Toxic Comment Classification

Group: Gradient Descent

Art yang
arty@sfu.ca

Hoang Viet Truong
hvtruong@sfu.ca

Abstract

Nowadays there are numerous comments on social networks, forums. Some of the comments are toxic, abusive. Since it is unfeasible to manually moderate them, most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we will create a toxic comment classification using machine learning methods. We will extract data from a Kaggle competition of the relevant topic and design a machine learning LSTM (Hochreiter and Schmidhuber, 1997) multiple-classification model.

1 Problem Statement and Motivation

Online toxicity is a pervasive problems today. Due to the anonymity of being behind a computer screen, as well as the lack of direct consequences, internet users are much more likely to make toxic comments than they normally would in person.

These toxic comments discourage users from expressing themselves as well as participating and contributing to the conversations. Therefore, the goal of our project is to categorize toxic comments, in order to help with moderation and help promote a healthy online environment for communication.

We plan to create a multi-label classification LSTM (Hochreiter and Schmidhuber, 1997) model (6 classes) that computes the probability of a comment being toxic with ROC AUC score of above 0.8. In particular, given an English comment, our model will result in 6 probabilities indicating the comment has zero or more labels. Below are some sample inputs and expected outputs from our training dataset. The labels follow the order of the set {toxic, severe toxic, obscene, threat, insult, identity hate)}.

Input	Expected output
D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	[0, 0, 0, 0, 0, 0]
You, sir, are my hero. Any chance you remember what page that's on?	[0, 0, 0, 0, 0, 0]
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	[1, 1, 1, 0, 1, 0]
Hey... what is it.. @ — talk . What is it... an exclusive group of some WP TALIBANS...who are good at destroying, self-appointed purist who GANG UP any one who asks them questions abt their ANTI-SOCIAL and DESTRUCTIVE (non)-contribution at WP? Ask Sityush to clean up his behavior than issue me nonsensical warnings...	[1, 0, 0, 0, 0, 0]

Table 1: Some sample inputs and expected outputs

2 Related Work

This work was completed in a Kaggle competition "Toxic Comment Classification Challenge" (2017) (Kaggle, 2017) , with over 4500 participated teams. There were some good models designed for this Kaggle competition. For instance, the blog Toxic Comment Classification using LSTM (Hochreiter and Schmidhuber, 1997) and LSTM-CNN (Varudandi) implemented two models: traditional LSTM and LSTM-CNN with final ROC AUC scores of 0.977, 0.971 respectively. Those two models preprocessed data to obtain clean texts and applied **fastText's** pre-trained word embedding as Transfer Learning to the toxic comment classification task. Figure 1 shows the graph representation for the LSTM (Hochreiter and Schmidhuber, 1997) model's structure.

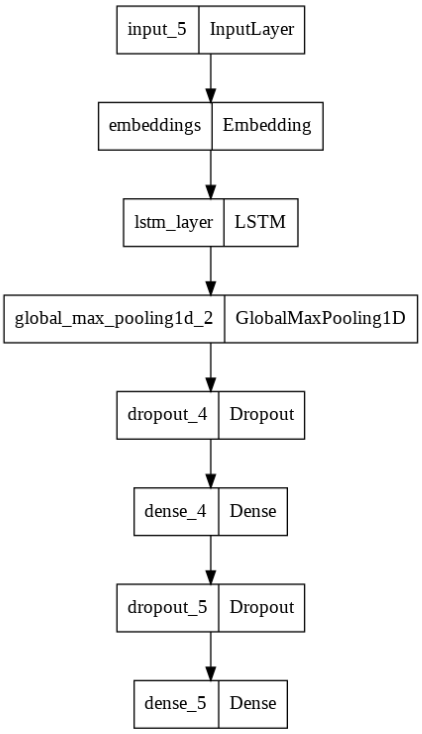


Figure 1: LSTM structure with 97.7% accuracy (Varudandi)

Apart from the LSTM model which we will focus on in our experiments, the BERT transformer model was also another popular choice amongst the highest scoring teams. A submission by Kaggle user Gtskylar uses a pre-trained BERT tokenizer and model from Tensorflow to achieve a score of 0.986 (Gtskyler, 2021).

