macech at SemEval-2021 Task 5: Toxic Spans Detection

Maggie Cech
SAS Institute
margaret.cech98@gmail.com

Abstract

Toxic language is often present in online forums, especially when politics and other polarizing topics arise, and can lead to people becoming discouraged from joining or continuing conversations. In this paper, I use data consisting of comments with the indices of toxic text labelled to train an RNN to determine which parts of the comments make them toxic, which could aid online moderators. I compare results using both the original dataset and an augmented set, as well as GRU versus LSTM RNN models.

1 Introduction

In this digital era we live in, almost everyone is communicating online. As of January 2021, Facebook, YouTube, and WhatsApp each have over 2 billion users, which means many differing viewpoints and perspectives being shared (Statista, 2021). With such a huge exchange of ideas, there is bound to be some toxicity within the comments. Aside from discouraging users to continue with or join conversations, toxic comments can also taint users' perceptions on news sites (Tenenboim et al., 2019). Thus it is important to moderate online conversations without fully censoring users.

While forums typically rely on human moderators, with such vast amounts of data coming in, it can be difficult for humans to keep up (Nobata et al., 2016). Advances in deep learning and machine learning is making text processing a viable option to replace, or at least assist, human moderators clean up comment sections (Consultants, 2019). Some methods rely on simply classifying whether a comment is toxic or not, but identifying what parts of the text are actually toxic can assist moderators and provide insight into what makes language toxic. The SemEval task 5 aims to evaluate systems that detect toxic spans wihtin text using datasets where spans within the comments are labelled as toxic, differing from previously released datasets where whole comments were labelled as toxic or non-toxic (Pavlopoulos et al., 2021).

This is inherently a natural language processing task, similar to text classification and sentiment analysis. This study focuses on training a recurrent neural network to determine the indices of a given string that represent the toxic portions of a comment. Recurrent neural networks are classically used for natural language and sequence labelling task, and one could view this task as a form of sequence labelling. The goal of sequence labelling is, given a sequence as input, assign a sequence of labels. Because recurrent neural networks (RNNs) are flexible in their use of context information and can recognize sequential patterns, they are an attractive and commonly used choice in sequence labelling (Graves, 2012). This paper approaches the task at hand with a sequence labelling methodology, applying an RNN and comparing the use of gated reccurent unit (GRU) and long-short term memory unit (LSTM) layers in the RNN.

2 Related Work

Aggression in text is complex, often clouded by sarcasm or including repeat words that cause a model to incorrectly identify words as toxic. A study by Vaidya, Mai, and Ning found that comments including identities, such as LGBTQ+, Black, Muslim, and/or Jewish identities, often resulted in false positives for toxic comments, so this was a bias we wanted to be aware of in our study (Vaidya et al., 2020).

Detecting toxic spans is not as common of a task as toxic comment detection or sentiment analysis. Many studies surrounding toxic comments have been completed largely in part due to the availability of a large corpora of data released by the

Wikimedia Foundation, as well as several Kaggle competitions hosted by Google Jigsaw. Other studies generally take a text classification approach similar to sentiment analysis, which as previously mentioned, is not exactly the task at hand (Nobata et al., 2017).

One of the comparisons made in this research is between GRU and LSTM recurrent models. Generally, LSTM units have issues with vanishing gradients when text sequences are too long, so GRUs are used instead. GRU controls the flow of information like the LSTM unit, but without the use of a memory unit (Chung et al., 2014). Previous research has shown GRU outperforms LSTM for all depths in speech recognition. The same study determined that bi-directional RNN models consistently outperform uni-directional models and found that models with 5 or more recurrent layers did not improve the results (Khandelwal et al., 2016). As a result, in our research, we compare GRU and LSTM models to find if GRU will consistently outperform LSTM in a sequence labelling task, as opposed to speech recognition, and use bi-directional RNN models with fewer recurrent layers, as they were not shown to have any benefit. This paper will also compare the results of augmented and non-augmented datasets to determine if the use of synonyms will improve the performance after training.

3 Methodology

3.1 Pre-Processing

Quite a bit of pre-processing was required to prepare this data for training. First, we label the comments using the given indices representing the toxic spans within the comments. A word labelled with a "/1" is toxic and with "/0" is not toxic. The comments are then run through a function that goes through the following text preprocessing steps:

1. Text to lowercase
2. Remove URLs
3. Remove numbers
4. Remove extra whitespace
5. Expand contractions ("it's" to "it is", "they're" to "they are", etc.)
6. Remove punctuation
7. Tokenize the strings - here one token is one word in the comment
8. Lemmatize the tokens
9. Remove stop words (list of stop words from nltk.corpus package)

After this text processing is completed, we use SAS DLPy[1] and SAS SWAT Python[2] packages to continue with the processing. From the SAS SWAT package, we are able to connect to a server where we can upload the cleaned text data into a CASTable (similar to a Pandas DataFrame). Then the text data needs to be converted to embeddings. To do this, we use GLOVE 100-dimension trained word vectors and apply the vectors to the CASTable containing the comments. Because text data from human sources is so varied and can often include unknown words or misspellings, we remove any comments that could not be converted to numeric embeddings and place them in a separate table. Instead of using these unknown embeddings in training and scoring using a neural network, we will make simple predictions based on if any common toxic words are present in the comments. This is not ideal but only about 2-5% of data ends up in this separate table and saves us the trouble of dealing with finding all of the words that could not be converted to embeddings.

