Toxic/Abusive Comments Detection - CMPT825 Final Report

Martin Ambrozic, Matthew Canute, Adriena Wong
{mambrozi, mcanute, adrienaw}@sfu.ca

Abstract

Social media has always been a convenient and useful way to share content and ideas online, but interactions between users can often become toxic, with harassment, abusive comments, and hate speech. This may make it difficult for some users to express their opinions freely or may detract conversations. The goal of this project was the common task of building a toxicity classifier, but with the additional goal of determining whether having the context of the Twitter community surrounding each Tweet would have any meaningful impact on the model results. However, we did not find any meaningful difference between the Twitter communities we collected, and this paper will also investigate some of the reasons as to why we think this may be the case. We also investigated other measurable language differences between communities.

1 Introduction

While social media provides a means for users to express their thoughts freely and interact with friends or strangers online, platforms are constantly trying to censor out any inappropriate and toxic comments that violate their guidelines and policies. It would be important to distinguish these toxic comments away from regular comments using a model, especially for platform moderators. This model can also be used to analyse other trends related to toxic comments including toxicity pattern differences in certain online communities, and toxicity development within certain discussion topics. For example, we'd expect that a tweet such as "I hate you" could be considered rude on "academic Twitter", but if it's written as an ironic message among friends in a younger age group, then it's obviously not a message that requires any moderation. In particular, we were interested in analysing toxicity within tweets using a custom-built dataset from Twitter's streaming API.

In 2018, Kaggle released a toxicity classification competition on a dataset containing comments from Wikipedia's talk page [4]. We have used this dataset for model training, although the nature of Wikipedia comments seemed to differ slightly from the tweets we were trying to classify.

2 Motivation

Although there has been extensive work on tackling the problem of classifying levels of toxicity in a message, we have not found much work has been done on taking the context of the community into account for these classification models. We know for example that there are emerging different communities on Twitter with measurable differences in the way they use language [1].

While there has been a lot of analysis done on discovering Twitter communities through an unsupervised approach involving natural language processing (NLP) techniques, and defining Twitter communities through the network analysis of mutual follower graphs, we were unable to find any literature that attempted to combine the two approaches. Should we expect to see correlated unsupervised groups from both approaches?

3 Approach

We started by building various deep learning architectures, specifically a Convolution Neural Network, a Long-Short Term Memory, and a Gated Recurrent Units, on the well-known Kaggle competition for toxic comment classification (involving conversations on Wikipedia), with the goal of achieving a mean area under the curve (mean AUC) metric that would have earned us a top 200 rank on the leaderboard. We then re-used these network weights and adjusted embeddings to classify context-dependent Twitter comments as being toxic or not through transfer learning. The reader is encouraged to skip to section 4 if already familiar with these approaches.

3.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning algorithm that is usually used for image input. However, the CNN approach performs well for local phrase classification and when there is no strong dependence on previous sequences far from the current cell. A CNN is made up of an input layer, an output layer, and several hidden layers consisting of convolving filters.

3.2 Long-Short Term Memory

A Long-Short Term Memory (LSTM) is an artificial recurrent neural network (RNN) that processes long chains of data sequences and has feedback connections, making RNNs good for phrasal text input. A LSTM has three internal gates that control the flow of information, namely the input gate, output gate, and forget gate. The input gate controls the amount of information used from the new input. The output gate determines the how much of the cell's information is used for the next hidden state. The forget gate determines how much of the previous hidden state is considered in the calculation of the output. Refer to figure 1 for the structure of a LSTM cell.

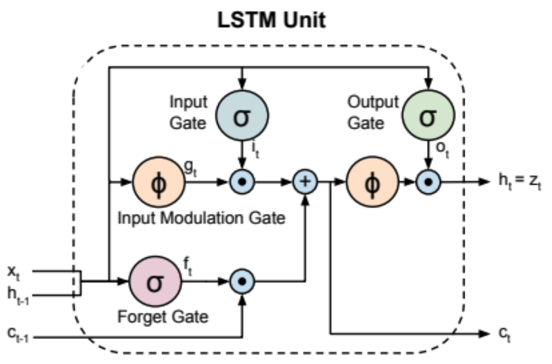

Figure 1: LSTM Network Cell [6]

The following are the equations for a forward pass of a LSTM cell:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}$$

3.3 Gated Recurrent Units

A Gated Recurrent Units (GRU) is also an artificial RNN and aims to address the vanishing gradient problem of standard RNNs. It is similar to LSTM in that it also has gated units to control information flow within each cell. GRU uses two gates: an update gate and a reset gate. The update gate determines how much of the information from previous time steps is used for the current unit. The reset gate decides how much of previous information to forget. Refer to figure 2 for the general structure of a GRU cell.

