Master of IT in Business

ISSS610 APPLIED MACHINE LEARNING

Final Project Report:
Toxic Comment Classification

Project Group 5:

Ayushi JAISWAL

Milouni DESAI

SOH Hui Shan

TEO Yaling

Table of Contents

1. Introduction 3
2. Objective 3
3. Dataset 3
3.1 Exploratory Data Analysis 4
3.2 Data Preprocessing 4
4. Feature Engineering 5
5. Evaluation Metrics 5
6. Approach 6
6.1 Initial Approach 6
6.1.1 Models 6
6.1.2 Results and Model Selection 8
6.1.3 Challenges 8
6.2 Revised Approach 8
6.2.1 Flow Chart 9
6.2.2 Models 10
6.2.3 Results and Model Selection 14
7. Insights 15
8. Conclusion and Future Work 16
References 17

1. Introduction

Online discussion is an integral aspect of social media platforms and it has become an avenue for abuse and harassment. Toxic comments are comments that are rude, disrespectful or otherwise likely to make someone leave a discussion. These include attacks on identity, insults, profanity, threats, sexually explicitness, etc. Human facilitation on online comments is labor-intensive. Platforms have struggled to effectively facilitate conversations. On the extreme end, some platforms have resorted to shutting down user comments completely. Social media platforms are under pressure from the public to combat toxic comments.

Several machine learning methods have been developed to assist online content publishers and moderators to flag toxic comments on their platforms to enhance the exchange of ideas on the internet. Perspective was created by Jigsaw and Google's Counter Abuse Technology team in a collaborative research project called Conversation-AI. Most of their models are based on Convolutional Neural Network (CNN) trained with word-vector inputs (Perspective API, n.d.). However, these models were not recommended by the developers to be used as a form of automated moderation as they acknowledged that the existing models have made too many errors.

2. Objective

The objective of our project is to build a multi-label classification model that can accurately detect different types of toxicity for each comment. Many models built by Kaggle competition participants center around word embeddings and deep learning models. We would like to observe if complex approaches of word embeddings coupled with deep learning pose any significant improvement in accuracy over simpler approaches. Hence, we have explored traditional approaches for vectorization using Bag-of-Words (BoW) or Term Frequency – Inverse Document Frequency (TF-IDF) representations, as well as newer approaches like word embeddings. As for the models, we have deployed a range of classification methods, from simpler classification methods like Naïve Bayes, logistic regression, support vector machines, to more complex methods of neural networks and CNN.

3. Dataset

The dataset, taken from a Kaggle competition (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data), contained Wikipedia Talk Page Comments with human-labelled toxicity reasons. There were 6 classes of toxicity, namely, Toxic, Severe Toxic, Obscene, Threat, Insult, Identity Hate. Each comment is associated with a set of 6 binary labels, 0 or 1, and each comment can be labelled with more than one type of toxicity.

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
1	Hi! I am back again! Last warning! Stop undoing my edits or die!	1	0	0	1	0	0
2	Gays are disgusting. It's just my opinion but gays are disgusting.	1	0	0	0	1	1
3	IM GOING TO KILL YOU ALL!! I'm serious, you are all mean nasty idiots who deserve to die and I'm going to throw every single one of you in a shredder!!!!!!!!!	1	1	0	1	1	0

Figure 1. Sample of dataset

The dataset was divided into separate train set with 159571 comments and test set with 153164 comments. Out of the 153164 comments in the test set, only 63978 comments were associated with known labels.

3.1 Exploratory Data Analysis

From our EDA, we see that almost 90% of the comments are clean, while the rest of the 10% are distributed as shown below.

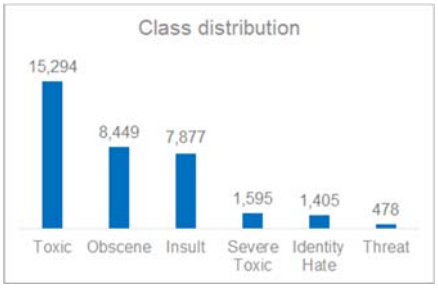

Figure 2. Distribution of each label

While the word clouds for toxic and clean comments are distinctly different, the word clouds for all toxic labels look very similar to each other which suggests that our models must be able to detect the nuances between the 6 toxic labels.

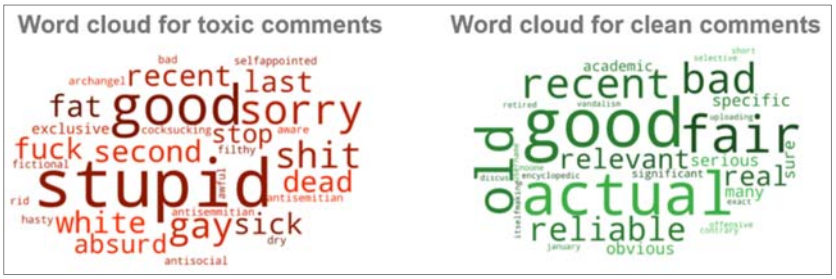

Figure 3. Word clouds of toxic and clean comments

3.2 Data Preprocessing

Comments from both train and test datasets with known labels were combined to pool together a larger dataset for model training and testing. This resulted in a dataset comprising of 223549 comments. Down sampling was performed to reduce the proportion of clean comments (Toxic label = 0) from 90% to 60%.