For our work, we will be using Google Colab. We adapted the preprocessing process for the two LSTM models to clean up raw texts from training dataset. Then we will use the pre-trained GloVe vector `glove.6B.100d.retrofit.txt` that we generated from Programming Homework 2 to create our Embedding layer. From this step, we will create our own model using Keras (Chollet et al., 2015) and submit the final classifications to Kaggle for evaluation.

3 Approach

3.1 Preprocessing

We completed the preprocessing step by adapting and expanding the process of the works above. The preprocessing step consists of: converting all texts to lower cases, replacing abbreviations such as "what's" to "what is", "'ve" to "have", ..., removing repeated words in each sentence, punctuations, stopwords, numbers, from NLTK corpus (Bird et al., 2009). Below are some sample comments after preprocessing:

Raw text	Clean text
D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	aww match background colour seemingly stuck thanks talk january utc
You, sir, are my hero. Any chance you remember what page that's on?	sir hero chance remember page
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	cocksucker piss around work
Hey... what is it.. @ — talk . What is it... an exclusive group of some WP TALIBANS...who are good at destroying, self-appointed purist who GANG UP any one who asks them questions abt their ANTI-SOCIAL and DESTRUCTIVE (non)-contribution at WP? Ask Sityush to clean up his behavior than issue me nonsensical warnings...	hey talk exclusive group wp taliban good destroying self appointed purist gang one asks question abt anti social destructive non contribution wp ask sityush clean behavior issue nonsensical warning

Table 2: Some samples before and after preprocessing

3.2 Tokenizing

We applied `Keras Tokenizer` from `tensorflow.keras.preprocessing` to tokenize our comment texts with num_words = 20000 and pad the corresponding numerical sequences with maxlen = 100. For this step, first we fitted all the training texts to `Keras Tokenizer` and resulted in a dictionary in which each word has a unique integer value. Then, we obtained the integer sequences by replacing each word in texts with its corresponding value. Those sequences would represent the inputs of our model. Below are some samples of preprocessed comments, and their sequences:

Clean text	Integer sequence
ok whatever separate frankish province existed still believe included separate entry disambiguation page live current version page well talk	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 337 472 708 14019 1947 2098 76 131 375 708 348 1288 2 454 298 234 2 32 4]
put article let black supremacist talk page stop outnumbered non racist	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 113 1 105 542 5776 4 2 92 13510 185 958]
u delete robero de neroes spanish thats valid info	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 221 628 1075 1108 664 342]
ok sorry keep trying tnx reminder may noticed soon actually comprehend policy try follow still comment though copyright line issue educational purpose would appear cover context bow yr claim valid note original discussion inclusion line hamsun obituary hitler reason deletion copyright deleter want text claiming irrelevant entry copyright permission due course obtained monday hope support inclusion obituary entry incidentally approx many editor writing	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 337 172 125 153 4128 20 501 566 123 5571 73 174 548 76 37 148 141 290 81 3148 576 5 581 652 580 5388 5779 130 664 91 203 52 797 290 17542 8156 1558 71 38 141 18535 30 158 1169 881 348 141 1012 457 307 3009 4684 135 209 797 8156 348 3694 10944 58 36 360]

Table 3: Some sample texts to sequences

3.3 Embedding

This seems to us as the most crucial part of the whole architecture. Our baseline used the pre-trained GloVe vector `glove.6B.100d.retrofit.txt` generated from Homework 2 to get pre-trained weights for our embbeding layer. We adapted the method from the article on Keras Using pre-trained word embeddings (fchollet, 2020) to create a fixed embedding matrix, meaning that they will not be trainable. Figure 2 is the visualization of the pre-trained weights for our Embedding layer:

```
[[ 0.          0.          0.         ...  0.          0.
   0.        ]
 [-0.02328654  0.01661578  0.0347484  ... -0.08114112  0.03868156
   0.0362244 ]
 [-0.04174801  0.00954155  0.02114576 ... -0.04753175  0.03944544
   0.02908922]
 ...
 [ 0.0742953  -0.0319478  -0.0733457  ...  0.2241829  -0.0217099
  -0.038797  ]
 [ 0.          0.          0.         ...  0.          0.
   0.        ]
 [ 0.05068332  0.05030753  0.00888725 ...  0.02128703  0.00471647
   0.05786167]]
```