The following are the equations for a forward pass of a GRU cell:

$$\begin{aligned}
z_t &= \sigma(W_z[h_{t-1}, x_t]) \\
r_t &= \sigma(W_r[h_{t-1}, x_t]) \\
\tilde{h}_t &= tanh(W[r_t * h_{t-1}, x_t]) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}$$

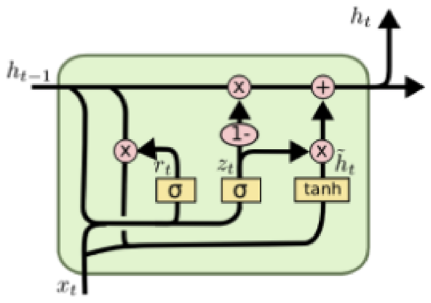

Figure 2: GRU Cell [8]

4 Data

4.1 Datasets

The dataset we used to train our model is from the Kaggle competition mentioned above. This training set includes 159,571 comments, each labelled as `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, and `identity_hate`. About 9.6% of these comments are considered toxic. This can be downloaded here: [9].

The dataset we are using to retrain our model is from our Twitter API collection process described below. It includes the columns `TweetID`, `TweetText`, `TweetDateTime`, `ScreenName`, `UserID`, `FriendsCount`, `FollowerCount`, `OverallTweetCount`, `OverallLikesCount` and `WasDeleted`. About 10% of these tweets were deleted.

4.2 Data Collection

We analyzed the n-grams of the Kaggle competition dataset to find highly toxic phrases, which were fed into the Twitter API's listener to scrape real-time tweets using Twitter's back-end to determine whether they matched any of those phrases. This was using Twitter's 1% API Streaming listener. Research has shown that this will be a representative sample of the entire Twitter dataset as well [12]. Random words from a dictionary were also fed into the listener's matching criteria in order to hopefully introduce some non-toxic tweets into the dataset.

This list of phrases can be found on our repo here:
`DataCollection/seed_phrases.txt`

After collecting a sample of roughly 400k tweets, we ran a second API call on each TweetID a few days later to check whether or not it still existed on Twitter. If not, it was probably deleted for abusive language, or was self-deleted by the user in retrospect. Additionally, each user's most recently 50 follows were collected to compute a Jaccard index similarity between each user, upon which we built

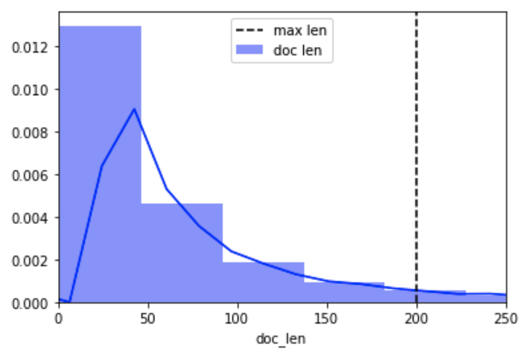

Figure 3: Wikipedia Comments Length

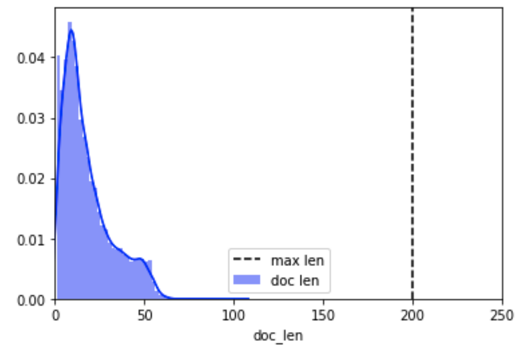

Figure 4: Twitter Tweets Length

agglomerative hierarchical clusters as further features.

4.3 Dataset Statistics

Differences between the Wikipedia dataset and our custom Twitter dataset: The vocabulary size of the Twitter dataset is 208,782 words while the Wikipedia dataset has 157,319 words after removing stop words and misspelled words. Additionally, as shown in figures 3 and 4, the tweet length from Twitter is much shorter in comparison to Wikipedia comments.

Exploratory Structured Data Analysis: With some initial exploration, it became clear that determining whether or not a tweet would be deleted was not a trivial task based only on the Twitter structured features.

