TOXIC COMMENT CLASSIFIER USING CNN AND LSTM LAYERS

Group Members:
Aakarsh Mishra (21MIS1025)
Nancy Saxena (21MIS1024)
Utkarsh Mishra (21MIS1074)

School of Computer Science Engineering (SCOPE)
Vellore Institute of Technology
DATA SCIENCE CLUB
FFCS 2022-2023
Date: 08 April 2023

Abstract:

This report discusses the challenges of categorizing remarks due to the lack of organization and a wide range of vocabulary. Despite this, our approach was able to correctly identify comments with toxic behaviour. The model, which utilized CNN and LSTM, outperformed their baseline model and other models in the literature. However, it was still outperformed by the top Kaggle model. It was also found that preserving class balance is crucial for binary classifiers, and data augmentation helped to reduce false positives. It is believed that more advanced techniques, such as back-translation, could further improve the training set.

Introduction:

More people than ever are online due to the growing pandemic. In fact, Facebook is used actively by 84% of young Canadians to communicate with others. According to Shannon Martinez of the Free Radicals Project, the internet also provides negative people with "a safe area to explore extreme views and develop their hatred without consequence... until it bursts in the real world."

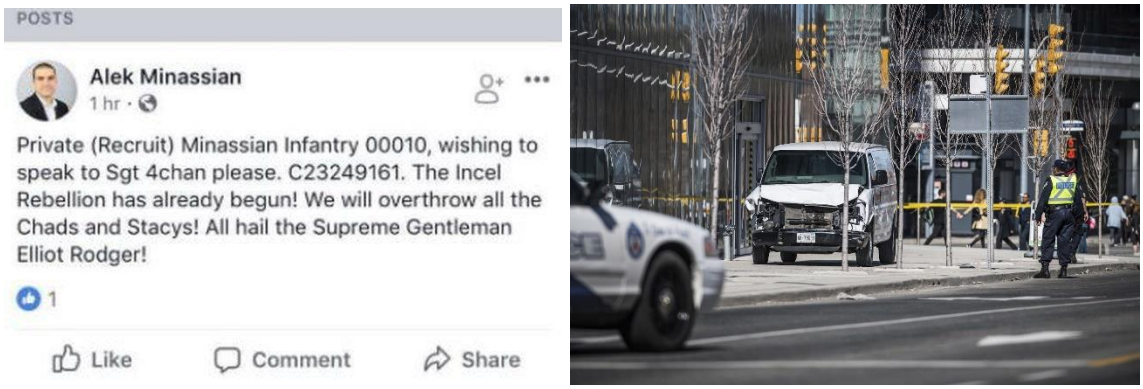


Figure 1: Alek Minassian, accused in the Toronto van attack, praised “incel rebellion.” on Facebook [3]

This effort aims to make the internet a safer place. Toxic comments on social media can be quickly reported and deleted by looking for them. Long-term, this would improve interpersonal connections in this increasingly digital environment.

Because it is difficult for hard-coded algorithms to interpret and recognise human speech and expensive for businesses to engage humans to identify these comments, a neural network is the most effective way to solve this problem. NLP using neural networks has been discovered to be the most efficient method in the industry, with a market size predicted to reach US$26.4 billion by 2024 [4].

Literature Review:

Prior to deep learning (NLP), businesses used inefficient techniques to detect hate speech, such as basic keyword searches (bag of word). This technique unintentionally eliminates everyday talk while having "good recall but leads to high percentages of false positives."

Deep learning research has already been done recently to recognise hate speech. Using data from numerous social media organisations, a work released in August 2019 employed multiple-view stacked

Support Vector Machine (mSVM) to reach over 80% accuracy. Another research from 2018 trains a CNN GRU model using a variety of word embeddings and achieves 90% accuracy on 3 different classes.

Also, a lot of social media businesses have invested in strategies to stop hate speech online. Facebook Canada announced in July 2020 that it is "collaborating with the Centre on Education at Ontario Tech

To establish what it refers to as the "World Network Against Hate", for which Facebook will provide $500,000 in order to identify online extremism and implement countermeasures.

Modules:

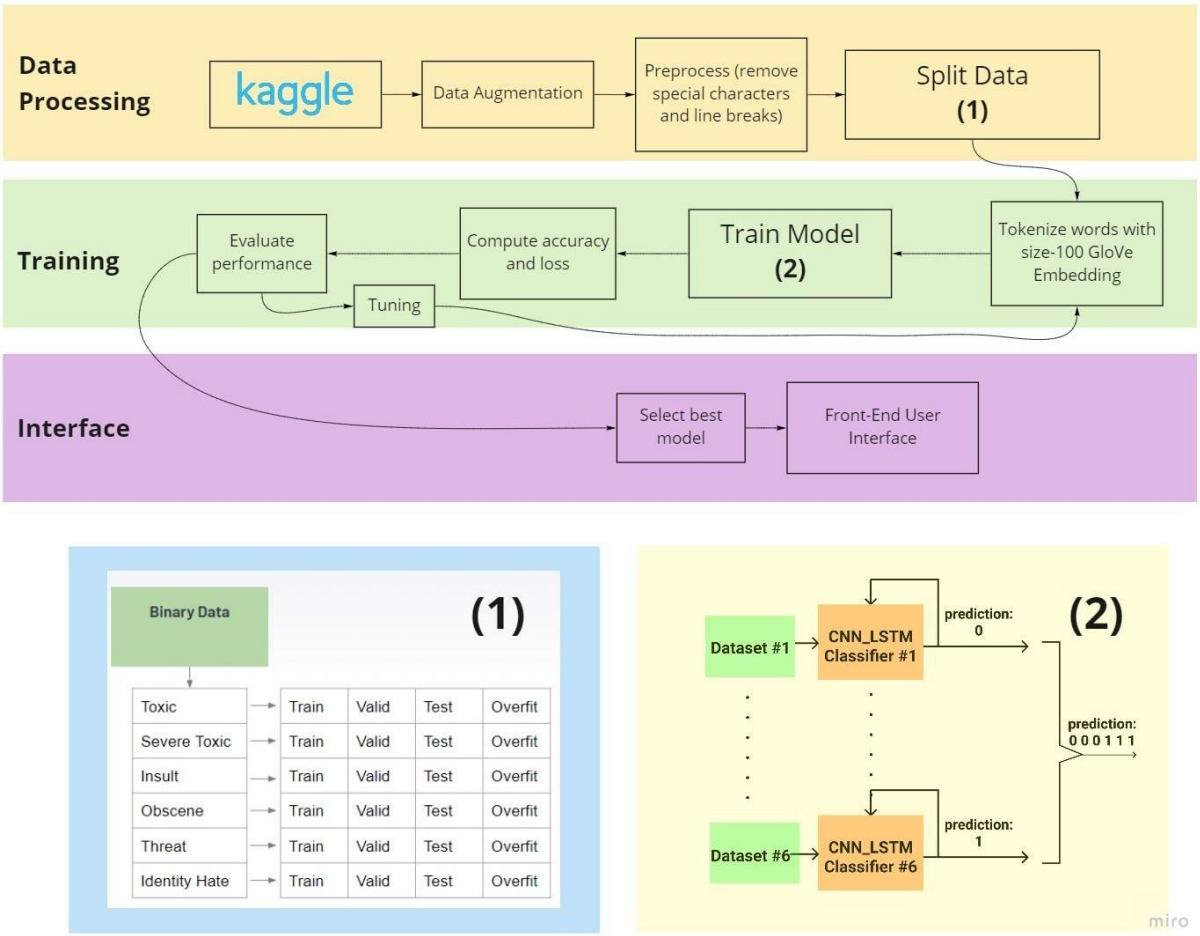


Figure 2: Model Production Pipeline

Data source- We used data from the "Toxic Comment Classification Challenge" on Kaggle, which includes individual human volunteers' labels for Wikipedia editors' comments . Comments are categorised into 6 categories: identity hate, obscene, threat, toxic, and severe toxic. For instance, a comment marked as 100100 is toxic and dangerous. Nevertheless, we discovered that the labels for some of the comments are inaccurate, so we manually relabelled them.

Examine data-Pandas was then used to evaluate the data. Figure 4 shows that there are more than 150000 total comments. Yet, due to the extremely low means in each class, the majority of the responses are positive. Hence, throughout the processing stage, we must balance the positive and negative comments.

	toxic	severe_toxic	...	insult	identity_hate
count	159571.000000	159571.000000	...	159571.000000	159571.000000
mean	0.095844	0.009996	...	0.049364	0.008805

Figure 4: the count and mean of the classes as output of .describe()

We also observed that certain classes lacked sufficient comments. Less than 500 of the more than 15000 comments marked as toxic are dangerous. It is simple for the models to overfit and produce a low accuracy for the validation and test set when there are so little comments in some classes.

obscene	8449
insult	7877
toxic	15294
severe_toxic	1595
identity_hate	1405
threat	478

Figure 5: Number of comments in each class

Data Augmentation- We utilised the package nlpaug to artificially enhance the amount of data in the minority classes by replacing synonyms in order to resolve the aforementioned problem [10]. The number of comments grew in the majority of courses as a result of the comments' multi-labelling.

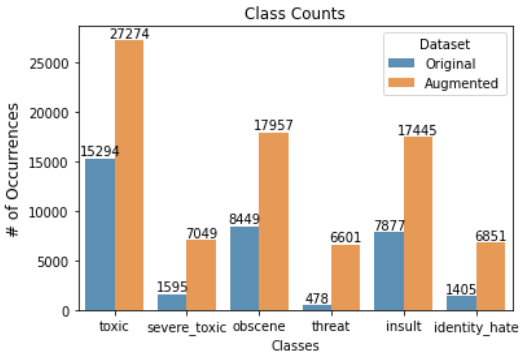


Figure 6: Amount of comments in each class before and after augmentation

```
[82] aug1 = naw.ContextualWordEmbsAug(
         model_path='distilbert-base-uncased', action="substitute")
     aug2 = naw.ContextualWordEmbsAug(
         model_path='roberta-base', action="substitute")
     aug3 = naw.SynonymAug(aug_src='wordnet')

     augmented_text1 = aug1.augment(text)
     augmented_text2 = aug2.augment(text)
     augmented_text3 = aug3.augment(text)
     print("Original: " + text)
     print("\nAugmented Text 1: " + augmented_text1)
     print("\nAugmented Text 2: " + augmented_text2)
     print("\nAugmented Text 3: " + augmented_text3)
```

```
Original: The quick brown fox jumps over the lazy dog

Augmented Text 1:the striped brown fox jumps over the muddy grass

Augmented Text 2:The quick brown fox jumps Into the bull dog

Augmented Text 3:The quick wild fox hurdles over the lazy dog
```