Figure 2: Pre-trained weights from GloVe vector

3.4 Long-Short Term Memory

For this task, we focused mainly on implementing a Long-Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). The LSTM model is one of the more simple yet most commonly used recurrent neural network models for sequential inputs like languages (Jurafsky and Martin, 2021).

For our implementation, we padded our input to standardize the input length, so the inputs will have dimension (`batch_size`, 100). For the output layer, we used `Sigmoid` activation function with binary cross entropy loss over each label. Finally, the Adam optimizer with default learning rate 10^{-3} is used to optimize the loss.

3.5 Drop-out

To prevent over-fitting, we added drop-out layers in training to achieve a regularizing effect for some of our models. Unlike other regularization techniques like L2 which relies on mathematical functions, we ignore 10% of the nodes randomly (using coin flips) in our weights during in each training step to reduce the over-dependencies on the weights of some nodes.

3.6 Model

For the baseline, we first created a simple structure including 4 layers: an Embedding layer, a LSTM layer with 100 hidden units, a Global Max Pooling layer and a fully connected layer (100, 6) to generate outputs of 6 classification labels. In this baseline, the Embedding layer is fixed with the matrix from pre-trained GloVe mentioned above. Therefore, the Embedding layer will have dimension size `(len(Tokenizer), 100)`. Figure 4 and Figure 5 represent the visualization of our baseline model:

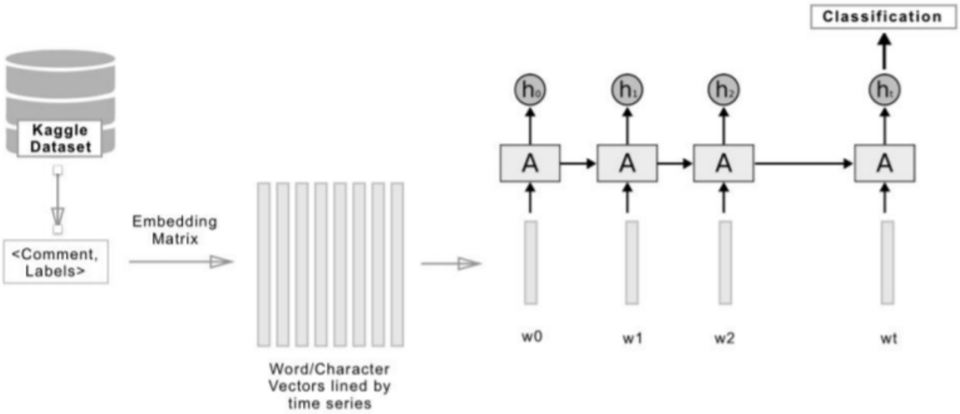


Figure 3: High level view of the classification model with LSTM (Khieu and Narwal)

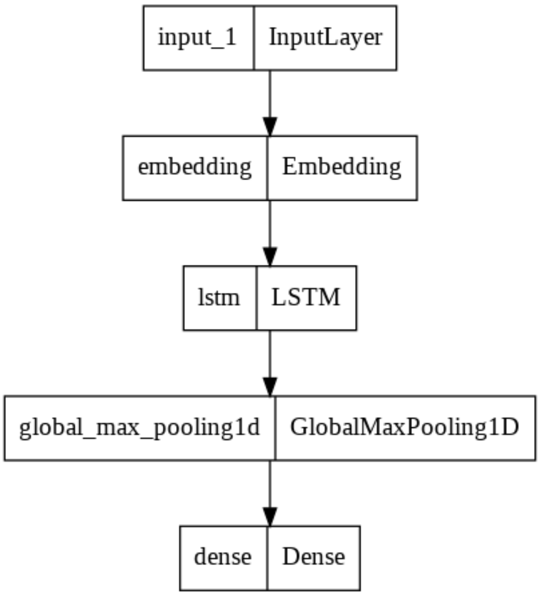
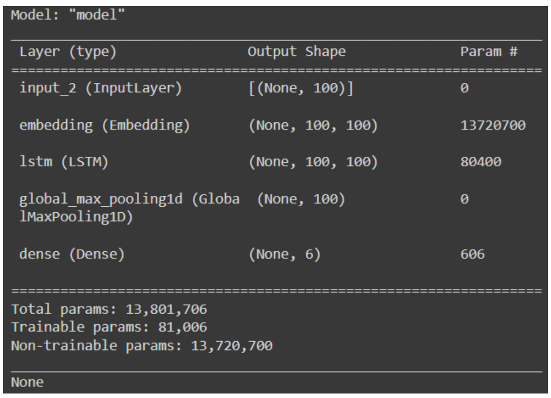