After sampling, the dataset comprised of 54936 comments. Similar proportions of each of the 6 labels were maintained in the dataset before and after sampling.

The following data processing techniques were applied to all comments:

- Removed URLs, HTML tags, stop words, numerals.
- Removed special characters except for exclamation marks "!"
- Retained the case structure of alphabets (lower-case, upper-case, etc.)
- Performed lemmatization for all word tokens

The rationale for the deliberate retention of exclamation marks and case structure is that toxic comments tend to be associated with more exclamation marks or upper-case letters.

4. Feature Engineering

We have created word-level features and comment-level features. For word-level features, we have performed pre-processing steps as discussed above, followed by vectorization methods such as Bag of Words (BoW), Term Frequency Inverse Document Frequency (TF-IDF) and binary representation of each word. We have also used pre-trained GloVe word embeddings, namely *glove.6B.zip* and *glove.840B.300d.zip*, containing 400,000 and 2.2 million vocabulary respectively.

As for comment-level features, we have a few hypotheses about the statistics of the comment. First, we hypothesized that the total length of toxic comment may be shorter than clean comment. Second, toxic comment may have lesser number of unique words. The boxplots below have shown that these hypotheses are true.

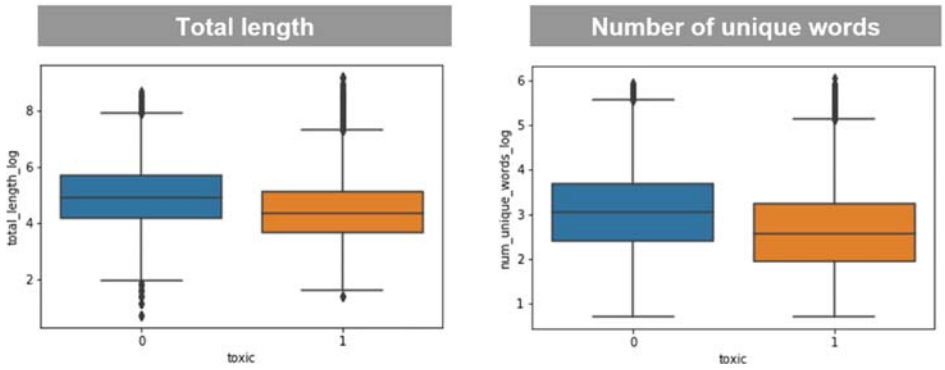

Figure 4. Boxplots showing Total Length and Number of Unique Words of each class of the Toxic label

Based on this idea, we created 9 comment-level features.

Comment-level features	Description
Word count	Number of words
Character count	Number of characters, including punctuation marks
Word density (Word count / Character count)	To find out if long words or short words are used
Total length	Sum of word count and character count
Capitals	Number of upper-case characters
Proportion of capitals	Hypothesis: Toxic comment has more capitals
Number of exclamation marks	Hypothesis: Toxic comment has more '!'
Number of unique words	To find out if the same word repeats many times
Proportion of unique words	Hypothesis: Toxic comment has lower pp. Of unique words

Table 1. Comment-level features

5. Evaluation Metrics

Evaluation of multilabel classification is different than that of a binary classification. For binary classification, the model's prediction is either correct or wrong. Hence, simple metrics such as Precision, Recall, F1-score, Accuracy, etc. could be used. However, multi-label classification requires averaging of

these metrics. This is because, even if the model predicts two or three labels correctly out of all six labels, it's still better than predicting no labels correctly at all.

There are stricter evaluation metrics for multi-label classification, such as Exact Match ratio (EMR), which is simply the percentage of samples that have all their labels classified correctly. For our case, we don't require such a strict evaluation metric. As long as our model predicts a few of the labels correctly, it will still serve its purpose to help identify toxic comments on Wikipedia. Hence, we can go with a slightly lenient metric such as micro or macro averaging.

Macro-Averaging is straight forward. We simply take the average of precision and recall of all the labels and F1-score is the harmonic mean of precision and recall.

$$Macro\ Precision = \frac{\sum Precision}{No.of\ labels}$$

$$Macro\ Recall = \frac{\sum Recall}{No.of\ labels}$$

$$Macro\ F1\ score = \frac{2 * Macro\ Precision * Macro\ Recall}{Macro\ Precision + Macro\ Recall}$$

Micro-Averaging, on the other hand, sums up the True Positives, True Negatives and False Negatives for all labels and then takes average.

$$Micro\ Precision = \frac{\sum TP}{\sum TP + FP}$$

$$Micro\ Recall = \frac{\sum TP}{\sum TP + FN}$$

$$Micro\ F1\ score = \frac{2 * Micro\ Precision * Micro\ Recall}{Micro\ Precision + Micro\ Recall}$$

Micro-Averaging is generally used when the target is imbalanced, which is true for our case. Hence, we will use Micro Averages to evaluate our models.

6. Approach

6.1 Initial Approach

The initial approach is to classify the comments into 6 labels – toxic, obscene, insult, severe toxic, identity hate and threat. We have built 3 different models – Bag of Words + Naïve Bayes, Glove Embeddings + Logistic Regression and Glove Embeddings + Neural Network.