For example, we compared the deletion rate of tweets with versus without emojis and we found that the difference was small with the tweet deletion rate with emojis being about 13.136% and the deletion rate without emojis being about 12.092%.

We also tried to find any meaningful difference in the number of tweets posted based on the time of day between deleted and non-deleted tweets. We found that both

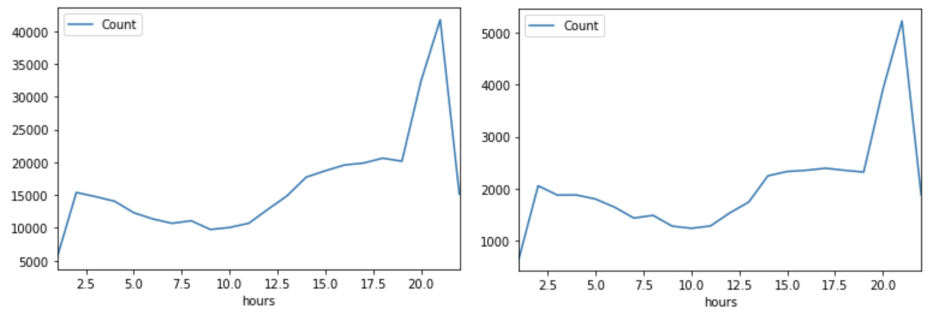

Figure 5: Count of Tweets By Time of Day; The left figure is showing not deleted tweets and the right figure is showing deleted tweets

groups had similar posting time patterns, with the number of tweets dramatically increasing at around 22:00. There was no significant difference in terms of deletion rate. Refer to figure 5 for a comparison. It was reassuring that there was no significant difference, as it implied that our retrospective deletion label approach was less likely to be an accidental a spam-bot detector, since the frequency of messages and the time of day can often be enough to successfully detect bots [13].

Additionally, by using Latent Dirichlet Allocation (LDA), we analysed the topic distributions for the Twitter dataset and compared them between deleted and non-deleted Tweets. Again, we found that the differences were not significant.

5 Code

All the open-source packages that we used for model-building and supplemental analysis are listed in the `requirements.txt` file in our code repository. We also started the initial model-training code by leveraging open notebooks shared within the Kaggle competition, and made modifications from these to more appropriately fit our custom dataset: [5], [7], and [2].

6 Experimental Set-Up

Evaluation Metrics: For the Kaggle competition models, we used the mean AUC metric. For the Twitter dataset, we used the top N% precision since the model results could most likely be used within the context of creating a priority queue of messages to be reviewed by a limited capacity of moderation resources.

Training Details: The Kaggle dataset was preprocessed through the following ways: removal of null comments, tokenization of the sentences into words, converting tokens to an index representation, and padding sequences to the same length.

We started with testing with a simple Naive Bayes model. This involved feeding the

Wikipedia data through a pipeline consisting of first creating a tf-idf matrix before inputting it into a Multinomial Naive Bayes model. However, due to its poorer performance in comparison to the other 3 models, we decided to focus on the other models' performances. It served as a good baseline.

The LSTM model used was based on [5]. This model includes an embedding layer, followed by a LSTM layer, which is then pooled using max pooling. Finally, the output is fed into two sets of dropout and dense layers.

The pooled GRU model was modified from [7]. This model involved an initial embedding layer, a spatial dropout layer, and a bidirectional GRU layer. The output was then fed into two pooling layers before finally a dense layer.

The CNN model used was based on [2]. This model is a sequential model with the following layers: embedding layer, convolutional layers, pooling layers, dropout layer, and finally two dense layers.

All of these models were trained using pretrained word embeddings, specifically the FastText [11] and the GloVe [10] embeddings.

7 Results

Model comparison on Wikipedia dataset: We trained models with LSTM, GRU, and CNN architecture using FastText and GloVe. To compare their performance, we split the Kaggle dataset into 75% training and 25% validation sets. Table 1 shows validation AUC scores for all six combinations. While results were close overall, the best scoring model was the combination of GRU and GloVe.

	GRU	**LSTM**	**CNN**
GloVe	0.9786	0.9648	0.9739
FastText	0.9770	0.9645	0.9734

Table 1: Comparison of AUC scores on the validation set using CNN, GRU, and LSTM neural network architecture and GloVe vs FastText pretrained word vectors.

Since the GRU architecture performed best, we adopted it for some further parameter exploration. Firstly, we compared performance while varying the number of internal nodes in each GRU unit. Table 2 contains the validation AUC scores. Results are close as long as there are more than 10 internal nodes and peaked at 80 nodes. Training times increased somewhat with more nodes added, but the differences were not too significant as shown in Table 2.