Figure 4: Baseline model


Model: "model"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 100)]	0
embedding (Embedding)	(None, 100, 100)	13720700
lstm (LSTM)	(None, 100, 100)	80400
global_max_pooling1d (GlobalMaxPooling1D)	(None, 100)	0
dense (Dense)	(None, 6)	606

Total params: 13,801,706
Trainable params: 81,006
Non-trainable params: 13,720,700

None

Figure 5: Baseline model summary

After several experiments of different architectures, our best model contains 6 layers: an Embedding layer, a LSTM layer with 100 hidden units, a Global Max Pooling layer, a fully connected layer (100, 50) followed by ReLU activation function, a dropout layer (0.1), and a fully connected layer (50, 6). In out best model, we did not use pre-trained GloVe so the dimension size of the Embedding layer is (20000, 100). Also, 10% of the nodes in the fully connect layer (100, 50) was removed in dropout to prevent over-fitting. Figure 6 and Figure 7 represent the visualization of our best model.

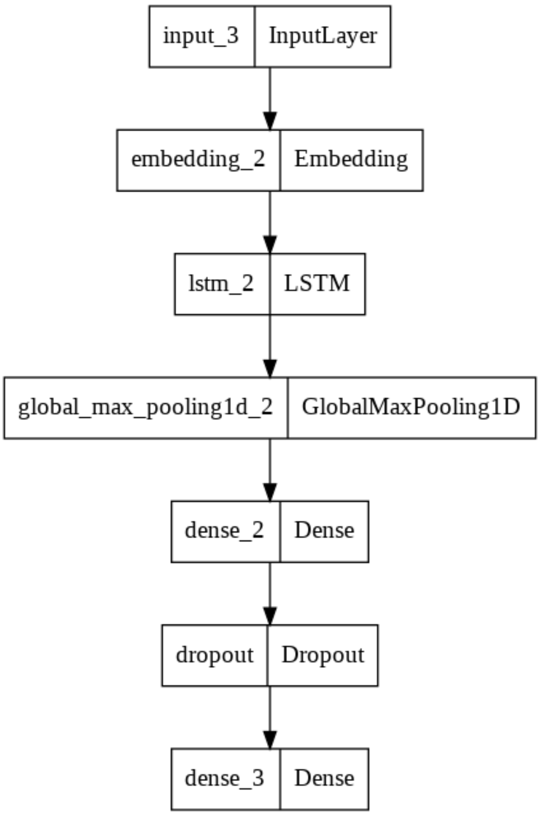
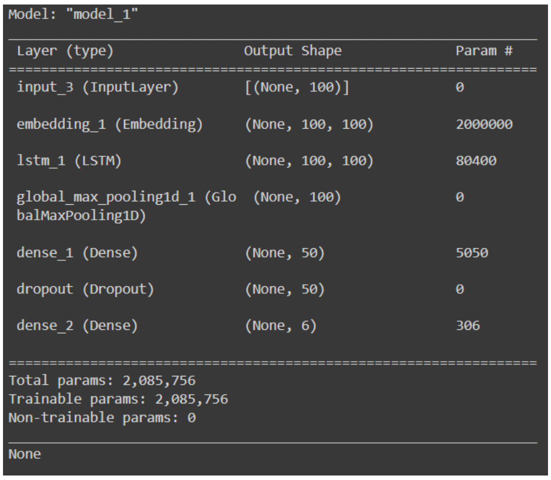


