Offensive Twitter Language Detection

Dino Tan 1003368, Suhas Sahu 1003370,
Ong Li-Chang 1003328, Noorbakht Khan 1003827

December 13, 2020

Intweetion

Contents

1 Abstract

This project is part of 50.038 Computational Data Science Module (SUTD). Our team aims to explore the data science segment of Multi-Class Text Classification, specifically in the sphere of offensive language detection. Using labelled twitter data, we explored various text embedding methods, followed with the implementation of statistical and deep learning models for hierarchical class prediction.

Link to Dataset: https://sites.google.com/site/offensevalsharedtask/olid
Link to code: https://github.com/Noorbakht/Offensive-Tweet-Detection
Github Link for UI: https://github.com/Noorbakht/CDS-Offensive-Tweet-Detection

2 Introduction

We have chosen to use Offensive Language Identification Dataset (OLID) that contains a collection of 14,200 annotated English tweets using an annotation model that encompasses the following three subtasks:

- A:Offensive Language Detection
- B: Categorization of Offensive Language.
- Offensive Language Target Identification

Using OLID dataset, we will conduct data cleaning and visualisations, followed by applying various machine learning methods in a hierarchical approach from subtasks A - C.

3 Project Motivation

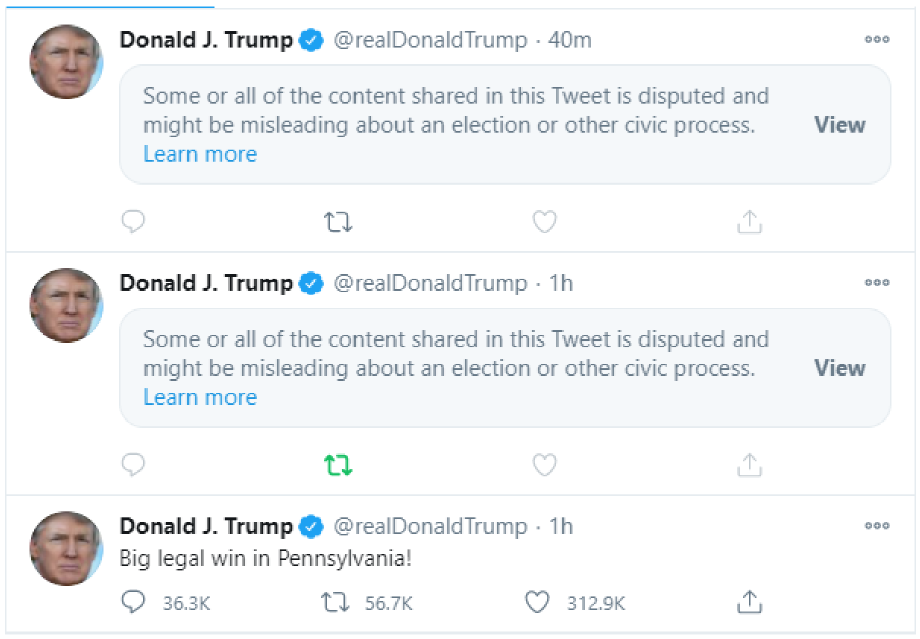
This report was written in 2020, in the middle of a global pandemic, life as we knew it changed rapidly. COVID-19 forced billions to shutter themselves in and locked down entire nations.

Now, one might have thought that in such a crucial moment, humans as a collective would have found it in themselves to rally against the virus. Instead, we now live in a world that is more divided than ever before and technology has become the weapon of choice to disrupt the peace. One of the main tools used is Social Media.

Nearly a third of the world's population is on Facebook alone. Social Media is definitely capable of influencing a large proportion of the world. As more people have moved online, 'individuals inclined toward racism, misogyny, or homophobia have found niches that can reinforce their views and goad them to violence.'

Thus, it is extremely important to discover ways to monitor and discourage offensive language from being used on social media platforms. Offensive language is the term that is applied to hurtful, derogatory or obscene comments made by one person to another person. The goal of our project is to inform Twitter users if a tweet they are about to put out is offensive. We hope that in doing so, we can help users build better habits and become more conscious of their output.

We chose Twitter because it is where a disproportionate amount of abuse occurs. In fact, a study of 134,000 abusive social media mentions showed that 88% of them occur on Twitter. Currently Twitter censors tweets that its deems offensive but it does not have a widespread system that warns users that their tweet is offensive before sending. This is very important because it shifts the responsibility to the tweeter and also many individuals find it challenging to discern what is offensive.



Donald J. Trump @realDonaldTrump · 40m

Some or all of the content shared in this Tweet is disputed and might be misleading about an election or other civic process. Learn more

View

Donald J. Trump @realDonaldTrump · 1h

Some or all of the content shared in this Tweet is disputed and might be misleading about an election or other civic process. Learn more

View

Donald J. Trump @realDonaldTrump · 1h

Big legal win in Pennsylvania!

36.3K 56.7K 312.9K

Figure 1: Trump Tweet

Thus, we want to construct an objective system that can help people spot the error in their ways, we might be able to contribute to reforming these individuals.

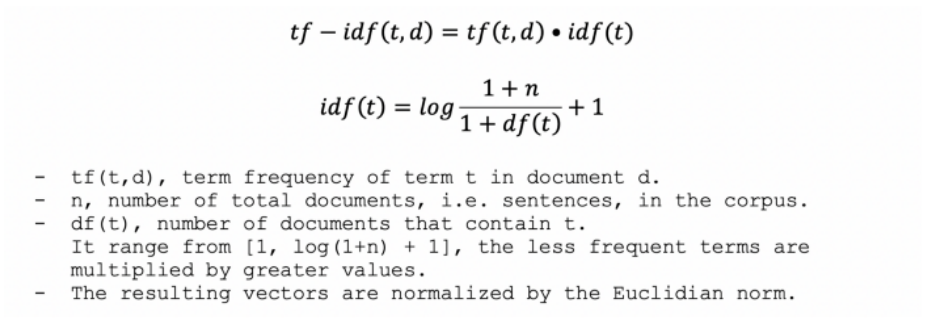
4 Text Representation

The problem that we are trying to address is in the subfield of Natural Language Processing (NLP). In particular, we are dealing with multiclass hierarchical text classification.

For text representation, we considered 3 methods: TF-IDF, Word2Vec and GLOVE. Below we will provide a short description, highlighting the pros and cons of each representation. We will also provide our initial hypothesis on the best method.

4.1 TF-IDF

Term-Frequency, Inverse Document-Frequency or TF-IDF is usually the starting point for any Text Representation problem. Count vectors in comparison are not very capable of representing sentence context because of their lack of token ordering. TF-IDF solves this by re-weighting the count features according to how many different sentences each token has appeared in. It punishes the feature count of a term if it's contained in more sentences. Instead, it assumes 'an inverse relationship between a term's relevance to the context and its number of appearances in different documents.'



$$tf - idf(t,d) = tf(t,d) \bullet idf(t)$$

$$idf(t) = log\frac{1+n}{1+df(t)} + 1$$

```
-  tf(t,d), term frequency of term t in document d.
-  n, number of total documents, i.e. sentences, in the corpus.
-  df(t), number of documents that contain t.
   It range from [1, log(1+n) + 1], the less frequent terms are
   multiplied by greater values.
-  The resulting vectors are normalized by the Euclidian norm.
```

Figure 2: TF-IDF Formula

In summary, TF-IDF is a scoring scheme for words, it acts as a measure as to how important a word is to document. However, it does have its disadvantages. Because it is 'based on the bag-of-words model, it is not capable of capturing position in text, semantics or co-occurrences in different documents', this makes it 'useful only as a lexical level feature'.

4.2 Word2Vec

Word2Vec is an unsupervised model for generating vectors. They are 'shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.' Pairs of context/target words are extracted from a large corpus for training, with the target word being a randomly chosen word and the context words being those located within a given window around the target word.

The main disadvantage of using Word2Vec is that it has difficulty handling unknown or OOV words. In these scenarios, the model will be forced to assign random vectors to said words, which is not ideal. If Word2Vec is used to interpret data from Twitter, which is often very noisy and sparse, it is possible that we might see a dip in performance. Additionally, there is no shared representation at the sub-word levels. Each word in the corpus is represented as an independent vector despite many words being morphologically similar.

4.3 GloVe

Similar to Word2Vec, 'GloVe is an unsupervised learning algorithm for obtaining vector representations for words.' However, GloVe is trained differently. 'Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.' It builds word embeddings in a way that a combination of word vectors relates directly to the probability of these words' co-occurrence in the corpus'.

In essence, GloVe embeddings are effectively a summary of the training corpus with low dimensionality that reflects co-occurrences.' The pros of using GloVe would be the shorter training time and the possibility of performing better in semantic relatedness tasks as compared to word2vec. We will be using Stanford Glove 200d pre-trained word embeddings.

4.4 Conclusion

Based on the considerations above, we hypothesised that amongst the three text representations highlighted, GloVe embeddings should offer the best performance to efficiency ratio. Tweets have the tendency to contain emotion, hidden and polarizing content. Hence, using GloVe in incorporating global statistics (word co-occurrence) to obtain word vectors would be the most applicable. However, we shall deterministically test which method is the most effective later in the report.

5 Data Collection and Visualisation

5.1 Dataset Selection

We retrieved our dataset from OffensEval, a dataset generated for a competition to identify and categorize offensive language in social media. There were 2 datasets present for us to choose from, Offensive Language Identification Dataset (OLID) and Semi-Supervised Offensive Language Identification Dataset (SOLID). As mentioned, OLID comprises 14,200 English Tweets while SOLID has over 9,000,000 English tweets. Both the datasets followed an annotation model that encompassed the three levels below.

- A:Offensive Language Detection
- B: Categorization of Offensive Language.
- C: Offensive Language Target Identification

We eventually decided on using the OLID dataset given the constraints of our computational resources.

5.2 Dataset Exploration

The dataset is split into 2 main categories, training and testing. There are 1 training file and 6 files for testing. The testing files were split according to their subtasks, A, B and C. For each of these testing files, there is a corresponding file for labels to allow us to test the accuracy of our models.