6.1.1 Models

BoW + Naïve Bayes

The comments were tokenized using NLTK's word tokenizer and the top N most frequently occurring words are extracted from each of the 6 labels and pooled together to form a word feature set. The number of words (N) extracted from each label are listed in the following table.

Label	Class	N
Toxic	1	1000
Severe Toxic	1	300
Obscene	1	300
Threat	1	300
Insult	1	300
Identity Hate	1	300
Total		**2500**

Table 2. Number of words extracted from each label

The word feature set was then applied to every comment to identify if the comment expresses any of the words present in the feature set. These features were then vectorized and passed to a multinomial Naïve Bayes classifier. The model gave good accuracy of 96.65% but low micro-precision of 53.51% and low micro-recall of 74.88%.

Glove Embeddings + Logistic Regression

We used pre-trained Glove Embeddings (glove.840B.300d.txt) as our feature engineering to model the words into a global vector space. Next, we used the 2 million word vectors to create normalized vectors for the sentences and passed them to a Logistic Regression model. The model gave good accuracy of 96.79% but very low micro-recall and micro-F1-score of 14.28% and 24.83% respectively.

Glove Embeddings + Neural Network

Similar to the model above, we used pre-trained Glove Embeddings (glove.6B.zip) as our feature engineering. Next, this was fed to a neural network as shown below with one embedding layer, followed by LSTM layer and a Dense layer. Since we are training a multi-label classifier, our network had 6 output nodes predicting one for each label. We tried different vocabulary sizes, different epochs and batch size to ensure our model does not overfit. The model gave the best results for epochs = 5, vocabulary = 100 and batch size = 128 leading to overall accuracy of 96%.

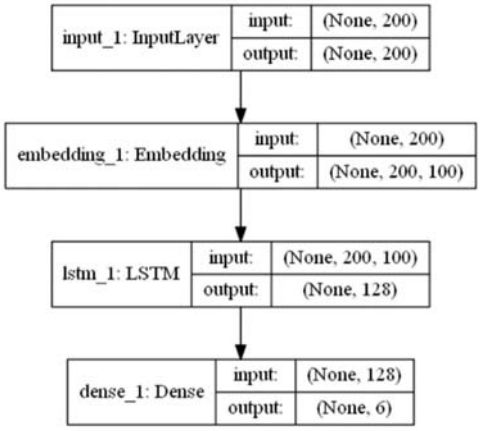

Figure 5. Architecture of Glove Embeddings with Neural Network

6.1.2 Results and Model Selection

The results of the three models are summarized in Table 2 and each of them has a high average accuracy of 96%. However, the micro-recall and micro-F1-score is very low for embeddings with logistic regression and neural network model.

Model	Multi-Label			
	Average Accuracy	Micro-Precision	Micro-Recall	Micro-F1-Score
BoW + Naïve Bayes	96.65%	53.51%	74.88%	62.42%
Embeddings + Logistic Regression	96.79%	95.01%	14.28%	24.83%
Embeddings + NN	96.30%	61.54%	1.06%	2.08%

Table 3. Results of each models based on initial approach

6.1.3 Challenges

As observed from the confusion matrix, false negatives are very high which suggest that the models are predicting the toxic comments as clean. From the business perspective, if we want to use this model to flag out and remove toxic comments from a discussion forum, the model will not be able to do so. Therefore, we know that the high accuracy of our model is driven by the highly imbalance data where 90% of our comments are clean.

Embeddings + Logistic Regression

	Predicted: Clean	Predicted: Toxic
Actual: Clean	TN 9203	FP 8
Actual: Toxic	FN 777	TP 204

Embeddings + Neural Network

	Predicted: Clean	Predicted: Toxic
Actual: Clean	TN 9209	FP 2
Actual: Toxic	FN 971	TP 10

Figure 6. Confusion Matrix showing high false negatives

6.2 Revised Approach

Down-Sampling

To overcome the issue of imbalanced data, we will down-sample the clean comments so that the ratio of clean to toxic comments is about 60:40. Below figure shows the distribution of labels before and after sampling. Note that this is a multi-label classification, hence the sum of percentages does not add up to 100.

