of 20000 comments - potentially limiting the informativeness of our exploratory analysis, although we had been able to ensure to some extent that our sample statistics closely match population statistics with respect to sentiment scores, curse word count distributions, correlation between different labels etc.
- In other words, our exploratory analysis (performed on a sample of original data) is likely to scale well for the whole data.

7.0.2 Hypotheses.

- **Limitation**: One of our hypotheses was to analyze use of profanity with positive sentiment. Unfortunately this line of analysis hasn't provided an statistically significant and conclusive evidence, as seen in Figure 14 below.
- Figure 14 is plotted by taking sentiment scores and corresponding number of curse words in each of the 10000 non-toxic sentences. As we can see, the sentiments of non-toxic sentences range from strong positive to strong negative polarity.
- There are very few instances in our 10000 non-toxic comments which have curse words. And moreover since the sentiment scores are uniformly distributed even for non-toxic comments, this hypothesis is difficult to prove.

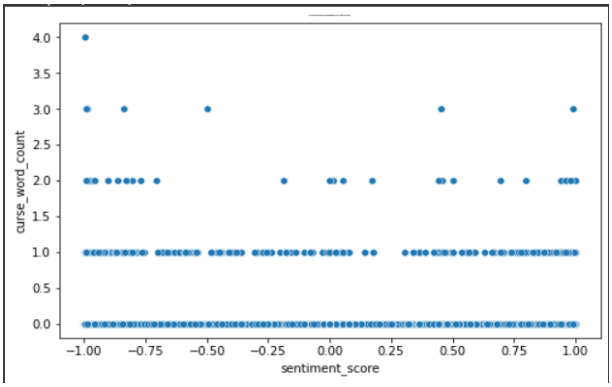

Figure 19: Count of curse-words vs sentiment score for non-toxic text

7.0.3 *Spelling Correction Task.*

- **Challenge**: As already discussed in future work, another big challenge was correction of misspelled words. We tried 4-5 different spelling correction functions, including pyspellchecker[1], context-based algorithms such as textblob'spelling correction algorithm and custom functions like Peter Norvig's Levenshtein distance based corrector[8], all of which exhausted GPU and TPU resources on Google Colab even for batch processing.
- Since the text is in informal language with noisy tokens, it has been extremely difficult task to come up with a spelling correction function that corrects the spelling while preserving the meaning.
- In our final implementation, we have not performed any spelling correction for misspelled words and noisy tokens.

7.0.4 *Model Fine-tuning.*

- **Limitation**: For faster generation of sentence embeddings, the trainable layers of pre-trained distilBERT models have been set to False. This means that our data doesn't manipulate the weights of pre-trained model in any way, and sentence embeddings were generated by leveraging original pre-trained weights.
- This design choice doesn't allow the model to leverage the patterns that occur in toxic comments, and may yield a comparatively poorer performance, but has been made for two reasons:
  (1) To speed up model training. When fine-tuning of weights is disabled, model is faster
  (2) To ensure scalability of model and generalization capability. Since our model is built on smaller dataset, we didn't want it to over-fit to the patterns found in this subset. Disabling trainability mitigates the issue of over-fitting.

7.0.5 *Applicability of findings.*

- **Limitation**: For the reasons stated above, like using a sample of data, class-imbalance, disabled fine-tuning approach, the scalability of the model remains to be explored further. It has not been taken up due to severe restrictions of computational resources.
- The model returns sigmoid outputs which take on the values between 0 and 1. Figure 13 illustrated ROC curve if the classification threshold is set to 0.1. This threshold is definitely going to change once larger data is brought into the picture.
- Hyper-parameter like token size, number of training epochs, layer properties of feed-forward neural network, all need to be revisited once larger dataset is used for training.

7.0.6 *Prerequisite - How to use our classifiers.*

- **Selection of right threshold based on task**: Both classification models discussed in this paper use a common threshold of 0.1; all labels above 0.1 are mapped to 1 and less than 0.1 are mapped to 0.
- Based on the necessity of task (e.g. having a model which has high sensitivity towards specifically, say, threats), the model's threshold can be tweaked to maximize prediction accuracies for a particular label
- The caveat here is that same threshold will not work for all labels, as we can clearly see from the pulled-down value of overall accuracy. So threshold must be fine-tuned based on prediction requirement. However, a value between 0.08 and 1.2 gave a generally higher overall performance during our experimentation.

8 CONCLUSION

Although this is a stand-alone project which uses static data-sources and focuses more on the statistical, analytical and algorithmic aspect of toxicity detection, the project can be further developed and incorporated as a feature in these applications with a capacity to analyze the user content in real-time and prevent uploading of offensive content for display to the general public, because, prevention is better than cure. The intended users of this feature would be social-media sites, government information/ discussion portals and news forums where the public interacts through chat forums that could be monitored and stored for data processing.

REFERENCES

[1] Tyler Barrus. 2021. *Python Spelling Checker.* https://github.com/barrust/pyspellchecker
[2] Ekaba Bisong. 2019. *Logistic Regression.* https://doi.org/10.1007/978-1-4842-4470-8_20
[3] Intel Corporations Fahim Mohammad. 2018. *Is preprocessing of text really worth your time for toxic comment classification.* https://arxiv.org/abs/1806.02908
[4] C.J. Hutto. 2014. *VADER-Sentiment-Analysis.* https://github.com/cjhutto/vaderSentiment
[5] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* https://arxiv.org/abs/1810.04805
[6] Geoff Holmes Eibe Frank Jesse Read, Bernhard Pfahringer. 2015. *Classifier Chains for Multi-label Classification.* https://www.cs.waikato.ac.nz/~eibe/pubs/chains.pdf
[7] Geoff Holmes Eibe Frank Jesse Read, Bernhard Pfahringer. 2019. *Classifier Chains: A Review and Perspectives.* https://doi.org/10.1613/jair.1.12376
[8] Peter Norvig. 2019. *How to Write a Spelling Corrector.* https://norvig.com/spell-correct.html
[9] Hugging Face Team. 2019. *Hugging Face: State-of-the-Art Natural Language Processing in ten lines of TensorFlow 2.0.* https://blog.tensorflow.org/2019/11/hugging-face-state-of-art-natural.html