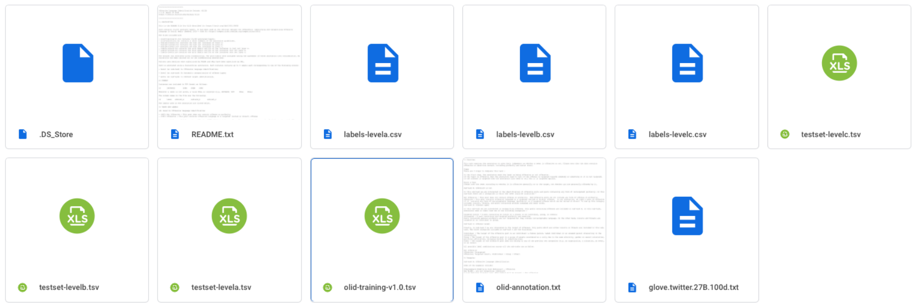
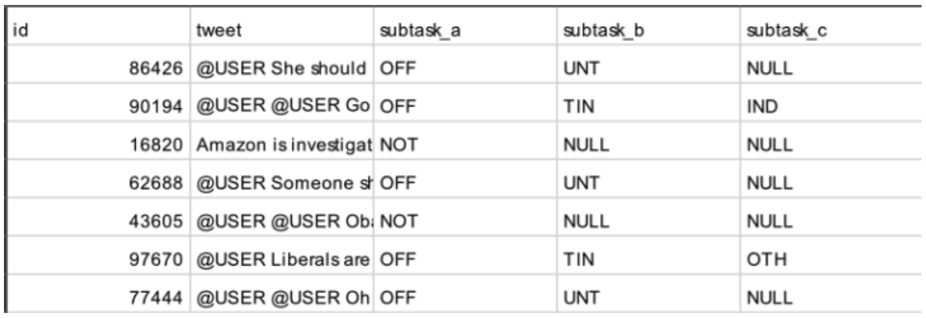


Figure 3: Distribution of Data

The OLID training data is split into 4 columns: id, tweet, subtask_a, subtask_b and subtask_c.



id	tweet	subtask_a	subtask_b	subtask_c
86426	@USER She should	OFF	UNT	NULL
90194	@USER @USER Go	OFF	TIN	IND
16820	Amazon is investigat	NOT	NULL	NULL
62688	@USER Someone sł	OFF	UNT	NULL
43605	@USER @USER Ob	NOT	NULL	NULL
97670	@USER Liberals are	OFF	TIN	OTH
77444	@USER @USER Oh	OFF	UNT	NULL

Figure 4: OLID training data sample

The training data follows the hierarchical class structure. Tweets classified as Not Offensive (OFF) will be labelled 'NULL' under subtask_b subtask_c. Tweets classified as Untargeted (UNT) will be labelled 'NULL' under subtask_c.

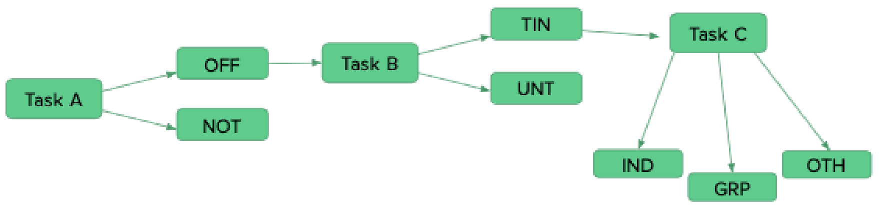


Figure 5: OLID Hierarchical Structure

Given the way the files had been segmented, we decided to take the approach of handling each of the levels individually. This meant training separate models for each of the levels, A, B and C.. To exemplify this, if a tweet was marked as not offensive, it was removed from the training set for level B and C as they would have 'NULL' values. Doing so allowed us to avoid an excessive number of tweets being classified as 'NULL' for levels B and C.

5.3 Data Visualisation

This data exploration phase was essential in influencing the subsequent steps of our project.

The first step was to visualise the distribution of the dataset. Figure 6 showcases the training data for the various subtasks. As mentioned in the section above each layer acts as a filter. Only a tweet labelled offensive(OFF) would pass through to subtask_b. If that tweet is then identified as targeted (TIN), it would then be passed to subtask_c for evaluation. We noticed a stark difference in the count of entries for the datasets. Subtask_a has around 14,000 tweets, while the number reduces to approximately 5,000 for subtask_b and 3,000 for subtask_c. This preempts us that performance of the models will deteriorate as our training size is significantly lower for subtask_b and subtask_c..

Alongside decreasing frequency with each subtask, it is also important to note the difference in the class distribution. All 3 subtasks have a class imbalance.

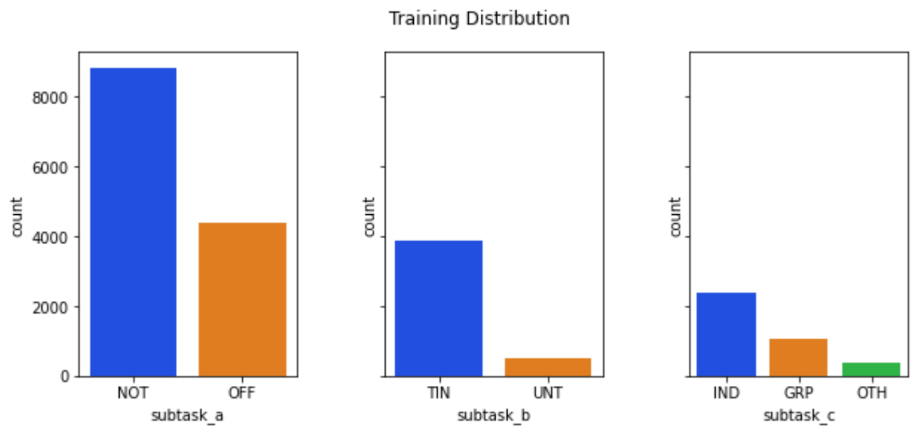


Figure 6: Training Distribution

The test data, shown in Figure 7, has similar distribution patterns as the training dataset. The data for each of the subtasks and the classes are also skewed in similar proportions.

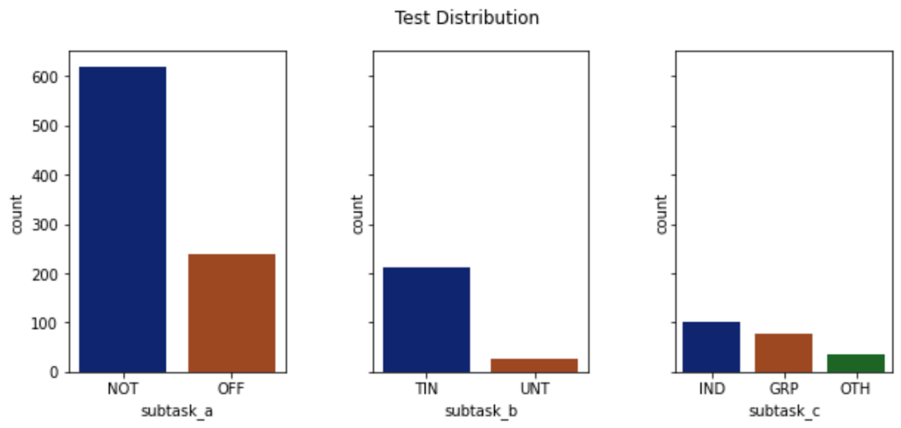


Figure 7: Test Distribution

After viewing the breakdown of the datasets and their classes, we decided to examine the textual data of the tweets. The first approach we took was to create a word cloud to help visualise the content of the tweets. As seen in figure 8, the Twitter data utilised for this project is largely from American users or users with interest in American politics. This can be seen from words such as 'MAGA', which stands for Make America Great Again, 'gun control', 'conservative' and 'liberal'. The prevalence of such words and their largeness reiterates the bias of the dataset towards tweets from the United States. This can be taken as one of the limitations of the projects as the model may not be able to effectively identify offensive tweets from other parts of the world.

As the usernames have been replaced with the user tag, it can be seen as one of the most frequently occurring words. The same occurs for the URL tag which replaces the URLs in the tweets. Further analysis would have to be done before making the decision of whether or not to remove them.

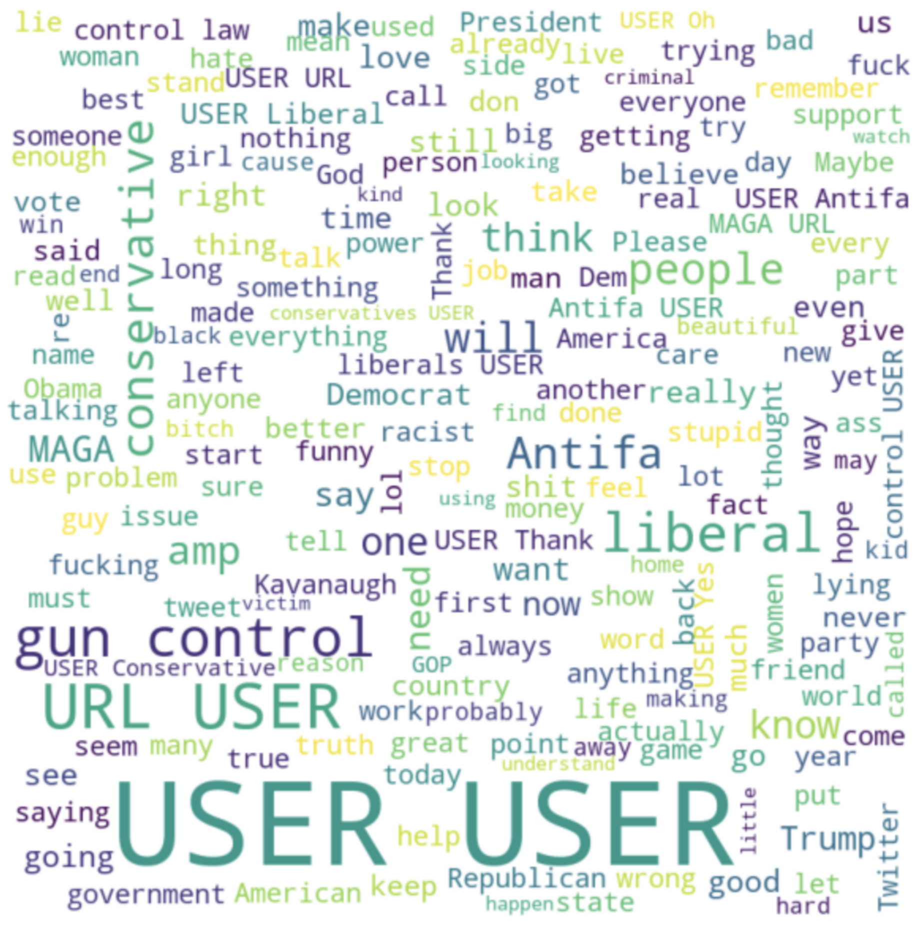


Figure 8: Word Cloud Uncleaned Data

Following the word cloud, we decided to analyse the n-gram sequences to see if we should consider the sequences of the word, or considering the word itself would suffice. We got the data for the top 20 single words, bigrams and trigrams for each of the levels according to their classes. This allowed us to look at keywords that could contribute to differentiating the classes for each level and if there are certain redundant words that can be removed before pushing the data into the model.