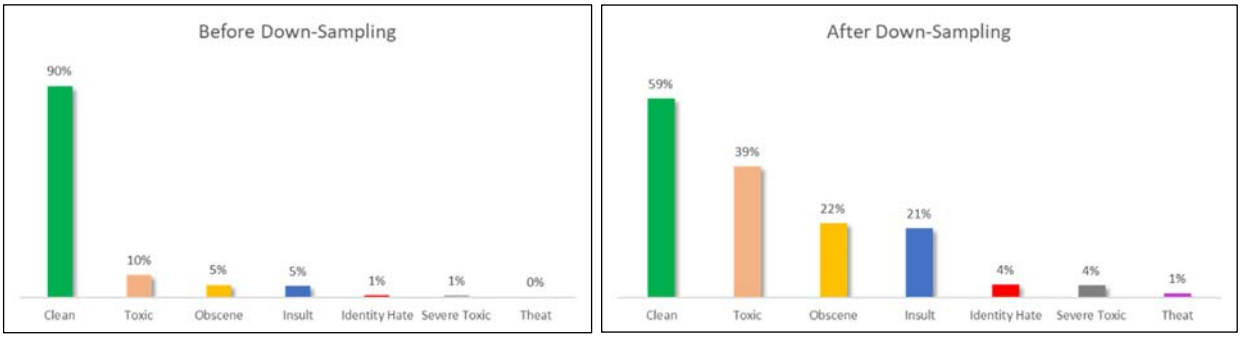

Figure 7. Distribution of each label before and after down-sampling

Word Embeddings

Next, in our first approach, we used word embeddings. One major challenge in using word embeddings is that it is not easy to interpret the models. For example, we would like to see which words drive which labels. For such cases, using word embeddings might not be a good idea. Thus, we will avoid using word embeddings in our revised approach.

Conditional Probabilities

Next, we look at the conditional probabilities for all the labels to understand how related the occurrences of the labels are. As shown in the Probability Matrix below, given that a comment is labelled as Severe Toxic, the probability that is it labelled as Toxic is 100%. Likewise, given that a comment is labelled as Obscene, the probability that is labelled as Toxic is 95%. Therefore, we observed that the other 5 labels are highly dependent on Toxic as the probabilities are more than 94%. This led us to use a 2-level classification approach.

Conditional Probabilities		Given					
		Toxic	Severe Toxic	Obscene	Insult	Identity Hate	Threat
Probability Of	Toxic	100%	100%	95%	95%	94%	95%
	Severe Toxic	9%	100%	16%	15%	21%	23%
	Obscene	54%	96%	100%	79%	74%	64%
	Insult	50%	87%	73%	100%	83%	65%
	Identity Hate	9%	23%	13%	16%	100%	21%
	Threat	3%	8%	4%	4%	7%	100%

Table 4. Probability matrix of 6 labels

6.2.1 Flow Chart

As illustrated in the flow chart below, after down-sampling and feature engineering, we have a binary classifier (1st level) to predict just one label – Toxic. Once this is done, we passed only the comments which were predicted as toxic to the multi-label classifier (2nd level) for predicting the other 5 labels. As for the

evaluation metrics for multi-label classifier, we used micro since there is still imbalance for some of the labels. We have built 10 models using the revised approach and they are discussed in the section below.

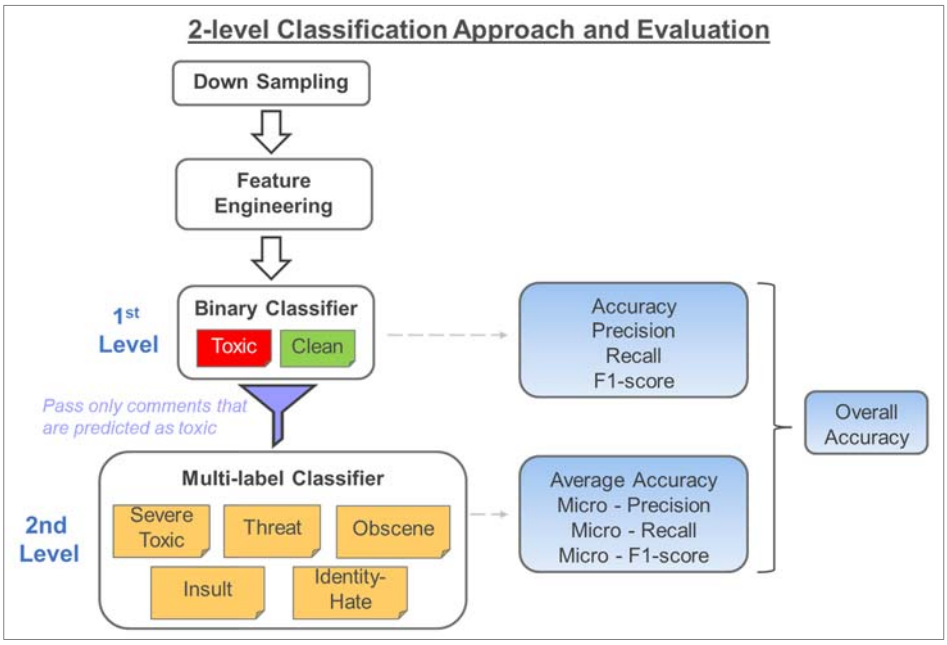

Figure 8. Flow Chart of 2-level classification approach and evaluation

6.2.2 Models

BoW + Naïve Bayes

1st level: The comments were tokenized using NLTK's word tokenizer and the top 1000 most frequently occurring words were extracted from toxic label, class = 1, to form a word feature set. The word feature set was then applied to every comment to identify if the comment expresses any of the words present in the feature set. These features were then vectorized and passed to the multinomial Naïve Bayes classifier.

2nd level: The comments were tokenized using NLTK's word tokenizer and the top N most frequently occurring words are extracted from each of the 6 labels and pooled together to form a word feature set. The number of words (N) extracted from each label are listed in the following table.