We also experimented with varying batch sizes during model training. We found a batch size of 32 to be optimal. Larger batch sizes offered faster training times but a decrease in AUC score, whereas a smaller batch size proved detrimental in both aspects. We found AUC scores generally peaked after 1 or 2 epochs of training. See Table 3 for details.

Nodes	**ROC-AUC**	**sec/epoch**
1	0.9526	415
5	0.9731	430
10	0.9752	435
80	0.9786	445
120	0.9781	450

Table 2: Comparison of AUC on validation set for varying number of nodes in the GRU layer of the neural network.

Epoch	**1**	**2**	**3**	**sec/epoch**
b = 16	0.9646	0.9641	0.9626	885
b = 32	0.9781	0.9786	0.9769	445
b = 64	0.9627	0.9602	0.9578	225
b = 128	0.9575	0.9594	0.9565	125

Table 3: Training times and AUC on the validation set for various batch sizes b.

Finally, we experimented with varying the amount of dropout in the model. The GRU based architecture in [7] used a 20% dropout layer after the embedding layer originally. We modified that in two ways: no dropout at all and a further 20% dropout layer after the pooling layer. We found having a dropout layer was not a major influence on the model performance - see Table 4 for details.

Epoch	**1**	**2**	**3**	**4**	**5**
AUC, no dropout	0.9772	0.9765	0.9736	0.9686	0.9657
AUC, 20%	0.9781	0.9786	0.9769	0.9725	0.9692
AUC, 30% and 20%	0.9783	0.9782	0.9769	0.974	0.9712

Table 4: Comparison of AUC scores on validation set with various dropout configurations: no dropout at all, a 20% dropout layer after embedding and a 30% dropout layer after embedding along with a 20% dropout layer after pooling.

Performance on Twitter dataset: Using the GRU-based model, we tested performance on the Twitter data, varying its training between Wikipedia data only, Twitter data only, and Wikipedia followed by Twitter. We looked at the top and bottom 500 tweets sorted by the model's toxicity prediction and compared deletion rates among the different training set-ups.

8 Analysis of Results

The results from Table 1 show that the results from the three models as well as the two types of pre-trained word embeddings were very similar. While GloVe did perform better than FastText in all three models, the difference was minuscule. Also,

Training	**Top 500 Deletion Rate**	**Bottom 500 Deletion Rate**
Wikipedia only	0.080	0.038
Wikipedia and Twitter	0.536	0.028
Twitter only	0.552	0.040

Table 5: Precision scores showing Performance of GRU model on Twitter dataset

the AUC score from the GRU model was slightly higher than the LSTM model. While this difference could be by chance, it may be due to the simplicity of the GRU model compared to the more complex LSTM model, which may have overfit the Wikipedia data, seemingly confirmed by the training/validation set disparities.

Additionally, from Table 5, we found that, looking at the top 500 deletion rate, training on only the Twitter dataset performed the best. However, qualitatively inspecting the different results showed that using the initial training on the Wikipedia toxicity dataset detected more of the deleted tweets that we were interested in, namely those that were probably deleted for being toxic/abusive rather than spam/pornographic. Focusing on the bottom 500 deletion rate, training on the combined Wikipedia and Twitter dataset performed the best. We think that the best model would use the combined dataset since looking only at the deleted Twitter comments would not be good for model training as tweets could be removed for reasons other than toxicity.

Taking the best models from the Wikipedia dataset, we then experimented with applying the context of the Twitter communities to the input as a new input vector appended onto the input layer, however this did not result in an improvement.

In our `project.ipynb` file, we have the cached model loaded up to calculate the precision score on the top N%

Analysis on vocabulary difference between Wikipedia and Twitter communities: We noticed immediately that our Twitter sampled dataset had a much larger vocabulary size between the Wikipedia dataset versus our Twitter dataset. This additional vocabulary size seemed to be coming from the trend on Twitter of either intentionally misspelling words for ironic / humourous purposes, or as a way to express the timing of how long vowels would get drawn out, such as ”aaaaahhhhh”, and the hundreds of other possible variations on those sounds. Contrast this to Wikipedia's social network, where at the time of the data collection, their discourse seemed to comprise of a much more formal way of communicating, similar to the habits of email correspondence. For this reason, we were hoping to find a network cluster that would have a low ”misspelling rate” as a proxy for determining a similar formal communication style to the Wikipedia dataset.