Figure 7: Example of augmented sentence

Data Cleaning- Using regex, we cleaned the data by identifying patterns in the comments and swapping them out for more logical alternatives. All blank spaces, line breaks, contractions, etc. were eliminated. Cleaner data results in a model that is more effective and accurate.

```
"id","comment_text","toxic","severe_toxic","obscene","threat","insult","identity_hate"
"0000997932d777bf","Explanation
Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closur
"000103f0d9cfb60f","D'aww! He matches this background colour I'm seemingly stuck with. Thanks.  (talk) 21:51, J
"000113f07ec002fd","Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing
"0001b41b1c6bb37e","""
More
I can't make any real suggestions on improvement - I wondered if the section statistics should be later on, or

There appears to be a backlog on articles for review so I guess there may be a delay until a reviewer turns up.
"0001d958c54c6e35","You, sir, are my hero. Any chance you remember what page that's on?",0,0,0,0,0,0
"00025465d4725e87","""

Congratulations from me as well, use the tools well.  · talk """,0,0,0,0,0,0
"0002bcb3da6cb337","COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK",1,1,1,0,1,0
id,comment_text,toxic,severe_toxic,obscene,threat,insult,identity_hate
0000997932d777bf,explanation why the edits made under my username hardcore metallica fan were reverted they were
000103f0d9cfb60f,d aww he matches this background colour i am seemingly stuck with thanks talk 21 51 january 11 2
000113f07ec002fd,hey man i am really not trying to edit war it just that this guy is constantly removing relevant
0001b41b1c6bb37e,more i cannot make any real suggestions on improvement i wondered if the section statistics shou
0001d958c54c6e35,you sir are my hero any chance you remember what page that on,0,0,0,0,0,0
00025465d4725e87,congratulations from me as well use the tools well talk,0,0,0,0,0,0
0002bcb3da6cb337,cocksucker before you piss around on my work,1,1,1,0,1,0
```

Figure 8: The same comments before (top) and after (bottom) cleaning

Data Processing- 10+ datasets were produced for model testing and prototype development. Six separate sub-datasets (one for each class) were produced in the final dataset, and each sub-dataset was divided into training, validation and sets. Each sub-dataset balances all the negative comments assigned to a certain class with an equal number of positive remarks.

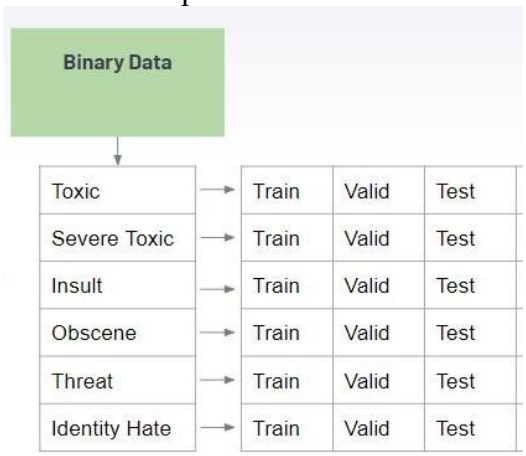


Figure 9: Visualization of dataset structure

Architecture:

The group employed 6 CNN LSTM Binary Classification Model to accomplish multi-label classification. With Embedding layer, the input data was first transformed into word vectors. We discovered that negative comments use comparable vocabularies. In order to recognise word patterns in texts independent of their position, two convolution layers were used. Each convolution layer has a maxpool layer that provides one output feature for each sentence from each kernel in order to adapt to changes in sentence length. The outputs were then combined to create a vector and sent into an LSTM (Long Short-Term Memory network), a unique class of RNN that can learn lengthy dependencies [11]. In order to decode the LSTM's output, a completely linked layer was used.