Level	Label	Class	N
1	Toxic	1	1000
2	Severe Toxic	1	600
	Obscene	1	1000
	Threat	1	600
	Insult	1	1000
	Identity Hate	1	600
	Total		**3800**

Table 5. Number of words extracted from each label

Similar to the 1st level approach, the word feature set was then applied to every comment to identify if the comment expresses any of the words present in the feature set. These features were then vectorized and passed to the multinomial Naïve Bayes classifier.

Binary / Count / TF-IDF + Logistic Regression

Like in the initial approach, we will try Logistic Regression in our revised approach as well.

However, this time we will try Logistic Regression with various vectorizers namely, TF-IDF Vectorizer, Binary Vectorizer and Count Vectorizer. TF-IDF vectorizer will fill the document-term matrix with TF-IDF scores for each of the words. Binary vectorizer will fill the matrix only based on whether the word appears in the document (comment) or not. Count vectorizer will fill the matrix with the number of times the word appears in the document (comment).

Furthermore, we will also observe the model performance for different values of C (inverse of regularization strength).

Binary / Count / TF-IDF + SVM

Support Vector Machines (SVM) have known to perform well in text classifications. Therefore, in addition to the models in the initial approach, we will try SVM in this revised approach.

Like Logistic Regression, we will try the same vectorizers and various values of C (penalty parameter).

TF-IDF + Neural Network

With the revised approach, we used TF-IDF Vectorizer, which was fed to a Binary classifier Neural Network. The network architecture is shown below. The output of this model predictions of toxic comments is fed to the next neural network which performs the multi-label classification.

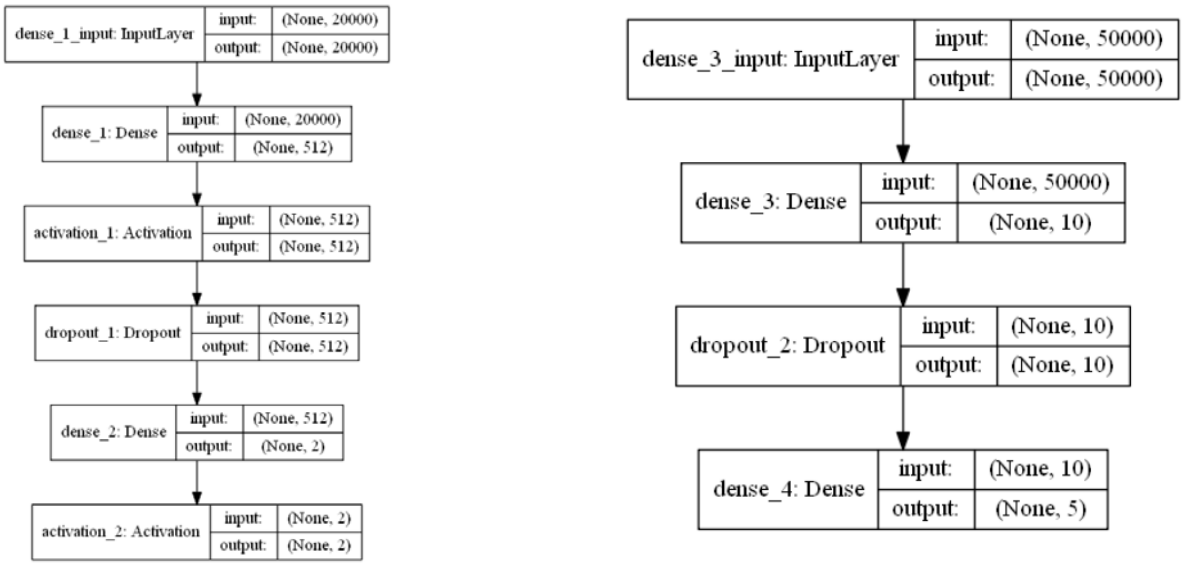

Figure 9. Overall architecture of NN. Left: Binary classifier; Right: Multi-label classifier

TF-IDF + CNN

For both 1st and 2nd level approaches, the comments were tokenized using Tensorflow's Tokenizer with the maximum number of words capped at 10000. Exclamation marks were omitted from the special character filter. The resulting tokens were transformed into a TF-IDF matrix. The resulting training and testing matrices were then reshaped to 3D to input into the following CNN architecture. The architecture for the 1st level had an output of 1 dimension whereas that of the 2nd level had an output of 5 dimensions.