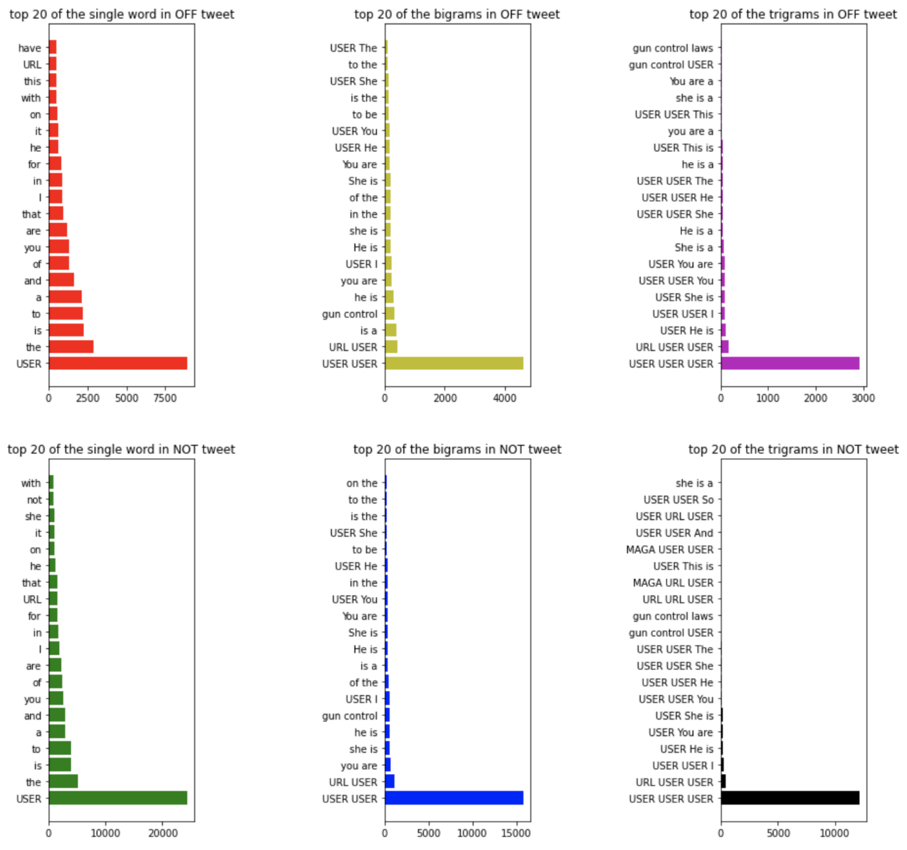


Figure 9: N-Gram for Subtask A

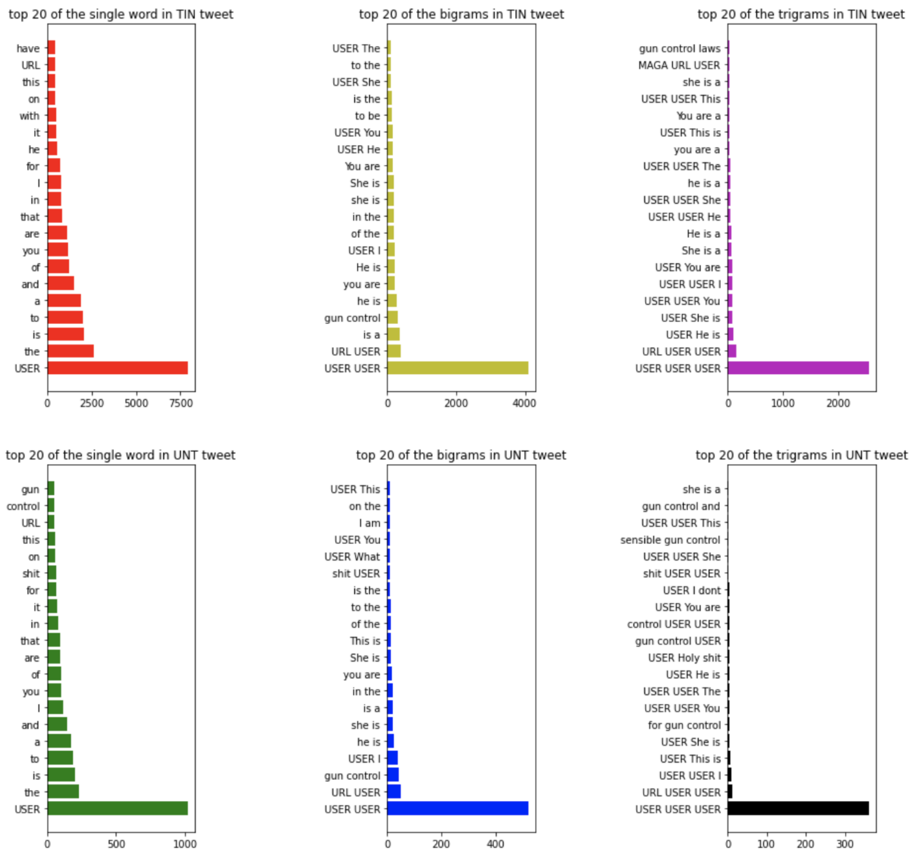


Figure 10: N-Gram for Subtask B

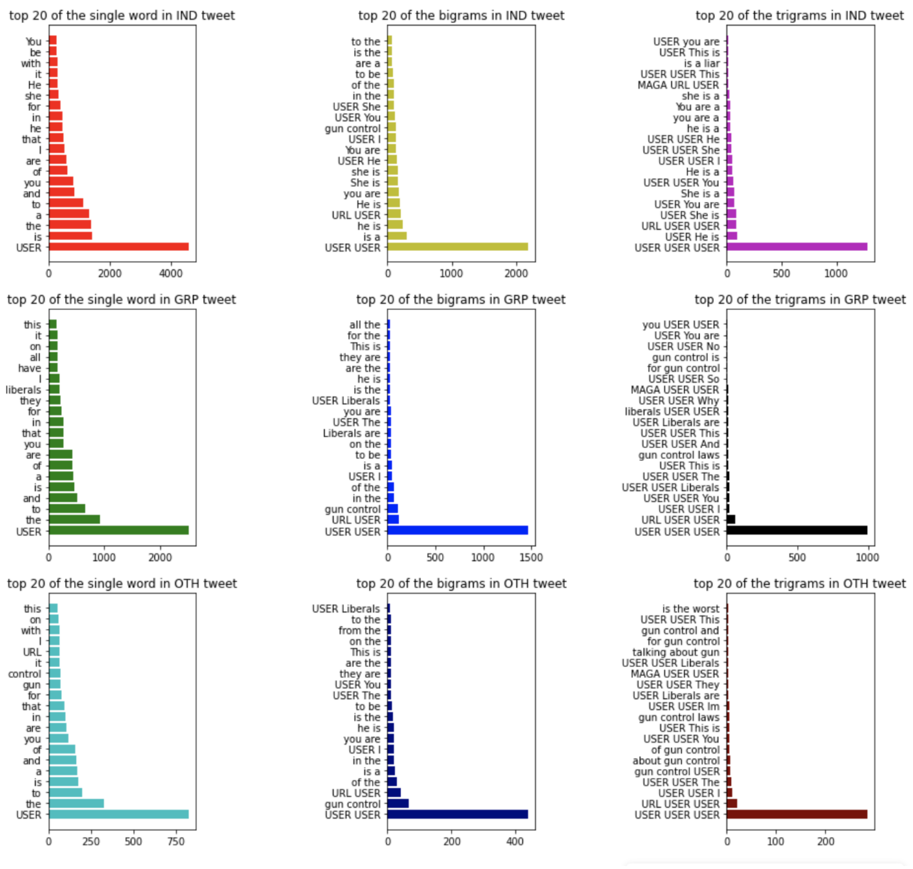


Figure 11: N-Gram for Subtask C

From Figure 9, we can observe that the most commonly occurring word for level A is that of 'USER'. This is consistent through the other levels as well. Given that 'USER' or an iteration of it occurs in the case of a single word, bigram and trigram, it can be viewed as having a large influence in the dataset. However, it does not aid much in the separation of the classes. Based on this analysis we considered dropping the word user from the dataset.

The difference in proportions of the subsequent words is not as significant as that between 'USER' and the next word. The words also tend to be pronouns, such as 'I', 'he' 'you' and words such as 'the', 'is'. While the order is slightly varied, the majority of the words are repeated across the classes. For instance 'you' is present in both 'OFF' and 'NOT' in level A. The same goes for level B where 'the' is seen to be the second most popular for both 'TIN' and 'UNT' classes. This indicates that it would be important to filter out the stop words as they do not add much value in differentiating the classes.

Adding on more words, bigrams and trigrams does add more context to the words but seem to be relatively similar to the words highlighted in a single word. For instance, in level C, we can observe gun control law occurring across the 3 classes, ‘IND’, ‘GRP’, ‘OTH’. This is similar to that of the single words, where gun takes place in all the 3 categories. As such, the added value of factoring in trigrams or bigrams may be grossly limited.

As we planned on using GloVe embedding for our model, we decided to utilise feature reduction to gain a feel and intuition of how the data is arranged in high-dimensional space. Given that the dataset has a strong American bias, we decided to test words that were appropriate for it. As seen in Figure 12, ‘trump’ is seen to be closer to ‘president’ than ‘biden’ who was not the president at that time. We also explored other words, such as that of countries as shown in Figure 13 which shows how Western countries are grouped away from China.

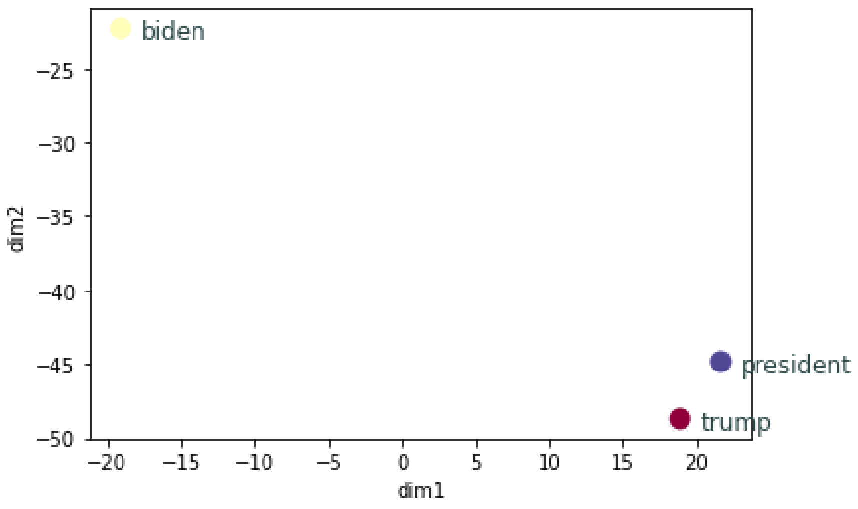


Figure 12: TSNE Diagram, American Politics

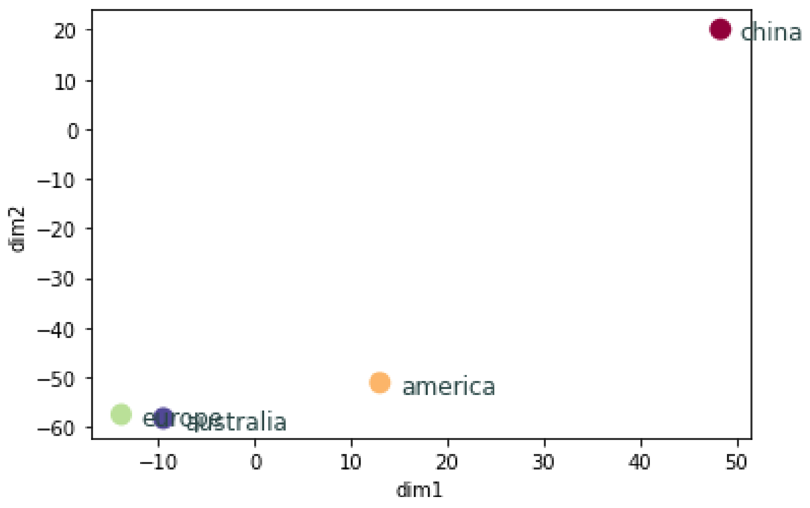


Figure 13: TSNE Diagram, Countries

The last visualisation that we carried out was based on our deeper understanding of the dataset and the way it was classified. As mentioned earlier, pronouns seemed to be occurring at a very high rate but did not seem to be adding much value. This was especially so for level A where the tweet is being classified as offensive and not offensive. However, upon a closer inspection, we can observe that they may have implications for level B and C. In level B for instance, we are trying to ascertain if a tweet is targeted. Including pronouns might be a factor that would help improve the model's performance, especially as those targeting someone would almost definitely use pronouns. The same can be said of level C, where the type of pronoun used could determine the size of the group the person is targeting, for instance, if it is an individual it would be 'she' or 'he' while if it is a group it would be 'they'. This was the conclusion we came to based on our understanding of the English language. To ascertain this we decided to view the distribution of such words across the different classes.