However, the misspelling rate was roughly 10% with no significant difference between different network clusters. This could be because, although there anecdotally seems to exist different communities on Twitter that are using their accounts in a more professional setting, such as the blue-check journalists, they were not well-represented in the noise of our API listener, especially given the vulgarity in many of our seed words. We could combat this by collecting more data, and down-weighting certain follows, in a similar fashion to the inverse document frequency weighting. For example, since there are some accounts with hundreds of millions of followers, such as Katy Perry, those accounts should have less of an influence on defining a network following cluster. This can be viewed in the project notebook.

Additionally, we analyzed the differences between the resulting word vectors from each community. The most dramatic word differences can be shown in the project notebook, but they were essentially common non-English words. It was surprising to see that despite seeding the Twitter dataset with English words, when we attempt to compare the linguistic characteristics between the mutual-following network clusters by inspecting the highest word vector distances, the main differences seem to be from them comprising of actual different languages.

9 Future Work

It would have been useful to do some more extensive parameter testing for all three models rather than just the GRU model, in order to obtain a more in depth comparison of the model performances. Additionally, we tried to test using BERT word embeddings in order to compare its performance with FastText and GloVe embeddings but were unable to produce results for this due to time constraints.

We can also evaluate our assumption that deleted tweets is due to tweet toxicity by asking the public, through Amazon Mechanical Turk, to identify whether certain deleted tweets are toxic or not in order to determine the percentage of toxicity in deleted tweets. This would improve the results as we can attribute a proportion of the deleted tweets to being not toxic.

Since we were constrained by the Twitter API limits, we were unable to build the full Twitter follower graph to detect communities as the context for our text classifier, and had to use sparse-sampling for each user to build similarity matrices for each cluster.

References

[1] Bryden, John et al. [Online], "Word usage mirrors community structure in the online social network Twitter", EPJ Data Science, 2013. Available: https://epjdatascience.springeropen.com/track/pdf/10.1140/epjds15. Accessed: Apr. 12, 2020.

[2] Smolyakov, Vadim [Online], "Keras CNN with FastText Embeddings", Kaggle, 2018. Available: https://www.kaggle.com/vsmolyakov/keras-cnn-with-fasttext-embeddings. Accessed: Apr. 1, 2020.

[3] LeCun, Yann et al. [Online], “Gradient-Based Learning Applied to Document Recognition“, Proc of the IEEE, 1998. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf. Accessed: Apr. 14, 2020.

[4] N/A [Online], “Toxic Comment Classification Challenge“, Kaggle, 2018. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview. Accessed: Mar. 1, 2020.

[5] Bongo [Online], “[For Beginners] Tackling Toxic Using Keras“, Kaggle, 2018. Available: https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras. Accessed: Mar. 25, 2020.

[6] Donahue, Jeff et al. [Online], “Long-term Recurrent Convolutional Networks for Visual Recognition and Description“, N/A, 2015. Available: https://arxiv.org/pdf/1411.4389.pdf. Accessed: Apr. 14, 2020.

[7] Demidov, Vladimir [Online], “Pooled GRU + FastText“, Kaggle, 2018. Available: https://www.kaggle.com/yekenot/pooled-gru-fasttext/code. Accessed: Mar. 25, 2020.

[8] Rathor, Saurabh [Online], “Simple RNN vs GRU vs LSTM:- Difference lies in More Flexible control“, Medium, 2018. Available: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57. Accessed: Apr. 14, 2020.

[9] N/A [Online], “Toxic Comment Classification Challenge“, Kaggle, 2018. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. Accessed: Mar. 10, 2020.

[10] Pennington, Jeffrey et al. [Online], “GloVe: Global Vectors for Word Representation“, Stanford https://nlp.stanford.edu/pubs/glove.pdf Available: https://nlp.stanford.edu/pubs/glove.pdf. Accessed: Mar. 21, 2020.

[11] Facebook [Online], “FastText“, Facebook Available: https://research.fb.com/downloads/fasttext/. Accessed: Mar. 23, 2020.

[12] Li, Quanzhi el al. [Online], “How Much Data Do You Need? Twitter Decahose Data Analysis“, Thomson Reuters Available: https://www.researchgate.net/publication/312550275_How_Much_Data_Do_You_Need_Twitter_Decahose_Data_Analysis. Accessed: Apr. 10, 2020.

[13] Van Der Walt, Estée el al. [Online], “Using Machine Learning to Detect Fake Identities: Bots vs Humans“, University of Pretoria Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8265147. Accessed: Apr. 12, 2020.