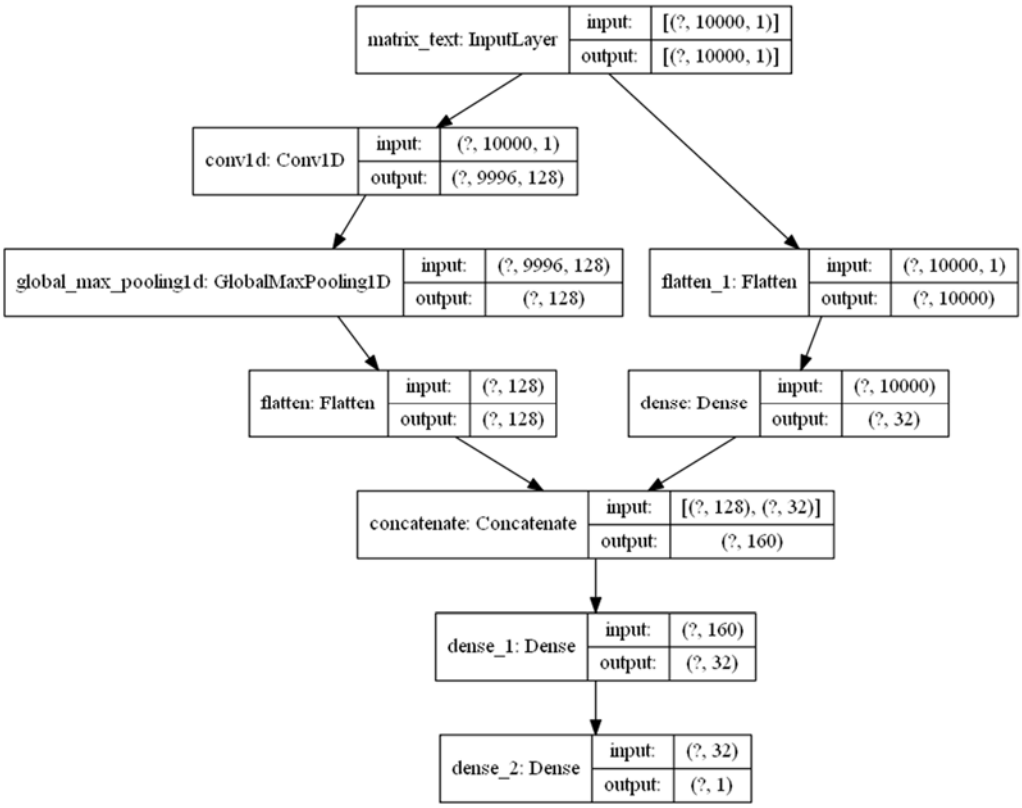

Figure 10. Overall architecture of TF-IDF with CNN

Glove Embeddings + CNN

We used Glove Embeddings along with a CNN architecture to see if the model performance would give any drastic improvement to our evaluation metrics. Although embeddings make the model interpretability difficult, if we achieve higher improvement, we can still consider this model.

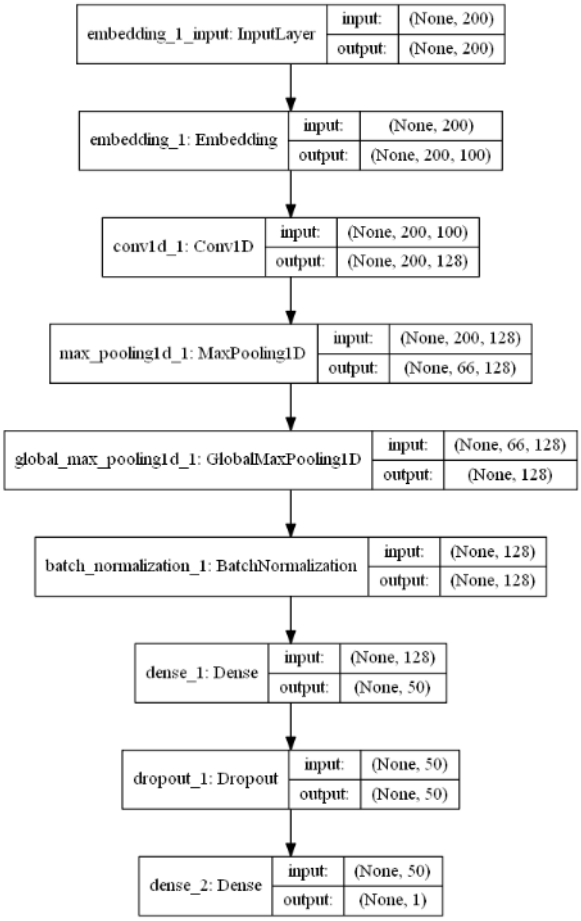

Figure 11. Overall architecture of Embeddings with CNN

The architecture of the model is shown above. The training and validation accuracy graphs have been plotted to check for overfitting. We performed hyperparameter tuning of the network by adding and excluding different layers and different epoch sizes to obtain the best model.

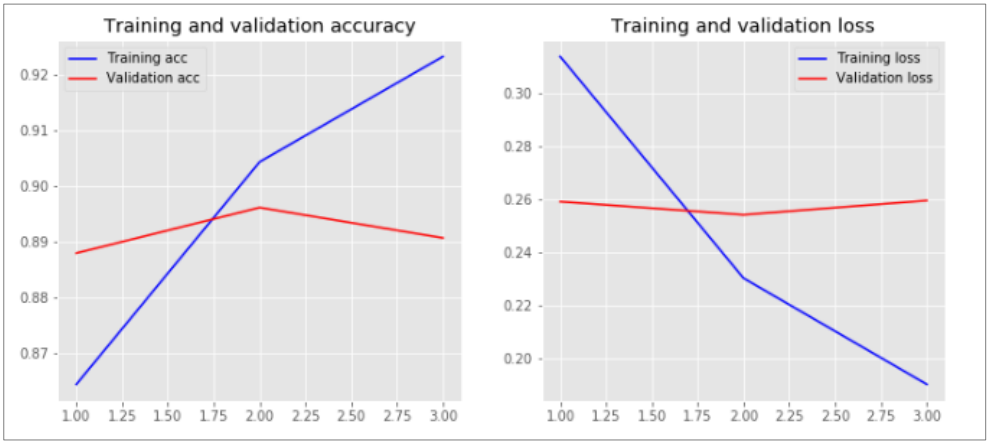

Figure 12. Training, validation accuracy and validation loss graphs

6.2.3 Results and Model Selection

For Logistic Regression and SVM, we tried various values of the parameter C. Following figure shows the best C values out of the tested values.