As seen from Figure 14, the singular words, such as 'I', 'he', 'she', occur a great deal more for the 'IND' class in level C as opposed to the 'GRP' and 'OTH'. This is in line with our expectations as singular words would be used to characterise individuals. The opposite can be said about the plural words which occur a great deal more in the 'GRP' class. While the counts of words for each class in plural words may seem closer than that of singular, it is important to note that the instances of each class, 'IND', 'GRP' and 'OTH' are not the same. They approximately have a ratio of IND: 8, GRP: 3 and OTH: 1.

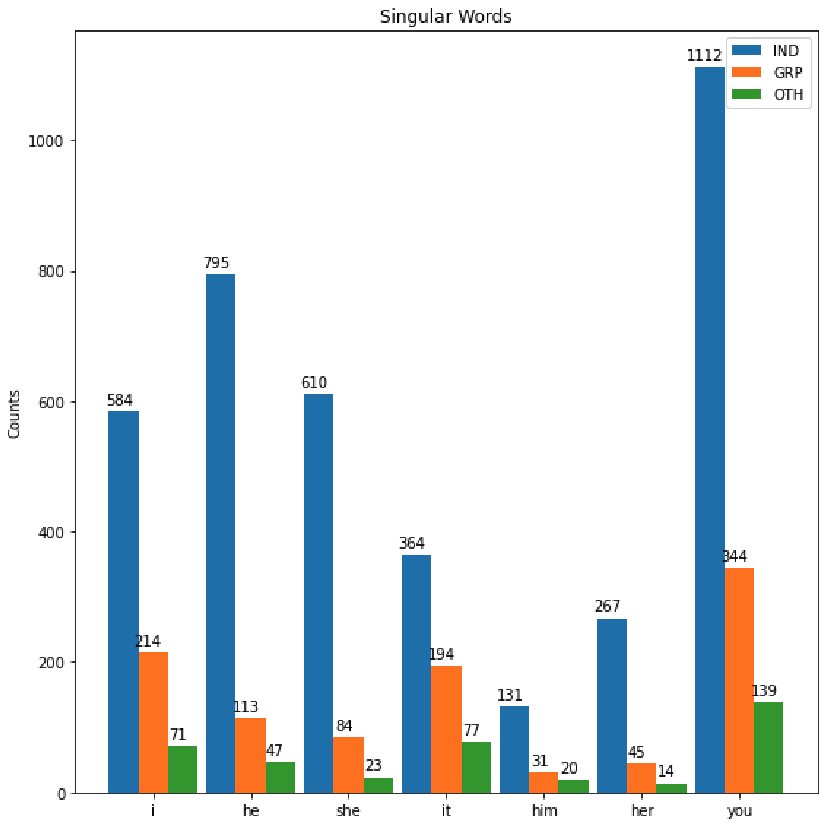


Figure 14: Distribution of Singular Words

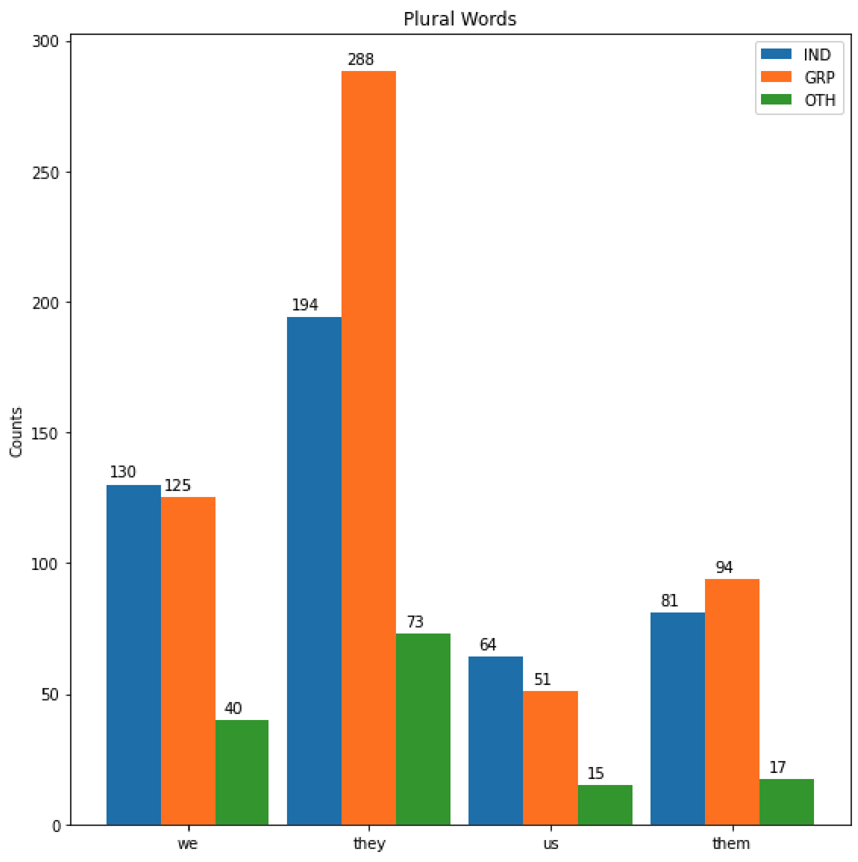


Figure 15: Distribution of Plural Words

6 Data Pre-Processing

The data pre-processing utilised the knowledge gained from the data visualisations to help better curate our model for peak performance.

6.1 Probing

We started off the preprocessing by probing to identify the impact of modifying the vocabulary, by adding and removing words from the tweet. Based on the distribution of singular and plural words above, we hypothesised that by removing those words from the stop words filter, the performance of our models would improve. To test it out, we added and removed certain words that might have an impact on the results and generated scores based on a simple logistic regression model. The key metrics that we considered were Recall and F1 Macro.

	Subtask A	Subtask B	Subtask C
	F1 Macro, Recall (OFF)	F1 Macro, Recall Macro	F1 Macro, Recall Macro
Without Any Changes to Stop Words	0.59, 0.52	0.48, 0.57	0.40, 0.48
Removing Additional Words [‘url’, ‘ha’]	0.59, 0.54	0.46, 0.53	0.41, 0.48
Adding Back Singular and Plural Words	0.59, 0.51	0.48, 0.56	0.39, 0.46
With Addition and Removal of Words	0.58, 0.53	0.46, 0.54	0.39, 0.45

The results shown were slightly different from what we expected. We had initially hypothesised that adding the back the singular and plural words would help improve the model’s performance in level B and level C. This was backed by Figure 15 that showed that there was a correlation between the occurrence of singular words and the ‘IND’ class alongside plural words and the ‘GRP’ class. However, this did not translate into an improvement in the model’s performance as the model’s performance was better without making any changes. The removal of additional words, however, did prove to improve the model’s performance for A and C. This was in line with our initial hypothesis as both these words occurred a large number of times across the classes and did not aid in differentiating between the classes. Based on the results from the probing, we decided to add [‘url’, ‘ha’] into the stopword list and let the singular and plural words remain in the stopword list as keeping them did not seem to improve the model’s performance.

6.2 Reformatting the Training Data

As mentioned earlier, there was only 1 training file provided to us. The drawback of this was that while training for levels B and C, there was an excessive amount of null data. This would be redundant in terms of our application as if a tweet is considered not offensive, ‘NOT’, it would not be sent to the subsequent layer. As such we came out with an individual training file for level A, B and C.

6.3 Removal of Hashtags, Reserved Words, Emojis

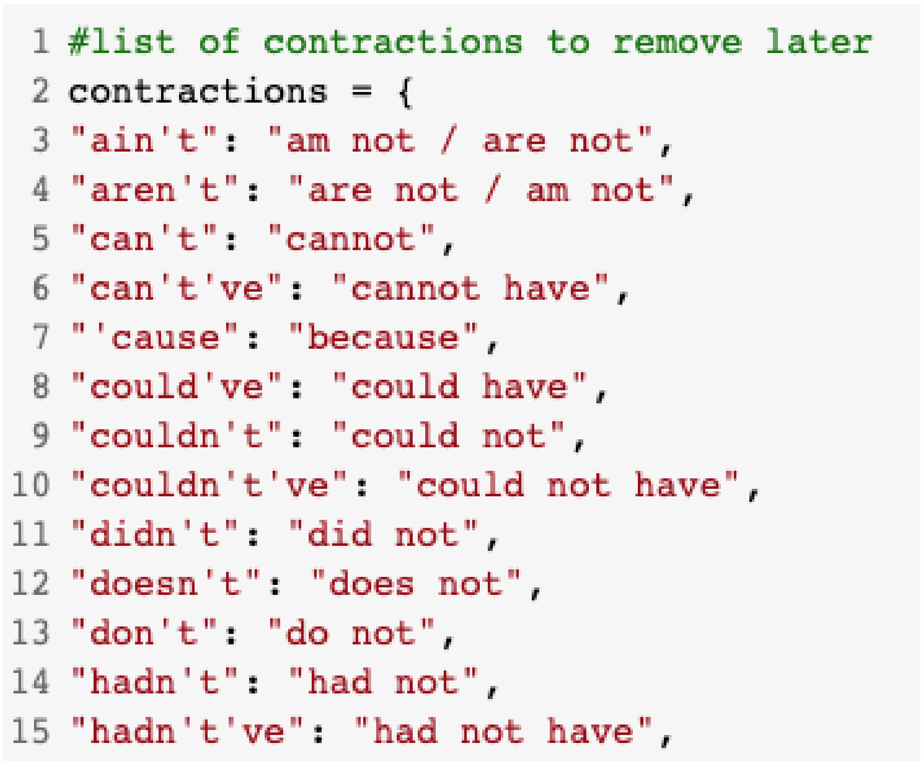
The dataset that we acquired had replaced the URLs with the tag ‘URL’. As such to remove the URLs we had to remove the ‘URL’ string itself, in section 6.9. We utilised the tweet preprocessor library to filter out the Hashtags, Reserved Words as well as Emojis. These were deemed to be non-essential for the classification of offensive tweets.

6.4 Changing to Lowercase

The words were then changed to lowercase to ensure that we captured the meaning of the words instead of unnecessary variations of it. We did believe that to a certain extent they may have played a role, for instance, typing in all caps could mean that a user is agitated which increases the chance of the tweet being offensive but found this case to be relatively rare based on literature reviews.