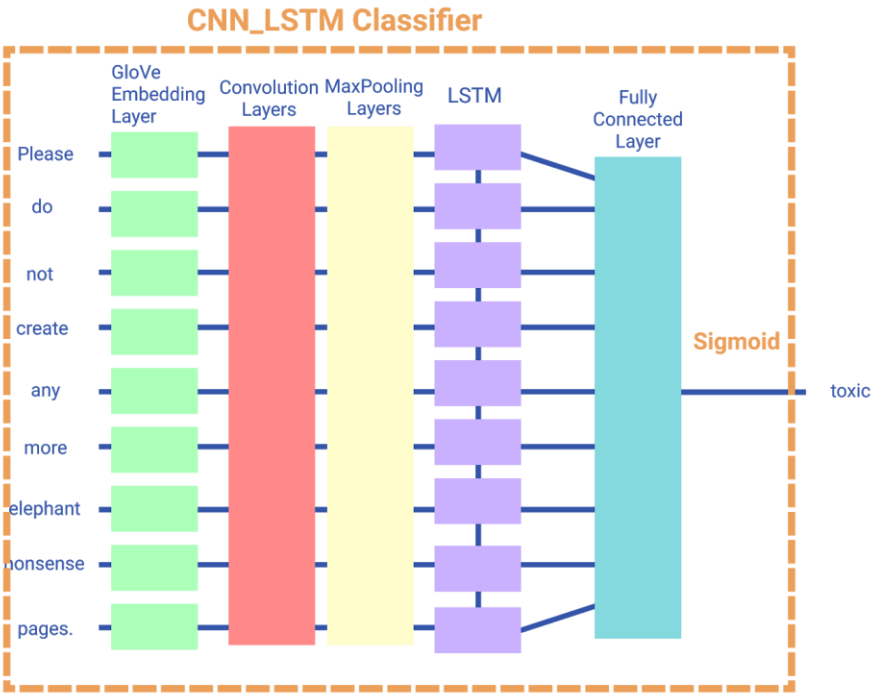


Figure 10: Model Architecture

Since the comment labels are multi-hot encoded, this study focuses on multi-labelling categorization. In order to guarantee that each classifier is fully trained, we used the training strategy depicted in Fig. 10. A CNN LSTM binary classifier was trained using 6 balanced binary datasets, each of which contained data from a different class. The outputs of the binary classifier were merged to provide a final forecast for each comment that was multi-hot encoded.

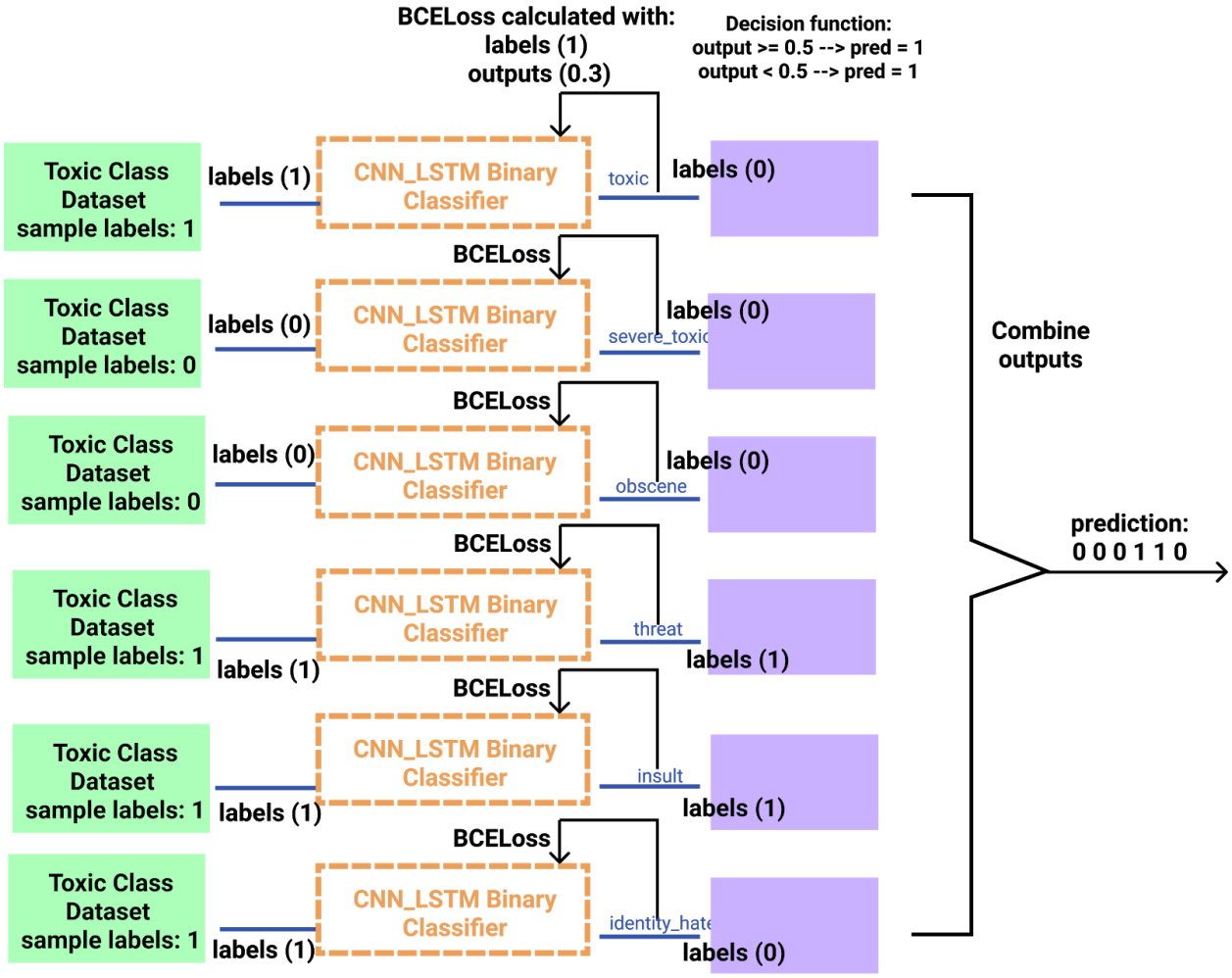


Figure 11: Best Training Approach

Baseline Model:

In order to create an output prediction for our base model, we first extract the glove embedding vectors for each word in each remark, average their values, and then feed this average vector through a fully connected layer. The model should be able to abstract some feeling of toxicity with just an average vector because comments are typically brief and comprised of few phrases.