Model	Level 1: Binary Label	Level 2: Multi-Label
TF-IDF + SVM	1.0	2.0
Binary + SVM	0.1	0.5
Count + SVM	0.1	0.5
TF-IDF + Logistic Regression	1.0	2.0
Binary + Logistic Regression	0.2	0.5
Count + Logistic Regression	2.0	0.2

Table 6. Best values of parameter C

Following table shows the results of all the models that were tested for revised approach.

Model	Level 1: Binary Label				Level 2: Multi-Label			
	Accuracy	Precision	Recall	F1-Score	Average Accuracy	Micro-Precision	Micro-Recall	Micro-F1-Score
BoW + Naïve Bayes	87.53%	87.76%	87.53%	87.59%	82.76%	63.15%	72.14%	67.35%
TF-IDF + SVM	88.74%	88.70%	88.74%	88.70%	86.97%	75.91%	69.01%	72.30%
Binary + SVM	88.50%	88.46%	88.50%	88.43%	86.87%	75.38%	69.41%	72.27%
Count + SVM	88.36%	88.32%	88.26%	88.29%	85.05%	72.36%	63.62%	67.71%
TF-IDF + Logistic Regression	88.14%	88.12%	88.14%	88.03%	87.21%	78.41%	66.38%	71.89%
Binary + Logistic Regression	88.28%	88.25%	88.28%	88.20%	87.58%	78.36%	68.56%	73.13%
Count + Logistic Regression	88.09%	88.05%	88.09%	88.00%	85.77%	76.38%	61.18%	67.94%
TF-IDF + NN	88.86%	88.81%	88.86%	88.81%	85.38%	74.53%	61.78%	67.56%
TF-IDF + CNN	88.06%	88.01%	88.06%	88.02%	86.21%	73.99%	67.95%	70.84%
Embeddings + CNN	89.20%	89.20%	89.10%	89.10%	86.90%	72.70%	74.70%	73.40%

Table 7. Results of each model for level 1 and level 2 approaches

It was observed that Embeddings with Neural Network performed best for almost all metrics and both levels.

However, the next best models, TF-IDF + SVM for level 1 and Binary + Logistic Regression for level 2 came very close to Neural Network. Hence, we select them as our final models. We do so because the model scores are not very far off from the best model (Neural Network), and they are far less complex than Neural Network. SVM and Logistic Regression have higher interpretability than Neural Network.

Predicting on Test Dataset using the selected models for both levels, following are the overall metrics.

Level	Model	Average Accuracy	Micro-Precision	Micro-Recall	Micro-F1-Score
1	TF-IDF + SVM	95.40%	85.72%	81.48%	83.55%
2	Binary + Logistic Regression				

Table 8. Results of the best model

7. Insights

We will be using the feature importance charts to understand the insights from our models and what makes them separate the labels based on the word importance for each label.