Figure 6: Best model



```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 100)]             0

embedding_1 (Embedding)      (None, 100, 100)          2000000

lstm_1 (LSTM)                (None, 100, 100)          80400

global_max_pooling1d_1 (Glo  (None, 100)               0
balMaxPooling1D)

dense_1 (Dense)              (None, 50)                5050

dropout (Dropout)            (None, 50)                0

dense_2 (Dense)              (None, 6)                 306

=================================================================
Total params: 2,085,756
Trainable params: 2,085,756
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 7: Best model summary

4 Experiments

4.1 Data

The model are trained and tested using data sets from the Kaggle competition mentioned above. This dataset includes a large number of human labelled comments from Wikipedia for toxic behaviours. The provided data contains 4 files – sample_submission.csv.zip, test.csv.zip, test_labels.csv.zip, and train.csv.zip (Kaggle, 2017).

The training data set, train.csv.zip, contains 159,571 entries, each entry contains the comment_text, and the labels of 6 classes: toxic, severe toxic, obscene, threat, insult, identity hate.

In the training set, about 90% of the comments have no marked labels, meaning that only 10% exhibited some type of toxicity (table 4).

Amongst the labels, "toxic" was marked most frequently, accounting for about half of the marked occurrences. Additionally, 58.2% of the toxic comments had between 2 to 4 marked labels, and usually contains "toxic" alongside other types of toxicity. Figure 8 provides a visualization of the distribution.

The average length of each comment is 394.074 words. For 159,571 entries, each with 6 labels, together is 957,426 labels, only 35,098 labels are marked as 1 and the rest are 0 (approximately 3.7% of all the labels are in 1 of the 6 classes).

Since training recurrent neural networks also requires validation data, we divided data from train.csv into training and validation datasets, with the ratio of 0.75 and 0.25.

Label	#	%
toxic	15294	9.6%
severe_toxic	1595	1.0%
obscene	8449	5.3%
threat	478	0.3%
insult	7877	4.9%
identity_hate	1405	0.9%
no label	143346	89.8%

Table 4: number and percentage of comments with a particular label

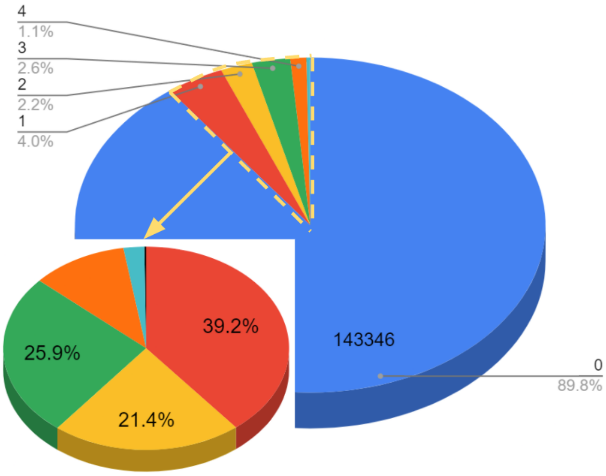


Figure 8: number of comments with n toxic labels

4.2 Baseline

We used the LSTM model with GloVe embedding as our baseline model. Since the pre-trained GloVe we used was retrofitted with semantic relations as in the paper Retrofitting Word Vectors to Semantic Lexicons (Faruqui et al., 2015), we want to study the effectiveness of retrofitted GloVe applied to this task. We hypothesized that the retrofitted GloVe can significantly improve the model. Under this hypothesis, we experimented with various architectures, with and without GloVe in our embedding layer.

4.3 Evaluation Metric

Since this work replicates a Kaggle challenge (Kaggle, 2017), we decided that the best way to evaluate the models is to generate csv submissions and submitted to Kaggle for evaluation. For this challenge, the score feedback is evaluated using a metric called the ROC AUC. Submissions are evaluated on the mean column-wise ROC AUC and the final score is the average of the individual AUCs of each predicted column.

ROC is the abbreviation for Receiver Characteristic Operator, it represents the curve plotted with True Positive Rate (TPR) against False Positive Rate (FPR).

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative} \quad (1)$$

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} \quad (2)$$