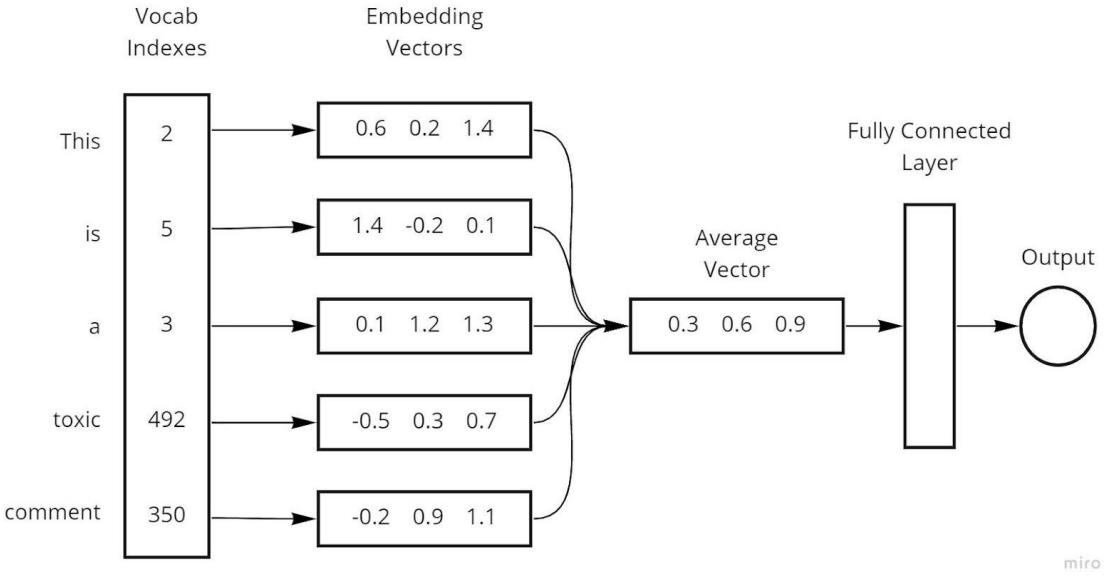


Figure 12: Baseline Model Pipeline

Quantitative Results:

The top-performing baseline model has an accuracy of 79.7%, a training loss of 0.617, and a testing loss of 0.619. This demonstrated that the feasibility of our project concept.

With the hyperparameters in Table 1, the ideal model was trained. The average loss and accuracy across the six binary classifiers are used to calculate the combined loss and accuracy. The most accurate model has an accuracy of 95.3%, a testing loss of 0.528, and a training loss of 0.510. The curves are smooth and exhibit exponential tendencies, as shown in Fig. 13. Overfitting is not evident in any observable way. The model's performance stabilised about epoch 10, while epoch 30 yielded the highest training and testing accuracy.