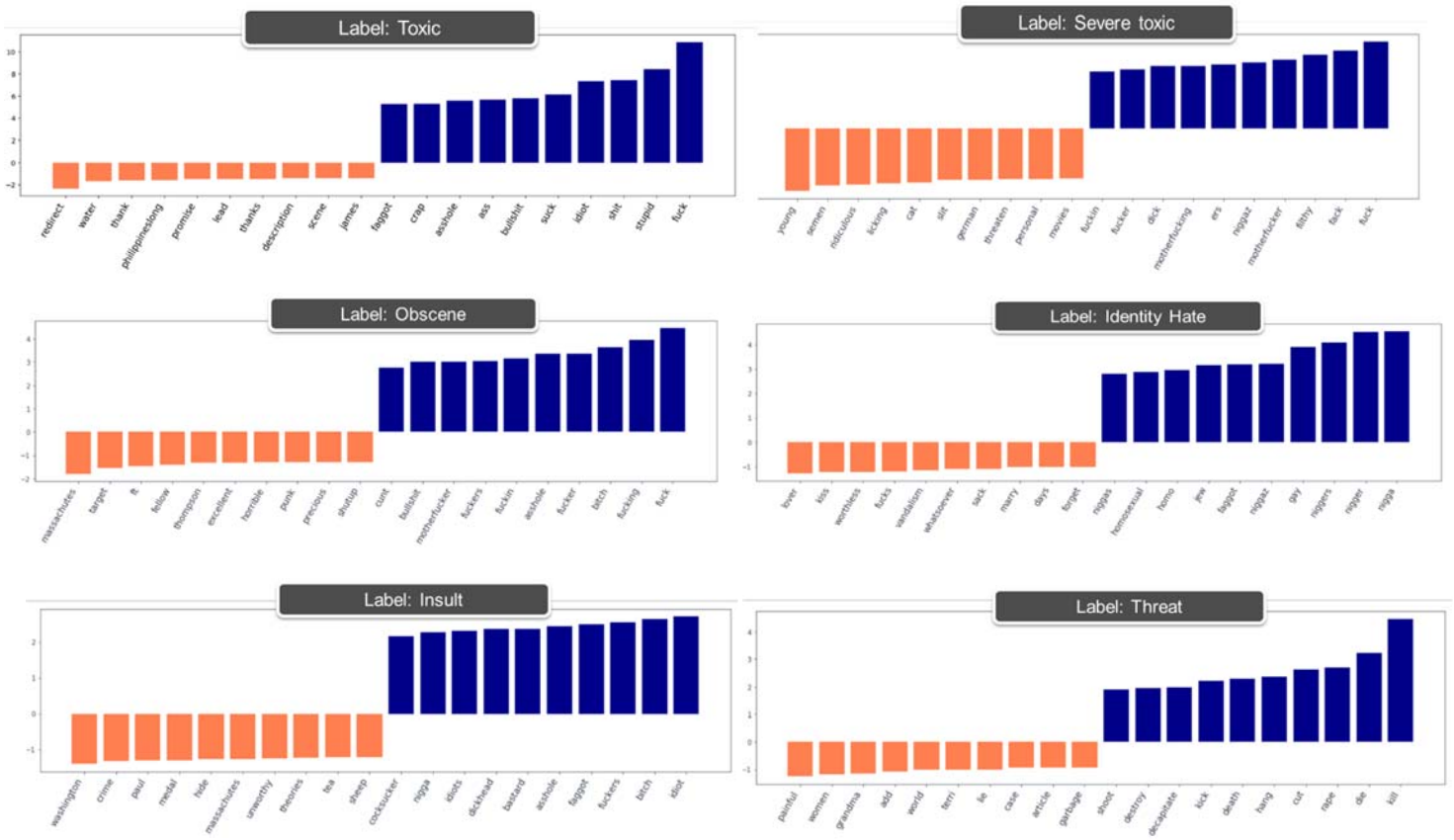

Figure 13. Word importance for each of the 6 labels

The word importance for each label is shown in the figure above. The blue color represents the positive related words with their respective labels and the negative color represents the negatively related words. Comparing toxic and severe toxic we can see that the words in severe toxic are harsher like f*cker, m*therf*cker and so on. However toxic has words like stupid, idiot, crap, which are slightly less harsh. Similarly, obscene label shares words from toxic and severe toxic labels. The Identity hate label can be seen picking words like gay, nigga, homos**ual, jew etc., which clearly are related to identities of people. The label insult has words like b*tch, idiot, b*stard which is commonly used on platforms to insult others. Threat label has words like kill, die, destroy, death and so on which occurs primarily in the context of threatening people.

Some of the examples from our model results are shown below:

Comment	True Labels	Predicted Labels
Hello, you disgusting transsexual. You are a grotesque blight upon the earth. Please kill yourself.	Toxic, Obscene, Threat, Identity Hate, Insult	Toxic, Obscene, Threat, Identity Hate, Insult
Wtf, Why You deleting Maashel? Bad things will happen to you now.	Toxic, Threat	Toxic
am going to shoot you in the head and laugh as your brains splatter onto the ground.	Toxic, Severe Toxic, Threat	Toxic, Threat

Table 9. Examples of model results

Based on the examples and word importance we can see that in first example, our model is classifying all the labels correctly. In 2nd and 3rd example, the model is missing out on picking threat and severe toxic label respectively. We would still consider this as a good model since our approach believes in primarily labelling toxic comments so that they can be automatically removed from social platforms. Classifying next level is a secondary goal and this is also difficult to achieve since the training data for the labels is not sufficient to train a good classifier.

8. Conclusion and Future Work

Using the different approaches, we can see that our model has performed well with a simpler model achieving a good overall accuracy. In choosing a simple model, we want to keep the interpretability of the model intact as the complex models did not achieve a tremendous improvement in accuracy. The modelling is an iterative process and can be improved with removing stop words using a manually created list. The list can be updated on a regular basis to remove unwanted words like names of people, places, commonly used words with no importance in our context. Similarly, different pre-processing techniques like POS tagging, different pre-trained embeddings can be considered as future scope for our project to improve the accuracy.

References

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

Do Pretrained Embeddings Give You The Extra Edge? Retrieved from: https://www.kaggle.com/sbongo/do-pretrained-embeddings-give-you-the-extra-edge

Georgakopoulos, S., Tasoulis, S., Vrahatis, A., & Plagianakos, V. (2018). Convolutional Neural Networks for Toxic Comment Classification.

He, S. (n.d.) Classifying Toxic Comment. Retrieved from: https://sijunhe.github.io/blog/2018/05/01/kaggle-toxic-comment/

Malik, U. (2019). Python for NLP: Multi-label Text Classification with Keras. Retrieved from: https://stackabuse.com/python-for-nlp-multi-label-text-classification-with-keras/

Nooney, K. (2018). Deep dive into multi-label classification..! (With detailed Case Study). Retrieved from: https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

Perspective API (n.d.) Retrieved from: https://github.com/conversationai/perspectiveapi/blob/master/2-api/models.md).