6.5 Handling Apostrophe

Being a means of informal communication, several tweets had contractions such as ‘ain’t’, ‘how’d’ and ‘i’ve’. These contractions are the same as ‘am not’, ‘how do you’ and ‘i have’. As such, we found a dictionary that had the keys as the contractions and the values as the expanded form of it. This helped to ensure consistency in the data and make the tweets clearer.



```
#list of contractions to remove later
contractions = {
"ain't": "am not / are not",
"aren't": "are not / am not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
```

Figure 16: Contraction Dictionary

6.6 Removal of Punctuations and Numbers

Punctuations were also removed as they did not really add much value while trying to discriminate between offensive and not offensive tweets. The same was for the case of numbers as in the majority of the cases they were not used in determining if a tweet could be considered offensive.

6.7 Spelling Correction

There were a number of tweets that had spelling mistakes in them. One of the consequences is that the dataset would have several iterations of the same word which were meant to convey the same meaning from the users' perspective. As such we utilised the SymSpell python library to fix the spellings of the tweets.

6.8 Lemmatization of Words

Next, we had to decide between Stemming and Lemmatization as the general advice based on literature was that both should not be utilised together. Given that by this step the spelling of the words had been corrected, we felt that Lemmatization was the better choice.

6.9 Removal of Stop Words

Instead of removing all the stop words, we decided to keep some of those that were deemed essential in discriminating between the classes, as shown in figure 14 and figure 15. As such, we formed 2 lists, that had some of the words that we believed should be added, such as 'he', 'she' and 'they' and another list that had certain words that we felt should be removed as they did not add much meaning to the tweets, such as 'URL'.

6.10 Word Cloud Following Cleaning of Tweets

Following the cleaning of the tweets, this is the word cloud we attained. The words attained provide meaning unlike the initial visualisation above. The apparent bias towards American tweets/context is also noticeable by the prevalence of words such as 'conservative', 'gun control' and 'trump'.

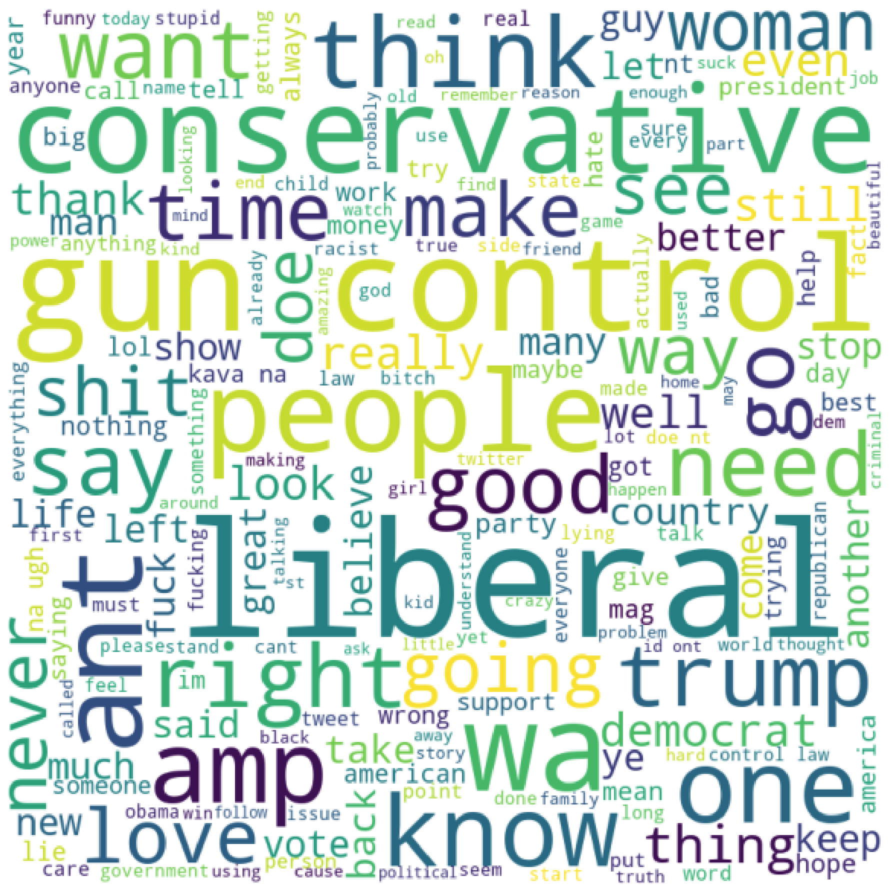


Figure 17: Cleaned Word Cloud

7 Text Representation Evaluation

For our dataset which consists of tweets, we noted that the tweets contained a variety of words and hashtags. Some tweets may contain offensive words but the tweet may not be offensive in reality. On the other hand, tweets that do not contain offensive words may in fact, be offensive. To capture the relationship between words and how they influence words that come before and after them, we hypothesised that using Twitter GloVe (pre-trained word vectors from Twitter) as the representation for the words in tweets would produce more meaning.

Thus, our approach was to carry out the experimentation of a logistic regression model on our validation set using the different word vectors namely, TFIDF, Word2Vec and GloVe.

Subtask A	Subtask B	Subtask C
Recall (Offensive Class)	F1 Macro	F1 Macro
0.68	**0.59**	**0.50**

Table 1: Logistic Regression Results (TF-IDF)

Subtask A	Subtask B	Subtask C
Recall (Offensive Class)	F1 Macro	F1 Macro
0.64	0.51	0.38

Table 2: Logistic Regression Results (Word2Vec)

Subtask A	Subtask B	Subtask C
Recall (Offensive Class)	F1 Macro	F1 Macro
0.55	0.50	0.37

Table 3: Logistic Regression Results (GloVe)

Based on the results we obtained, we noticed that TFIDF across all subtasks. Thus, we decided to run our models on our test data and noticed that TFIDF performed much worse than when using GloVe. We attributed this reasoning to the fact that we trained our model on both the training and validation sets, rather than unseen data points. Thus, the accuracy of using a logistic regression model with TFIDF was the highest when we tested our model on the validation testing. After running our models on the test data and noting that GloVe performed the best, we decided to follow through with GloVe for the remainder of our experimentation with different models.

8 Evaluation Metrics

In this project, our dataset for offensive and not offensive tweets were imbalanced. To account for this imbalance, we set the models to be 'balanced' during the training phase and looked at the F1 macro score and the Recall score for our loss functions.

8.1 Precision

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

i.e. Precision is the proportion of predicted positives were correctly classified

8.2 Recall

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

i.e. Recall is the proportion of actual positives were correctly classified

8.3 F1 Score

$$F_i = \frac{2P_iR_i}{Pi + Ri}$$

i.e. F1 Score is the harmonic mean between Precision and Recall where the score balances the 2 metrics for each label.

8.4 Macro F1 Score

$$F_{MACRO} = \frac{F_1 + ... + F_n}{n}$$

i.e. Macro F1 Score is the average of the per-class F1 scores where n is the number of classes.

For subtask A, we focused on getting a higher recall score for tweets that are offensive rather than a higher precision score. This is because we believe that predicting tweets to be offensive (even though they may/may not be offensive) is more suitable for this project where we focus on detecting offensive tweets to prevent them from being published.

As for subtasks B and C, we focused on achieving a high F1 macro score. This is because, in subtask B, we have labels 'UNTARGETED' and 'TARGETED' while in subtask C, we have labels 'GROUP', 'INDIVIDUAL' and 'OTHERS'. These labels are equally important as they determine the accuracy of our models. Thus, we looked into achieving a high F1 macro score for subtasks B and C.

8.5 Validation Loss and Accuracy Optimisation in Neural Network

In order to obtain optimal weights for our trained neural network models, we monitored the validation loss by implementing Early Stopping. We preferred monitoring the loss over accuracy as loss is able to quantify the certainty of our model on a prediction (1/0) while accuracy only accounts for the number of predictions.

9 Statistical Models

9.1 SMOTE

From the visualisation of the data earlier, one of the issues we identified is that they were imbalanced across the various subtasks A, B and C.

One approach we decided to proceed with was oversampling. Given the relatively small datasets, especially for subtasks B and C, undersampling was not considered. One of the relatively naive solutions that we found to work relatively well was setting the class weight to balanced for certain models. As expected the outcome of this was positive for one of the key metrics we were focusing on, recall score for 'OFF' class in level A and 'TIN' class in level B. This can be seen in Table 4 which shows a significant difference in the recall score when the models were balanced. However, there was a slight drop in the F1-score macro average, 0.01. Given the priority in classifying the tweets as 'OFF' and 'TIN', it can be said that the result is satisfactory.

Model	Recall (Offensive Class)
Logistic Regression A (w/o Balanced)	0.240
Logistic Regression A (w Balanced)	0.530
Logistic Regression B (w/o Balanced)	0.000
Logistic Regression B (w Balanced)	0.450

Table 4: Comparison with and without Balanced

We further discovered Synthetic Minority Oversampling Technique (SMOTE). SMOTE works by synthesizing new minority instances between existing instances. This not only leads to a rebalancing but generalises the decision region of the minority class. The results did show a slight improvement across the 3 subtasks, A, B and C, as seen from Table 5 but they were not too significant. One of the reasons for this is that SMOTE does not work too well with text data because of the high dimensionality of data that is created from it. However, we still proceeded with SMOTE given an overall improvement.

Model	F1 Macro Score
Logistic Regression (w/o SMOTE)	0.580
Logistic Regression (w SMOTE)	0.590
SVM (w/o SMOTE)	0.590
SVM (w SMOTE)	0.590
Random Forest Classifier (w/o SMOTE)	0.590
Random Forest Classifier (w SMOTE)	0.600

Table 5: Comparison with and without SMOTE

9.2 Results (using Cross Validation)

To ensure that the statistical models we trained on our training data generalize well on unseen data, we performed cross-validation. More specifically, we performed 10-fold cross-validation. This enabled us to determine the best hyperparameter settings such as the learning rate, the number of iterations and the number of trees for certain models. We then noted the F1 macro score for each model and the results can be seen in table 6, 7 and 8 for subtask A, B and C respectively.

Model	Recall (Offensive Class)
Logistic Regression (SMOTE)	0.610
Polynomial SVM (SMOTE)	**0.656**
Gaussian Naive Bayes (SMOTE)	0.606
Random Forest Classifier (SMOTE)	**0.685**
Gradient Boosting (SMOTE)	0.610
ADA Boosting (SMOTE)	0.612
XGBoost (SMOTE)	**0.649**

Table 6: Model Comparison for Level A

Model	F1 Macro Score
Logistic Regression (SMOTE)	0.686
Polynomial SVM (SMOTE)	**0.802**
Gaussian Naive Bayes (SMOTE)	0.688
Random Forest Classifier (SMOTE)	**0.868**
Gradient Boosting (SMOTE)	0.716
ADA Boosting (SMOTE)	0.750
XGBoost (SMOTE)	**0.866**

Table 7: Model Comparison for Level B

Model	F1 Macro Score
Logistic Regression (SMOTE)	0.501
Polynomial SVM (SMOTE)	**0.647**
Gaussian Naive Bayes (SMOTE)	0.488
Random Forest Classifier (SMOTE)	**0.699**
Gradient Boosting (SMOTE)	0.545
ADA Boosting (SMOTE)	0.484
XGBoost (SMOTE)	**0.8682**

Table 8: Model Comparison for Level C

9.3 Discussion of Results

Based on the F1 macro score we obtained by using cross-validation for all models on subtask A, B and C respectively, we noted that Random Forest Classifier, XGBoost and SVM classifiers had the highest F1 macro score for subtasks B and C and had the highest recall score for subtask A.