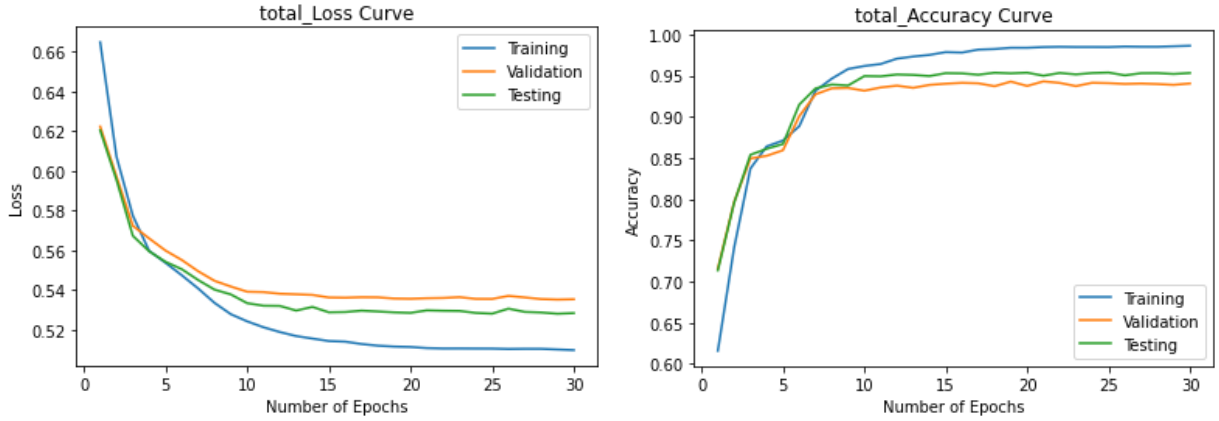


Figure 13. Loss and Accuracy Curve

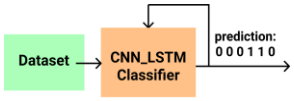
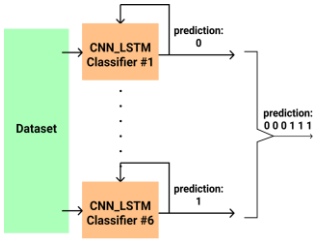
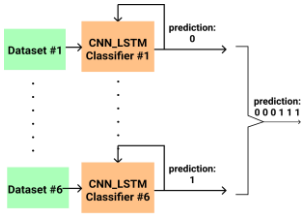
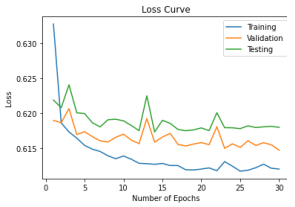
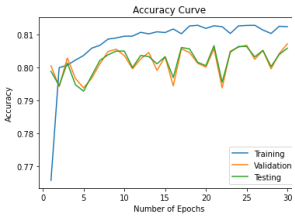
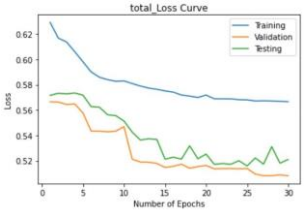
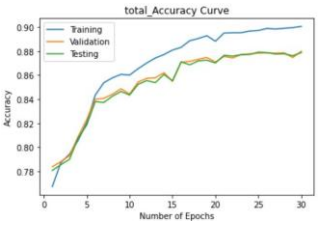
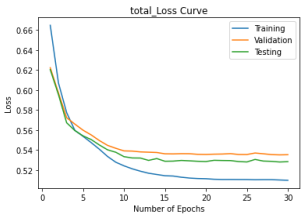
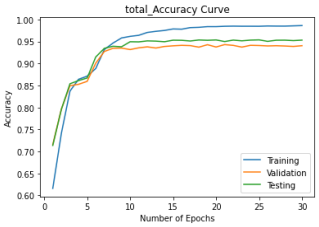
Table 1. Best Model Hyperparameters

CNN_LSTM Binary Classifier on Best Training Approach			
hidden_dim	100	# maxpool layer	2
embedding_dim	100	# conv2d layer	2
kernel_size	(2,100)	# LSTM	1
# kernels	50	# fc layer	1
batch_size	128	learning_rate	0.001
epochs	30	decision function	If output >= 0.5, prediction = 1 If output < 0.5, Prediction = 0

We trained the identical CNN LSTM classifier using each of the three methods shown in **Table 2** to assure the optimum training strategy. The first two methods were trained on an unbalanced, generic dataset. In method 3, six distinct balanced datasets were used to train the models. The third technique, which is the approach utilised in the best model, performs the best, as shown by the accuracy and loss data. With the other two procedures, the curves fluctuate and show overfitting.

Table 2. Three Training Approaches

	Approach 1	**Approach 2**	**Approach 3 (best)**
# classes in each dataset	**6**	**6**	**1**
Is # positive/negative labels in each class balanced?	**No**	**No**	**Yes**
# CNN_LSTM model	**1**	**6**	**6**
# Model Output	**6**	**1**	**1**
Training Loss	**0.612**	**0.612**	**0.510**
Training Accuracy	**81.2%**	**90.0%**	**98.6%**
Testing Loss	**0.618**	**0.537**	**0.528**
Testing Accuracy	**80.6%**	**87.9%**	**95.3%**

Model Architecture			
Loss and Accuracy Graphs	 	 	 

In the multi-label classification problem, the number of FP and the F1 score of the model should also be evaluated. Table 3 shows the confusion matrices of the 6 model classes tested with the balanced binary datasets. There are much more TP, TN than FP and FN. Moreover, the fact that most classes have fewer FP than FN proves that our best model gives desirable results. The F1 score for each class is higher than 90% thus showing that our classifying model has high precision and recall.

By comparing the F1 scores and the results shown in the confusion matrices, we notice that class toxic, has a relatively lower F1 score. It also gives more false predictions.

Qualitative Results:

Table 4. Sample Model Outputs, Predictions, and Actual Labels

Case	Comment	Value Type	Toxic	Severe_toxic	Obscene	Threat	Insult	Identity_hate
1	"Well that explains it thanks for clearing it up"	**Model Output**	3.3293 e-06	0.0003	2.5801e-05	0.0004	1.1351 e-05	0.0002
		Prediction	0	0	0	0	0	0
		Actual Label	0	0	0	0	0	0
2	"You are ugly I hate you."	**Model Output**	1.0000	0.9993	0.9999	0.9970	1.0000	0.9991
		Prediction	1	1	1	1	1	1
		Actual Label	1	1	1	1	1	1
3	"Hey loser get a life."	**Model Output**	0.9977	0.4768	0.9580	0.7029	0.9898	0.4662
		Prediction	1	0	1	1	1	0
		Actual Label	0	0	0	0	1	0
4	"You are just so freaking beautiful."	**Model Output**	1.0000	0.9928	0.9999	0.9553	0.9999	0.9949
		Prediction	1	1	1	1	1	1
		Actual Label	0	0	0	0	0	0

We selected 4 representative comments from the dataset displayed in Table 4 to provide further context on the model's performance. Just neutral vocabulary is included in Case 1, and the model was able to make accurate predictions. Words like "ugly" and "hatred," which are frequently used in negative comments, are present in Case 2. The forecast was accurate since the model recognised the words. In Case 3, however, the model's precision was just 50%. This is partially brought on by the dataset's labelling, which is inherently subjective. In case 4 scenario, the model's forecast is wholly incorrect. The model knew the word "freaking," but she misread what it meant in the phrase. This shows that while the model is proficient at recognising word patterns, it struggles to comprehend context.

Improvement:

It is challenging to categorise remarks because there is minimal organisation and a wide range of vocabulary. We are pleased with our testing outcomes given the difficulties of multi-label classification and dealing with a sizable amount of dirty, imbalanced data. Even if there are occasionally mistakes when identifying the precise classes, our approach is able to correctly flag comments that exhibit some hazardous behaviours. Some of our errors can potentially be related to incorrect labelling of the training data because toxicity is a subjective concept.