AUC stands for Area Under the Curve and is the gray area in figure 9. ROC AUC score ranges from 0 to 1 with 1 as perfect classification.

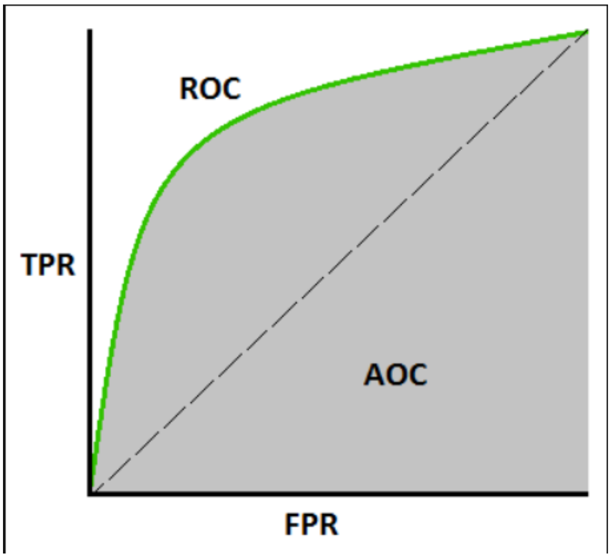


Figure 9: AUC - ROC Curve (Narkhede)

4.4 Results

The results of our experiments can be seen in tables 5 and 6. Table 5 contains the scores of models using pre-trained GloVe in the embedding layer while table 6 contains the scores of models without pre-trained GloVe.

Model	ROC AUC score from Kaggle (Private Score)
Baseline LSTM	0.94418
LSTM with a (100, 50) fully connected layer	0.95281
LSTM with a (100, 50) fully connected layer and one dropout layer 0.1	0.95008
LSTM with a (100, 50) fully connected layer between two dropout layers 0.1	0.95362

Table 5: Model performances with pre-trained GloVe embedding

Model	ROC AUC score from Kaggle (Private Score)
LSTM	0.97397
LSTM with a (100, 50) fully connected layer	0.96999
LSTM with a (100, 50) fully connected layer and one dropout layer 0.1 (Best model)	0.97621
LSTM with a (100, 50) fully connected layer between two dropout layers 0.1	0.96896

Table 6: Model performances without pre-trained GloVe embedding

4.5 Analysis

By comparing our results, we can see that modifying the basic LSTM model with more layers only marginally increased the AUC-ROC score if any improvements were made.

In the experiment with the pre-trained Glove embedding, the baseline LSTM model performed worst, and the addition of more complex layers resulted in a 1% score improvement. On the other hand, when trained without the pre-trained embedding, the baseline model scored second, with only a 0.2% difference from the best model. These results suggested that the complexity of the LSTM model does not necessarily correlate to better performance.

We suspect the reason for getting poorer result when using the retrofitted GloVe as the word embedding is due to the fact that the weights were adjusted specifically for lexical substitution in Assignment 2 (Chang, 2021). Since the retrofitted GloVe may have emphasized the weights of similar words, the model might mistake clean words for toxic comments if they are contextually similar to toxic phrases. The observations of drop-outs improving the predictions (table 5) could be an evidence for this speculation. By adding an extra dropout layer before the connected layer, we may reduce the skew for some higher weights in the retrofitted embedding that could lead to miss prediction. In contrast, without a pre-trained embedding, the model results in a reduced score with the addition of the drop-out layer before the connected layer (table 6).

Without using the pre-trained GloVe in our embedding layer, it seems that the scores increase 0.01 to 0.02. This seems to us as a significant success as all the scores have been over 0.9 and any small improvements is essential.

Among all the models we attempted, the best ROC AUC score we achieved from Kaggle is 0.9762. To further improve the accuracy, we may need to obtain better word embedding by retrofitting the GloVe to our toxicity task or obtain a more diversified set of embedding, as opposed to the retrofitted word embedding specialized for lexical substitution, which we used for the experiments in table 5.

Some qualitative comments and result probabilities produced by our best model can be seen in table 7.

5 Limitations

Our results are limited to using only LSTM in all the architectures. We acknowledged that there are more models we can attempt on like Bi-directional LSTM, LSTM-CNN, ... Therefore, we can only come up with the conclusion about using retrofitted pre-trained GloVe with LSTM. Moreover, we have not tried pre-trained 200d, 300d Glove and done much hyperparameters tuning so there are still spaces for improvements.