By creating several decision trees that learn from a training set, the random forest classifier produces robust predictions. Given that we are classifying tweets based on whether they are offensive or not, Random Forest Classifier is suitable for dealing with high dimensional noisy data in text classification. As can be seen in Figure 11, there are many tweets that contain similar words such as 'USER' and 'gun control'. This adds to the noise factor of our dataset. Given that Random Forest classifier consists of a set of decision trees which are trained using a random subset of features, this classifier helps to deal with the noisy data present in our dataset and also makes use of different features to make decisions. Similar to Random Forest Classifier, XGBoost is a decision-tree-based ensemble Machine Learning algorithm. Thus, both models work relatively well compared to the other models for our tweet classification.

As for Support Vector Machines (SVM) classifiers, their ability to learn is independent of the dimensionality of feature space. In addition to that, SVM classifiers are able to generalise even in the presence of many features if the data is separable. Given the high dimension of our data consisting of various tweets, SVM was able to work well for our subtasks A, B and C.

As for Gradient Boosting, we noted that it may not be a good choice for noisy data as it may result in overfitting and thus, it becomes harder to tune as compared to Random Forest Classifier. Also, given that Random Forest Classifier involves the development of independent decision trees on different samples in the data, it is less susceptible to overfitting. The main advantages of Random Forest Classifier over Gradient Boosting as well as Ada Boosting is that it is less affected by noise and generalizes better, reducing variance. Thus, we concluded that Random Forest Classifier performs the best as compared to other Boosting algorithms such as XGBoost, Ada Boosting and Gradient Boosting.

Gaussian Naive Bayes, on the other hand, does not work as well when training and validating our data. This is because Gaussian Naive Bayes is a probabilistic classifier that is suited for continuous data.

Thus, based on factors including the run time of our models and the F1 macro score, we decided to proceed with Random Forest Classifier, XGBoost and SVM to conduct the training and testing on all three subtasks (A, B and C). The results are shown in the table below.

	Subtask A	Subtask B	Subtask C
	Recall (Offensive Class)	F1 Macro	F1 Macro
Random Forest Classifier	0.37	**0.56**	0.37
SVM (polynomial for A and B, radial basis function for C)	**0.47**	0.51	**0.43**
Random Forest Classifier (SMOTE)	0.46	0.52	0.40

Table 9: Best Model Results From Statistical Models

After noting the results obtained for subtasks A, B and C, we found that the scores obtained for recall for subtask A and the score obtained for F1 macro for subtask C was not as high as the score obtained for F1 macro for subtask B. We thus decided to continue experimenting with other models and decided to look into Neural Network models as well.

10 Neural Network

10.1 Basis of Neural Netwrok Layer Selection

To justify the layer selections in our neural networks, we read up on various scientific literature on modelling NLP tasks. We came up with 4 basis in constructing our neural networks .

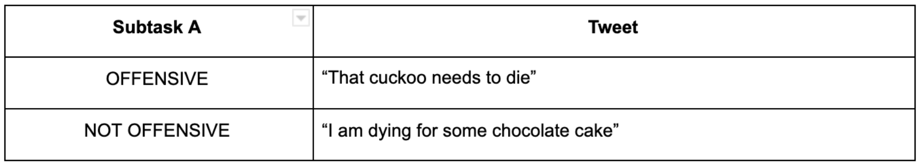
CNN Layer
Convolutional layer with Max Pooling in a 1-dimensional space will help us in extracting higher-level features in a sentence. Also, stacking multiple CNN layers in increasing filter sizes would allow a hierarchical decomposition of the inputs, abstracting our text features to higher levels.

LSTM Layer
Long Short Term Memory layer helps circumvent the vanishing gradient problem of our RNNs in capturing the long term dependencies. This will be especially useful for tweets that have a longer sequence, enabling the model to pick up features throughout the sentence. Also, stacking LSTM layers are understood to recombine the learned representation from prior layers and create new representations at higher levels of abstraction.

Bi-Directional LSTM Layer
Apart from simply applying an LSTM, we posit that using a BiLSTM layer will help capture information from two opposite directions of a sentence (past

and future). This makes full use of the contextual information rather than just picking up information from past in regular LSTM.

Self Attetion Layer
Within a tweet, it is definite that not all contextual words have an equal contribution to semantics. To address this, we posit that using a self-attention mechanism, we can reweigh word vector embeddings by relating words in each tweet, giving more context to a sentence.



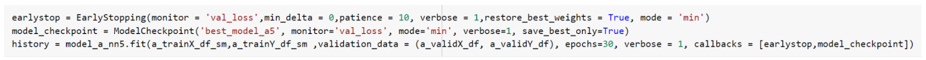
Subtask A	Tweet
OFFENSIVE	“That cuckoo needs to die”
NOT OFFENSIVE	“I am dying for some chocolate cake”

Figure 18: Self Attention Tweet Example

For example in the offensive tweet, a self-attention layer will pick up that a combination of ‘cuckoo’ + ‘die’ gives a context of offensiveness. Meanwhile, in the non-offensive tweet, a combination of ‘Dying’ + ‘Cake’ gives a context of non-offensiveness.

10.2 Preventing Overfit

In ensuring that our neural network models are optimized to increase accuracy and prevent overfitting, we made a conscious choice of minimizing validation loss at each epoch by applying an early stopping and checkpoint function. As validation loss is the sum of errors made for each example in validation sets, it implies how poorly or well a model behaves after each iteration of optimization. Benign able to recall the best model that optimizes validation loss prevents overfitting on our test dataset



```
earlystop = EarlyStopping(monitor = 'val_loss',min_delta = 0,patience = 10, verbose = 1,restore_best_weights = True, mode = 'min')
model_checkpoint = ModelCheckpoint('best_model_a5', monitor='val_loss', mode='min', verbose=1, save_best_only=True)
history = model_a_nn5.fit(a_trainX_df_sm,a_trainY_df_sm ,validation_data = (a_validX_df, a_validY_df), epochs=30, verbose = 1, callbacks = [earlystop,model_checkpoint])
```

Figure 19: Early Stopping and Model Checkpoints Functions

As a precautionary measure to check for the most optimized epoch, we plotted the validation loss and accuracy at each epoch.

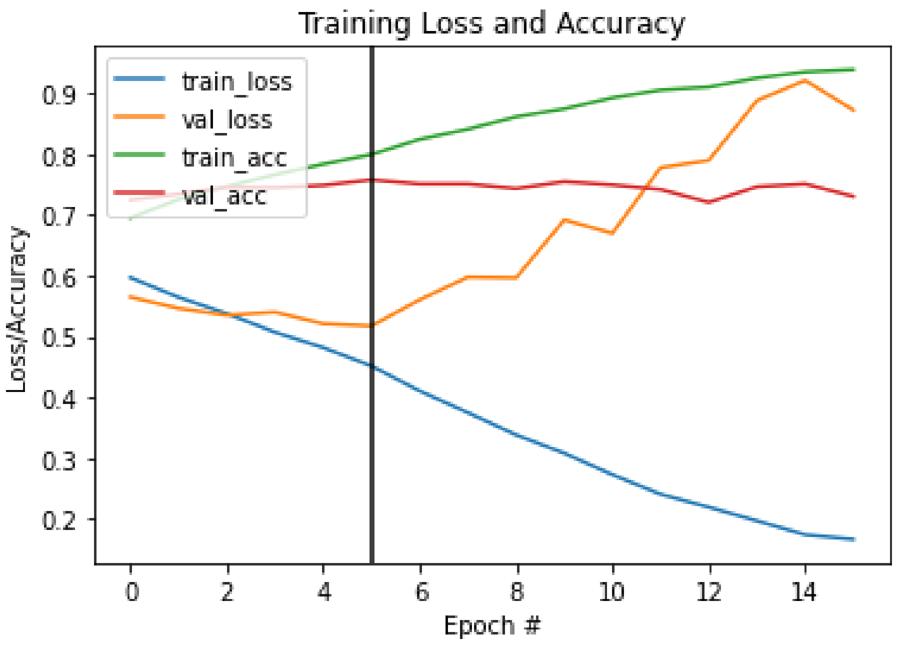


Figure 20: Validation Loss and Accuracy Visualisation

For example, in the plot above, we observe that after the 5th epoch, validation loss increases while validation accuracy slowly decreases. The 5th epoch will be taken as the best training epoch for model fitting.

To further prevent overfitting, we consciously applied dropout layers after each convolutional layer, pooling layer and activation layer to mute features. Using a dropout of 0.2 between convolutional layer and a dropout of 0.5 after activation layers seemed to work best for our models

10.3 Results

Subtask	Best Performing Neural Network Model	METRICS	STATS MODEL BENCHMARK
A	Stacked BiLSTM - Self Attention - CNN	**Recall (OFFENSIVE): 0.53**	Recall (OFFENSIVE): 0.47
B	Stacked CNN - LSTM	Macro F1: 0.52	Macro F1: 0.56
C	Stacked CNN - BiLSTM	**Macro F1: 0.56**	Macro F1: 0.43