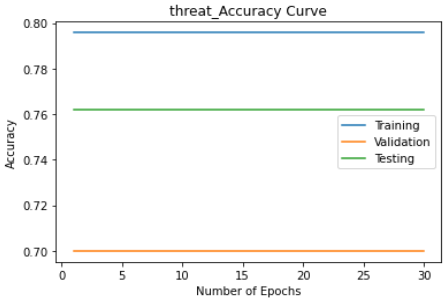
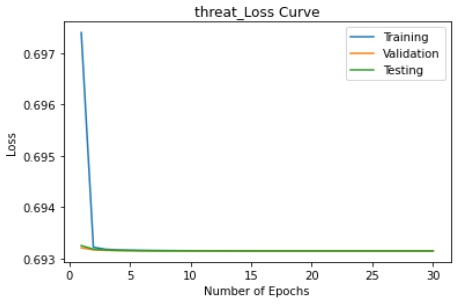
Our final model performed significantly better than our baseline model and models we cited in literature thanks to the implementation of CNN and LSTM in our architecture and passing in the comments word by word. This shows that extracting sequence and word patterns plays a role in determining the toxicity of a sentence.Our model still needs work, as it is outperformed by the top Kaggle model, which has an accuracy rate of 98.9%.

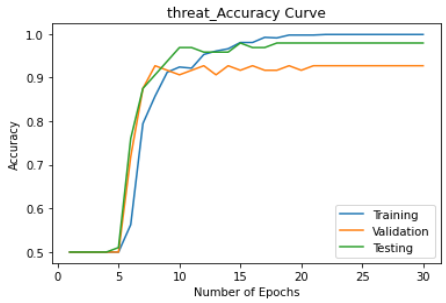
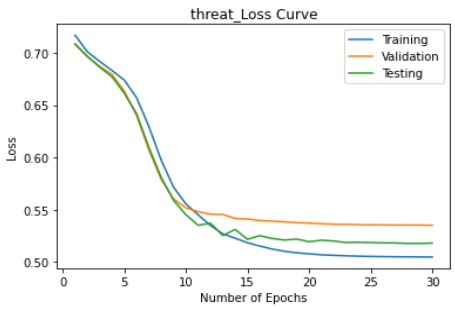
Our findings highlighted how crucial it is to preserve class balance when using binary classifiers. After using balanced data to train our models, we were able to significantly reduce the number of false positives (as seen in Table 5) that were our main worry. Following data augmentation, our model was able to generalise from the expanded training examples and moved in the direction of the reduced loss depicted in Table 6. Since augmentation has been shown to be successful, more advanced techniques—possibly through back-translation—might produce an even more varied and strong training set.

Table 5. Comparison of confusion matrices before and after class balancing

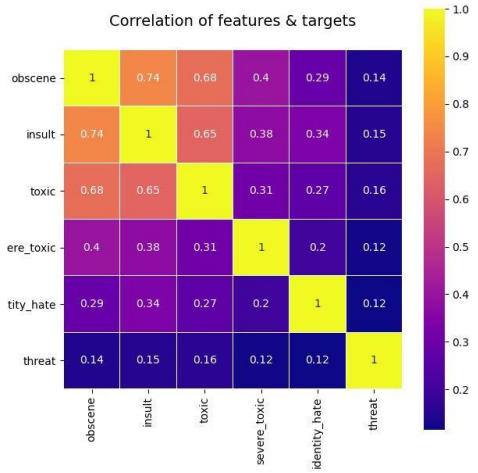
Reference		**Severe Toxic Confusion Matrix Before**		**Severe Toxic Confusion Matrix After**	
TP	FP	897	785	686	11
FN	TN	13	1350	19	694

Table 6: Loss and accuracy curves of “threat” class before and after data augmentation

Without data augmentation		

With data augmentation		

By analyzing the data, we discovered some classes had a strong association (for instance, 74% between obscene and insult). As a following step, we might look at implementing chained classifiers to take advantage of this trend and boost the accuracy of our model.



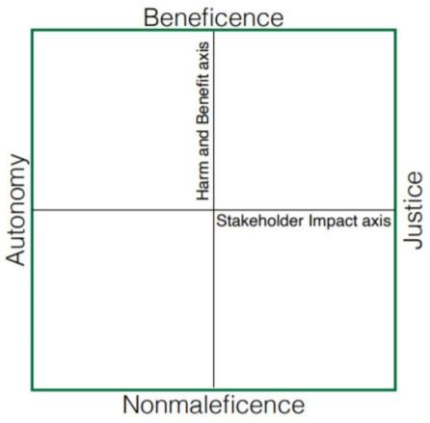


Figure 14: Plot of feature correlation **Figure 15:** An Imagined Plane of Ethical Reasoning

Conclusion :

Our project's guiding idea is to encourage nonmaleficence in online communities by spotting offensive comments and responding to them. Those who seek a secure and productive workplace free of detrimental distractions are more likely to experience this.

By restricting their right to free speech, those who publish harmful comments would have less autonomy, but this might also limit the range of viewpoints that are expressed in the forums. Long-term, this can have a negative impact on the "political correctness culture" [12]. Thus, it's critical to reduce the number of false positives our algorithm generates and to promote all productive dialogue.