We only trained the model with low number of epochs due to the limitation on computing power and time. It is possible that the pre-trained GloVe may provide a better accuracy than the untrained embedding if more passes are made through our LSTM cell.

During training, the Google Collab notebook indicated that only about 30% of the computing resources were used. In order to complete more training epochs in a reasonable amount of time, we may need to implement a distributed training algorithm to maximize the cpu and gpu usage.

6 Conclusion

We achieved our best model using LSTM without retrofitted pre-trained Glove with ROC AUC score of 0.97621. We concluded that for this toxic comment classification task, retrofitted GloVe with LSTM we generated in CMPT413's Programming Assignment 2 (Chang, 2021) does not improve the performance of the models. However, it is also evident that the modifications in the word embedding layer have a more significant effect on achieving better predictions than making tweaks on the LSTM models itself.

An improvement in our approach for this task is not to limit the models to using LSTM only, we may obtain more findings with other architectures such as CNN and LSTM-CNN.

Our results demonstrated that LSTM architecture is a good candidate for language classification tasks. The model is able to achieve scores over 90% with the benefit of being relatively simple when compared to other neural network models.

In the future, we aim to achieve higher performance through more complex deep-learning models. We also would like to test with 200d, 300d retrofitted Glove to verify our conclusion today.

Work Contribution:

- Art Yang: Preprocessed and analyzed the data, designed data loader, wrote the report.
- Hoang Viet Truong: Generated pre-trained weights, designed baseline model and best model, wrote the report.

Comment text	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity hate
Yo bitch Ja Rule is more succesful then you'll ever be whats up with you and hating you sad mofuckas...i should bitch slap ur pethedic white faces and get you to kiss my ass you guys sicken me. Ja rule is about pride in da music man. dont diss that shit on him. and nothin is wrong bein like tupac he was a brother too...fuckin white boys get things right next time.,	0.99	0.475	0.95	0.1	0.856	0.2459
== From RfC == The title is fine as it is, IMO.	0.0019	4.12 $\times 10^{-7}$	0.00015	6.79 $\times 10^{-6}$	1.75 $\times 10^{-4}$	4.15 $\times 10^{-6}$
I'd never think I'd need to say it, but Wikipedia isn't a fansite discussion board. If anything is unannounced by any authority, it might as well be false. MMORPGs are overrated,	0.0048	6.34 $\times 10^{-7}$	2.28 $\times 10^{-4}$	1.13 $\times 10^{-5}$	3.26 $\times 10^{-4}$	8.84 $\times 10^{-6}$
DJ Robinson is gay as hell! he sucks his dick so much!!!!!	0.968	0.228	0.848	0.194	0.79	0.41

Table 7: Some sample inputs and produced outputs from test.csv

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Angel Chang. 2021. Sfu nlp class: Homework — lexical substitution.

François Chollet et al. 2015. Keras. https://keras.io.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

fchollet. 2020. Using pre-trained word embeddings.

Gtskyler. 2021. Toxic comments bert.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Dan Jurafsky and James H Martin. 2021. Speech and language processing (3rd ed. draft).

Kaggle. 2017. Toxic comment classification challenge. Accessed 2021-10-25.

Kevin Khieu and Neha Narwal. Cs224n: Detecting and classifying toxic comments.

Sarang Narkhede. Understanding auc - roc curve.

Shaunak Varudandi. Toxic comment classification using lstm and lstm-cnn.