Table 10: Best Model Results From Neural Network Models

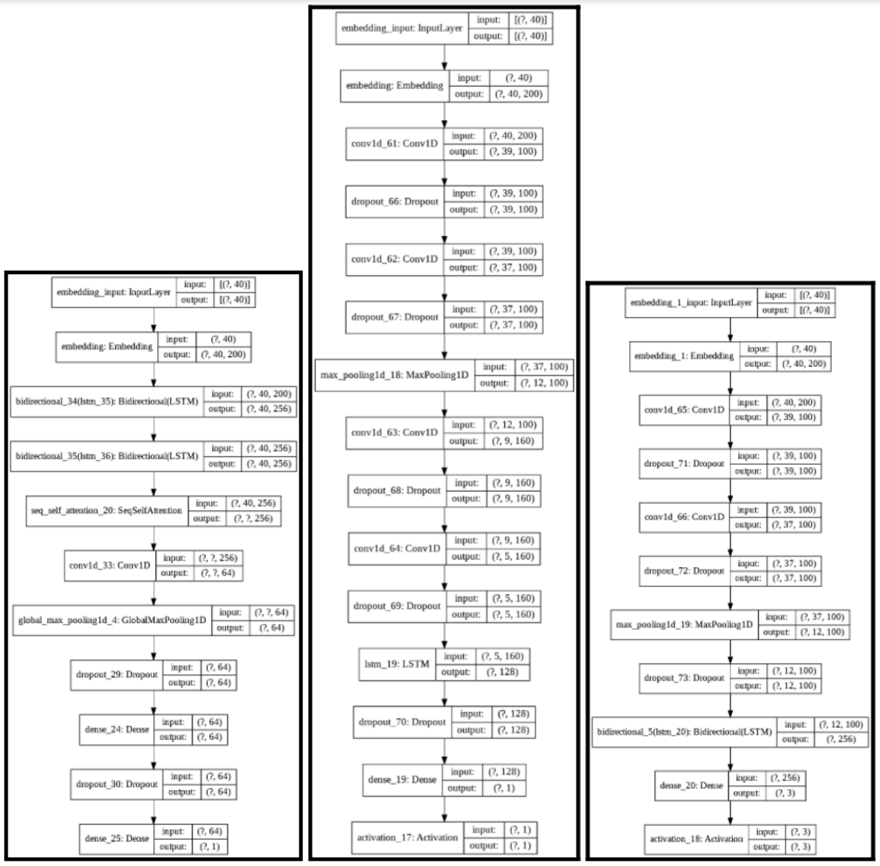

embedding_input: InputLayer input: [(?, 40)] output: [(?, 40)]
embedding: Embedding input: (?, 40) output: (?, 40, 200)
bidirectional_34(lstm_35): Bidirectional(LSTM) input: (?, 40, 200) output: (?, 40, 256)
bidirectional_35(lstm_36): Bidirectional(LSTM) input: (?, 40, 256) output: (?, 40, 256)
seq_self_attention_20: SeqSelfAttention input: (?, 40, 256) output: (?, ?, 256)
conv1d_33: Conv1D input: (?, ?, 256) output: (?, ?, 64)
global_max_pooling1d_4: GlobalMaxPooling1D input: (?, ?, 64) output: (?, 64)
dropout_29: Dropout input: (?, 64) output: (?, 64)
dense_24: Dense input: (?, 64) output: (?, 64)
dropout_30: Dropout input: (?, 64) output: (?, 64)
dense_25: Dense input: (?, 64) output: (?, 1)
embedding_input: InputLayer input: [(?, 40)] output: [(?, 40)]
embedding: Embedding input: (?, 40) output: (?, 40, 200)
conv1d_61: Conv1D input: (?, 40, 200) output: (?, 39, 100)
dropout_66: Dropout input: (?, 39, 100) output: (?, 39, 100)
conv1d_62: Conv1D input: (?, 39, 100) output: (?, 37, 100)
dropout_67: Dropout input: (?, 37, 100) output: (?, 37, 100)
max_pooling1d_18: MaxPooling1D input: (?, 37, 100) output: (?, 12, 100)
conv1d_63: Conv1D input: (?, 12, 100) output: (?, 9, 160)
dropout_68: Dropout input: (?, 9, 160) output: (?, 9, 160)
conv1d_64: Conv1D input: (?, 9, 160) output: (?, 5, 160)
dropout_69: Dropout input: (?, 5, 160) output: (?, 5, 160)
lstm_19: LSTM input: (?, 5, 160) output: (?, 128)
dropout_70: Dropout input: (?, 128) output: (?, 128)
dense_19: Dense input: (?, 128) output: (?, 1)
activation_17: Activation input: (?, 1) output: (?, 1)
embedding_1_input: InputLayer input: [(?, 40)] output: [(?, 40)]
embedding_1: Embedding input: (?, 40) output: (?, 40, 200)
conv1d_65: Conv1D input: (?, 40, 200) output: (?, 39, 100)
dropout_71: Dropout input: (?, 39, 100) output: (?, 39, 100)
conv1d_66: Conv1D input: (?, 39, 100) output: (?, 37, 100)
dropout_72: Dropout input: (?, 37, 100) output: (?, 37, 100)
max_pooling1d_19: MaxPooling1D input: (?, 37, 100) output: (?, 12, 100)
dropout_73: Dropout input: (?, 12, 100) output: (?, 12, 100)
bidirectional_5(lstm_20): Bidirectional(LSTM) input: (?, 12, 100) output: (?, 256)
dense_20: Dense input: (?, 256) output: (?, 3)
activation_18: Activation input: (?, 3) output: (?, 3)

Figure 21: Best Model Results From Neural Network Models

10.4 Discussion of Results

Overall, neural networks in subtask A and C outperformed that of statistical models.

Since subtask A contains the most number of tweets, we believe it is suited for our BiLSTM - Self Attention - CNN layered neural network in maximizing higher level feature extraction, hence outperforming statistical models due to the model's ability to pick up semantics in a tweet. The same cannot type of layering cannot be said for subtask B and C. This can be attributed to the significantly smaller sample size for subtask A and B, hence a much smaller pool of features to be extracted. The same layers applied on B and C proved to produce poor results under a benchmark score of 0.5.

For subtask B , our multi layered perceptron CNN model did not outperform that of Random Forest model used. This can be attributed to Random Forest's ability to deal with high dimensional noise that is inherent within tweets. Moreover, it is likely that a dense multi layered neural network would overfit such a small sample of subtask B.

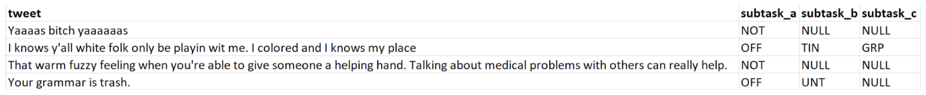
For subtask, our layered CNN - BiLSTM model outperformed our statistical models. Although this was surprising since its layered complexity would lead to overfitting, we did account for subtask C sparsity by increasing the rate of dropout beyond the extent applied for subtask A and B. We attribute this to the results seen.

11 Overall Results and Discussion

11.1 Public Dataset Testing Results on Subtask A, B, c

In order to determine if our pre-processing and model tuning helps to generalise and is not overfitted in terms of evaluation, we proceeded to evaluate our best performing models on unseen data.

Our team manually labelled 200 Tweets in order of subtask A, B, C. These tweets are generically offensive and non-offensive, and not skewed towards any event or occurrence in particular.



tweet	subtask_a	subtask_b	subtask_c
Yaaaas bitch yaaaaaas	NOT	NULL	NULL
I knows y'all white folk only be playin wit me. I colored and I knows my place	OFF	TIN	GRP
That warm fuzzy feeling when you're able to give someone a helping hand. Talking about medical problems with others can really help.	NOT	NULL	NULL
Your grammar is trash.	OFF	UNT	NULL

Figure 22: Example of Manually Labelled Tweets

Only 1 person was assigned to label the tweets to maintain consistency. For identifying 'Targeted' and 'Untargeted' tweets, any ambiguity detected like 'you' and 'your' without directing towards a user would be deemed offensive. Consequently, any offensive tweets directed towards a user with "@" would be deemed targeted.

11.2 Public Dataset Results

	Subtask A	**Subtask B**	**Subtask C**
MODEL	BILSTM - SelfAttention - CNN	Random Forest	CNN-LSTM
F1 Macro	0.64	**0.46**	**0.40**
Recall	OFF: **0.81**, NOT: 0.46	TIN: 0.64, UNT: 0.33	GRP:0.21, IND: 0.91, OTH: 0.21

Table 11: Public Data Results with Best Models

We observed encouraging results from our models, well beyond the benchmark from OLID dataset. In Subtask A, since it is more important to detect an offensive class, a recall score of 0.81 showed that the BiLSTM - Self Attention Model performed well. We can attribute these results to a relatively sufficient sample of 14,000 tweets in training the model. Our multi-layer perceptron BiLSTM, CNN layers helped to maximize the extraction of features including long term dependencies, as well as contextual recognition within the tweet with our self-attention layer.

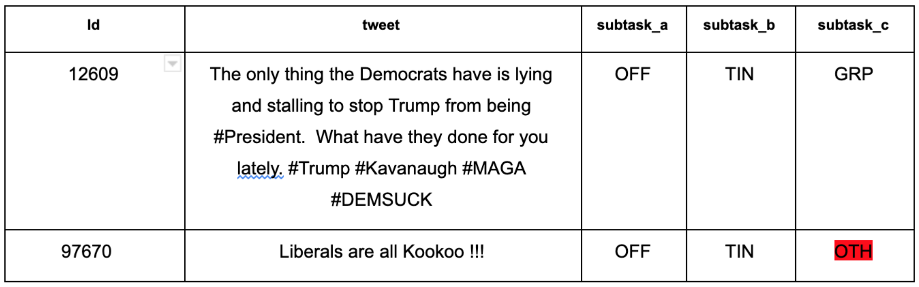
However, for Subtask B and Subtask C, we observe an F1 Macro score of 0.46 and 0.40 respectively. The results showed underperformance compared to our benchmarking with the OLID dataset. We can attribute these results to the minute sample sizes for training subtask B and C (5000 and 3000 tweets respectively). Hence, this could have resulted in overfitting due to the small pool of features that can be extracted.

11.3 Limitations

During analysis and preprocessing of the data, we realized that there are certain characteristics inherent in the dataset. Majority of the tweets were pertaining to American politics between 2016-2019, containing details of American elections and race rights. This makes it likely for any model trained to be high invariance and low in bias (overfitted) towards this specific tweet distribution.

Although our manually labelled public dataset produced relatively positive results, the majority of the tweets were purely foul or neutral in intention. If we had used a separate dataset, for e.g offensive Singaporean tweets, we believe that results would have been less than desirable given the grammatical structure and linguistic differences between OLID and Singaporean tweets.

Moreover, on further inspection of training and test data in OLID. We also observed that there were certain discrepancies inherent in the dataset. This was especially so for the determination of subtask C, a targeted tweet towards ‘Individual’, ‘Group’, or others.



Id	tweet	subtask_a	subtask_b	subtask_c
12609	The only thing the Democrats have is lying and stalling to stop Trump from being #President. What have they done for you lately. #Trump #Kavanaugh #MAGA #DEMSUCK	OFF	TIN	GRP
97670	Liberals are all Kookoo !!!	OFF	TIN	OTH

Figure 23: OLID Dataset Discrepancy

Looking at Figure 23, we would think that tweet 97670 should be classified as ‘GRP’ since it targets liberals, similar to how tweet 12609 targets democrats and is labelled as ‘GRP’. There are many such divergence seen across the OLID dataset. Thus, the results obtained from subtask C could be interpreted as less reliable compared to A and B.

11.4 Improvements

One of the possible improvements would be to factor in emoji description when classifying the tweets. During our data pre-processing stage, we removed emojis as they added complexity to the data. This was especially significant as we were utilising GloVe, which would not have been able to take in emojis as inputs. A possible workaround would have been to convert the emojis into textual data. For instance ‘:)’ could be converted into ‘smiling face’. Through this approach we would be able to capture a greater deal of meaning in our tweets, possibly leading to better results.

Moreover, since sparse samples of data for subtask B and C is a consistent barrier for our task, we could also explore data augmentation for tweets in future iterations of this project, preventing overfitting and limitations to neural network implementations.