Our model benefits platform hosts as well because it eliminates the need for human discussion moderation, saving time and money. Because all comments will be handled equally, using a machine learning model to filter comments promotes justice.

SCREEN-SHOT OF CODE:

0. Install Dependencies and Bring in Data

```
!pip install tensorflow tensorflow-gpu pandas matplotlib sklearn
```

```
import os
import pandas as pd
import tensorflow as tf
import numpy as np
```

```
df = pd.read_csv(os.path.join('jigsaw-toxic-comment-classification-challenge','train.csv', 'train.csv'))
```

```
df.head()
```

1. Preprocess

```
!pip list
```

```
from tensorflow.keras.layers import TextVectorization
```

```
X = df['comment_text']
y = df[df.columns[2:]].values
```

```
MAX_FEATURES = 200000 # number of words in the vocab
```

```
vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                               output_sequence_length=1800,
                               output_mode='int')
```

```
vectorizer.adapt(X.values)
```

```
vectorized_text = vectorizer(X.values)
```

```
#MCSHBAP - map, chache, shuffle, batch, prefetch  from_tensor_slices, list_file
dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
dataset = dataset.cache()
dataset = dataset.shuffle(160000)
dataset = dataset.batch(16)
dataset = dataset.prefetch(8) # helps bottlenecks
```

```
train = dataset.take(int(len(dataset)*.7))
val = dataset.skip(int(len(dataset)*.7)).take(int(len(dataset)*.2))
test = dataset.skip(int(len(dataset)*.9)).take(int(len(dataset)*.1))
```

2. Create Sequential Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Bidirectional, Dense, Embedding
```

```
model = Sequential()
# Create the embedding layer
model.add(Embedding(MAX_FEATURES+1, 32))
# Bidirectional LSTM Layer
model.add(Bidirectional(LSTM(32, activation='tanh')))
# Feature extractor Fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
# Final layer
model.add(Dense(6, activation='sigmoid'))
```

```
model.compile(loss='BinaryCrossentropy', optimizer='Adam')
```

```
model.summary()
```

```
history = model.fit(train, epochs=1, validation_data=val)
```

```
from matplotlib import pyplot as plt
```

```
plt.figure(figsize=(8,5))
pd.DataFrame(history.history).plot()
plt.show()
```

3. Make Predictions

```
input_text = vectorizer('You freaking suck! I am going to hit you.')
```

```
res = model.predict(input_text)
```

```
(res > 0.5).astype(int)
```

```
batch_X, batch_y = test.as_numpy_iterator().next()
```

```
(model.predict(batch_X) > 0.5).astype(int)
```

```
res.shape
```

4. Evaluate Model

```
from tensorflow.keras.metrics import Precision, Recall, CategoricalAccuracy
```

```
pre = Precision()
re = Recall()
acc = CategoricalAccuracy()
```

```
for batch in test.as_numpy_iterator():
    # Unpack the batch
    X_true, y_true = batch
    # Make a prediction
    yhat = model.predict(X_true)

    # Flatten the predictions
    y_true = y_true.flatten()
    yhat = yhat.flatten()

    pre.update_state(y_true, yhat)
    re.update_state(y_true, yhat)
    acc.update_state(y_true, yhat)
```

```
print(f'Precision: {pre.result().numpy()}, Recall:{re.result().numpy()}, Accuracy:{acc.result().numpy()}')
```

References:

[1] Macleans.ca. 2020. *Online Hate Speech In Canada Is Up 600 Percent. What Can Be Done? - Macleans.Ca*. [online] Available at: <https://www.macleans.ca/politics/online-hate-speech-in-canada-is-up-600-percent-what-can-be-done/> [Accessed 28 October 2020].

[2] R. Hatzipanagos, "Perspective | How online hate turns into real-life violence," *The Washington Post*, 30-Nov-2018. [Online]. Available: https://www.washingtonpost.com/nation/2018/11/30/how-online-hate-speech-is-fueling-real-life-violence /. [Accessed: 28-Oct-2020].

[3] "Court documents identify 13 injured in deadly van attack | CBC News," *CBCnews*, 25-Apr-2018. [Online]. Available: https://www.cbc.ca/news/canada/toronto/injured-van-attack-1.4633308. [Accessed: 28-Oct-2020].

[4] "Potentials of NLP: Techniques, Industry-Implementation and Global Market Outline," *Analytics Insight*, 26-Jan-2020. [Online]. Available: https://www.analyticsinsight.net/potentials-of-nlp-techniques-industry-implementation-and-global-market -outline/. [Accessed: 28-Oct-2020].

[5] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," dissertation, 2017.

[6] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLOS ONE*. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0221152. [Accessed: 28-Oct-2020].

[7] Z. Zhang and L. Luo, "Hate speech detection: A solved problem? The challenging case of long tail on Twitter," *Semantic Web*, vol. 10, no. 5, pp. 925–945, 2019.

[8] E. Thompson, "Facebook partners with Ontario university on 'global network' to counter rise in online hate | CBC News," *CBCnews*, 28-Jul-2020. [Online]. Available: https://www.cbc.ca/news/politics/facebook-hate-online-extremism-1.5664832. [Accessed: 28-Oct-2020].

[9]"Toxic Comment Classification Challenge," *Kaggle*, 2017. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview. [Accessed: 28-Oct-2020].