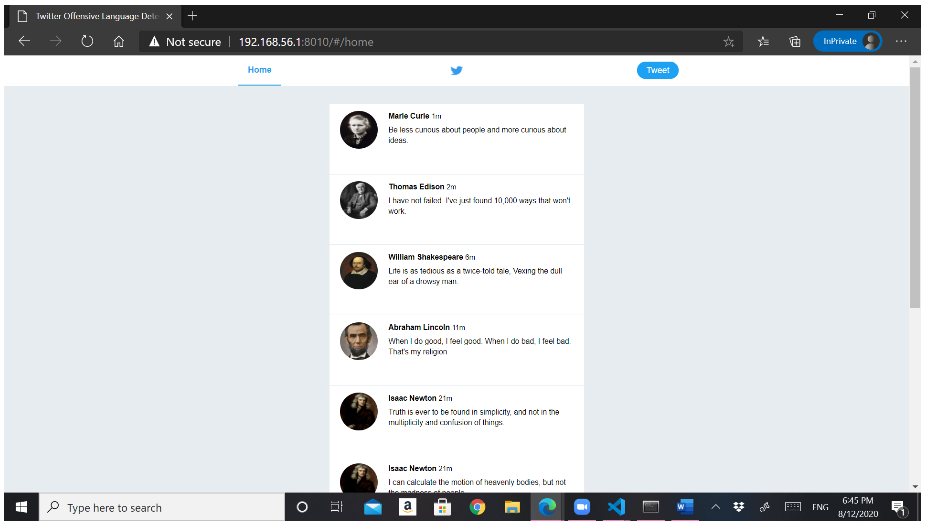
Alongside that, we would like to consider more powerful and recent word vector representation tools such as FastText, BERT and ELMo. The main advantage of utilising tools such as BERT and ELMo is that we would be able to capture the context of the word, which is the location it is present in a sentence.

12 Conclusion

Overall, this hierarchical, multi-class text classification problem spurred our team to apply our knowledge acquired from class and more. Rather than a straightforward application of the models, it was a learning experience to explore the scientific literature of algorithms utilized, and justifying why certain algorithms work better than others.

13 UI/UX Interface

For our UI, we built a webpage that mimics the homepage of Twitter using Vue JS framework (Figure 24). As a user clicks on the tweet button and types in their tweet (Figure 25), our machine learning model cleans the tweet and carries out the predicting of whether the tweet is offensive or not. For tweets that are deemed not offensive, the results will be shown on the page (Figure 26). In addition to that, when the user clicks on the homepage, their tweet will be published (Figure 27). On the other hand, for tweets that are deemed offensive, the results will also be shown on the page (Figure 28). Tweets that are deemed offensive will not be published despite the user clicking the tweet button.



Marie Curie 1m
Be less curious about people and more curious about ideas.

Thomas Edison 2m
I have not failed. I've just found 10,000 ways that won't work.

William Shakespeare 6m
Life is as tedious as a twice-told tale, Vexing the dull ear of a drowsy man.

Abraham Lincoln 11m
When I do good, I feel good. When I do bad, I feel bad. That's my religion

Isaac Newton 21m
Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things.

Isaac Newton 21m
I can calculate the motion of heavenly bodies, but not

Figure 24: Homepage of our UI

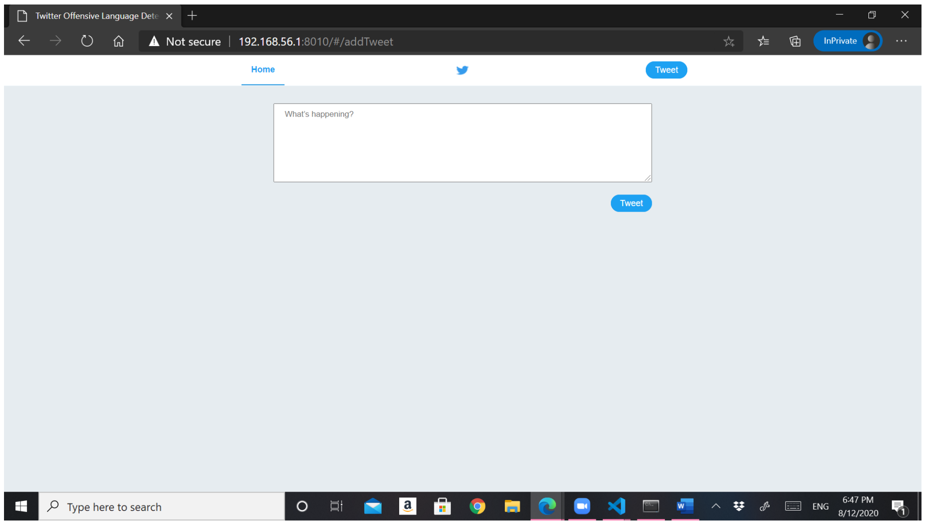
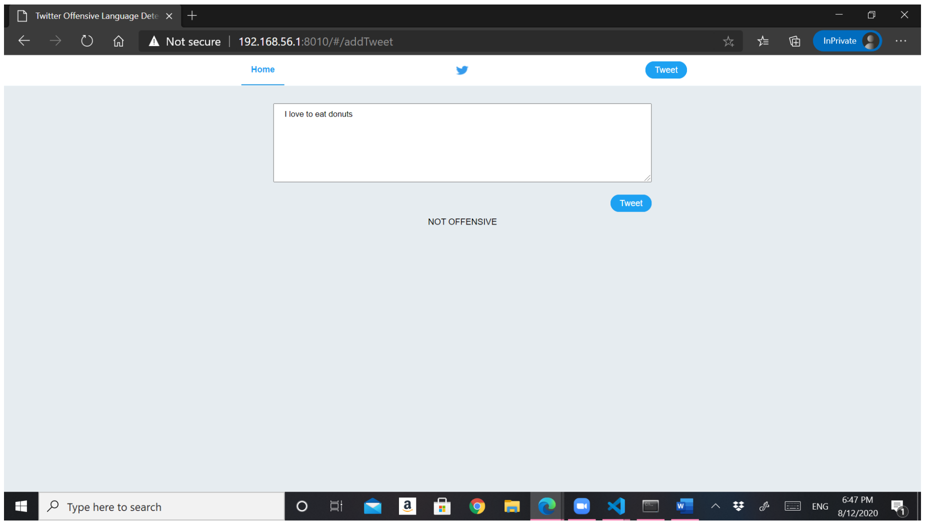


Figure 25: User Tweet Page


Twitter Offensive Language Dete

Not secure | 192.168.56.1:8010/#/addTweet

Home

Tweet

I love to eat donuts

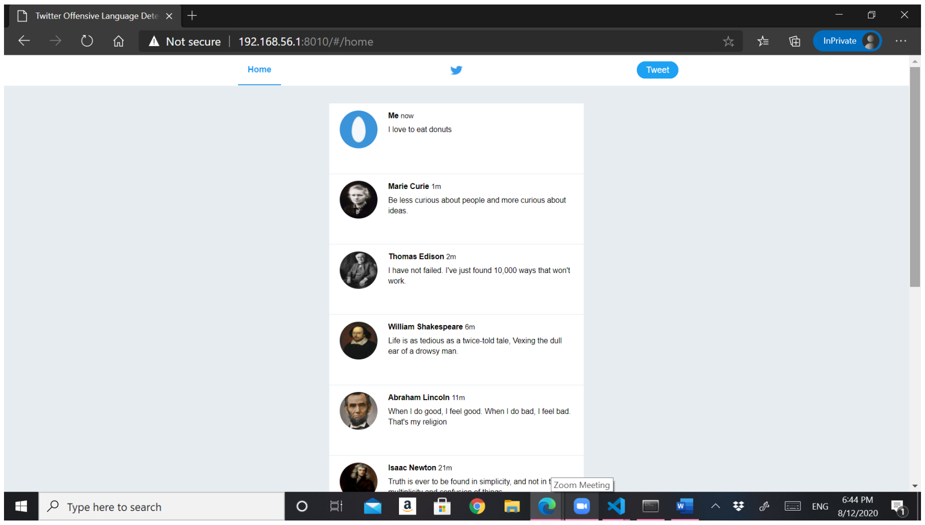
Tweet

NOT OFFENSIVE

Type here to search

6:47 PM
8/12/2020

Figure 26: Tweet marked as Not Offensive


Twitter Offensive Language Dete

Not secure | 192.168.56.1:8010/#/home

Home

Tweet

Me now
I love to eat donuts

Marie Curie 1m
Be less curious about people and more curious about ideas.

Thomas Edison 2m
I have not failed. I've just found 10,000 ways that won't work.

William Shakespeare 6m
Life is as tedious as a twice-told tale, Vexing the dull ear of a drowsy man.

Abraham Lincoln 11m
When I do good, I feel good. When I do bad, I feel bad. That's my religion

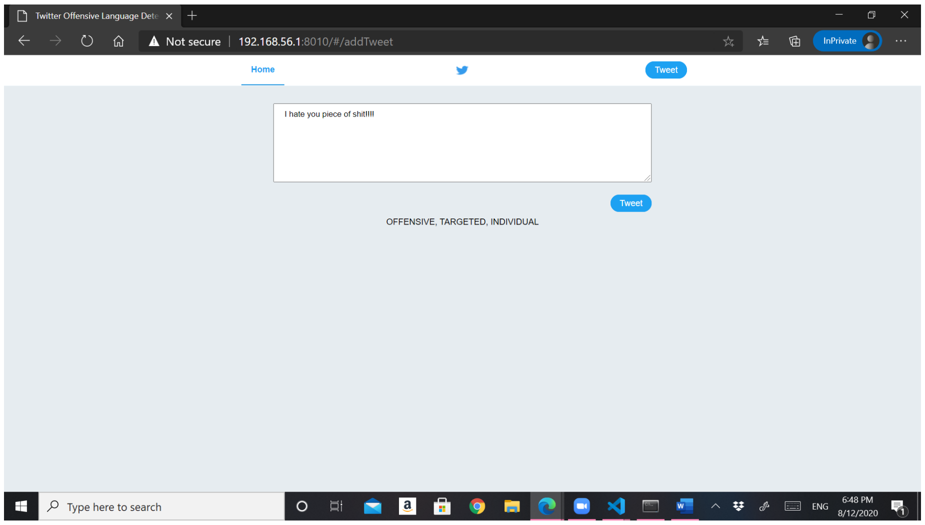
Isaac Newton 21m
Truth is ever to be found in simplicity, and not in t

Zoom Meeting

Type here to search

6:44 PM
8/12/2020

Figure 27: Tweet Published


Twitter Offensive Language Dete
Not secure | 192.168.56.1:8010/#/addTweet
InPrivate
Home
Tweet
Tweet
Type here to search
6:48 PM
8/12/2020

I hate you piece of shit!!!!

OFFENSIVE, TARGETED, INDIVIDUAL

Figure 28: Tweet Published

14 Bibliography

Laub, Zachary. “Hate Speech on Social Media: Global Comparisons” cfr.com, Jun 7, 2019. Web.

Edwards, Jim. “One statistic shows that Twitter has a fundamental problem Facebook solved years ago” businessinsider.com, Apr 17, 2015. Web.

Hu, Jiawei. “An Overview for Text Representations in NLP” towardsdatascience.com, Mar 5, 2020. Web.

scikit-learn. “sklearn.feature_extraction.text.TfidfTransformer” scikit-learn.org. Web.

Rebedea, Traian. “What are the advantages and disadvantages of TF-IDF?” quora.com, Nov 16, 2015. Web.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Bohm, Timo. “The General Ideas of Word Embeddings” towardsdatascience.com, Dec 30, 2018. Web.