Ideally we would not be removing observations from the dataset and further experimentation with different embedding sets could lead to better outcomes. This does affect the final results slightly as there is both less data being used in training and worse prediction accuracy from the comments not used in the main prediction set contributing to the final score, but because the percentage of data actually removed is so low, we decided it would not make a big enough difference to focus on. The comments removed from the main set are predicted using common toxic words gathered from the training set. Although misspellings are often a cause for the embeddings failing on a comment, the misspelled word may not have been part of the toxic span, so the comment could still be predicted using common toxic words and have a fairly acceptable accuracy.

After the comments are converted to embeddings, we find the maximum column count of a

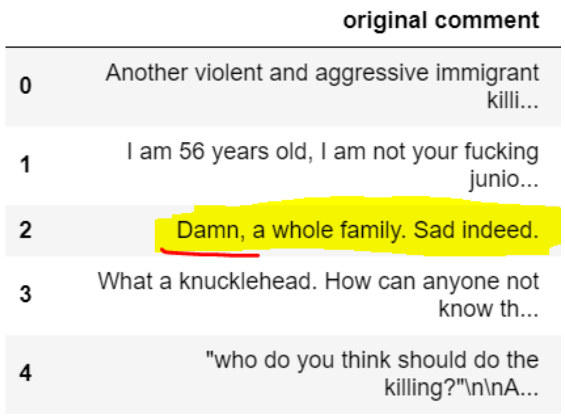
[1]https://sassoftware.github.io/python-dlpy/
[2]https://sassoftware.github.io/python-swat/

single embedding and pad all other embeddings to this maximum value. We then create output columns - one column for each 100-D word vector - and fill the columns with the toxicity labels from earlier. The final table used for training and scoring contains columns for the original comment, the labelled comment, the cleaned comment, 100 dimension embeddings for each word in the comment and the rest of the columns zero-padded to the longest value, and toxicity labels for each word in the comment, also zero-padded to the longest value.

We can then build an RNN using SAS' deepLearn actionset[3], train the model using the dlTrain action, and score using the dlScore action. When training, the embeddings columns are used as input columns to the model, where each token is 100 columns for each 100 dimension embedding, and the target columns are the label columns that contain either a 1 for a toxic word or a 0 for a non-toxic word, where each token is one column.

3.2 Augmented Data

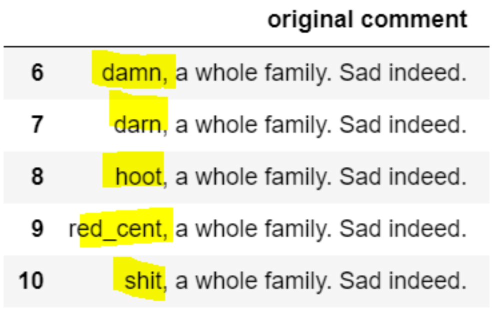
The goal of using augmented data was to increase the amount of data being used in training the models. To do this, we found every comment with only one toxic word and created up to five new comments with the toxic word replaced with a different synonym. We were able to do this by using wordnet from the nltk Python package that finds a list of synonyms for a given word. Figure 1 shows



	original comment
0	Another violent and aggressive immigrant killi...
1	I am 56 years old, I am not your fucking junio...
2	Damn, a whole family. Sad indeed.
3	What a knucklehead. How can anyone not know th...
4	"who do you think should do the killing?"\n\nA...

Figure 1: Original toxic comments

the original comments, with comment 2 being the example that is augmented, and figure 2 shows the new comments created using synonyms of the



	original comment
6	damn, a whole family. Sad indeed.
7	darn, a whole family. Sad indeed.
8	hoot, a whole family. Sad indeed.
9	red_cent, a whole family. Sad indeed.
10	shit, a whole family. Sad indeed.

Figure 2: Augmented toxic comments

word "damn". By augmenting only comments that contain a single toxic comment, we are able to increase the size of the data set from 7,939 comments to 21,822 comments - almost three times as many comments to be used in training. The idea here was to increase the size of the training data set to improve performance.

3.3 GRU vs LSTM

As mentioned in the introduction, we trained a bi-directional RNN model. After comparing performance, we found that a model with more than one bi-directional RNN layer was not improving the accuracy of the predictions, so we used a smaller model with only one bi-directional layer. The input layer connects to a fully connected layer, which goes into the bi-directional layer, into another fully connected layer, and finally to the output layer. Figure 3 shows the model architecture. We trained our data on two different models, one with a GRU cell used in the bi-directional layer and one that uses an LSTM cell in the bi-directional layer.

4 Experimental Results

In this section, we discuss the results from the two experiments - GRU vs LSTM models and augmented vs original data used in training. We use F1 scores to compare the two different models and types of data sets. The F1 scores, discussed later on, are a combination of both the predictions from the trained models and the comments that are predicted using common toxic words (which will henceforth be referenced as "guessed comments" for lack of better terms). The guessed comments do have a slight effect on the final results which will be described in this section

[3] *https : //go.documentation.sas.com/doc/en/pgmsascdc/9.4₃.3/casdlpg/cas− deeplearn − TblOfActions.htm*

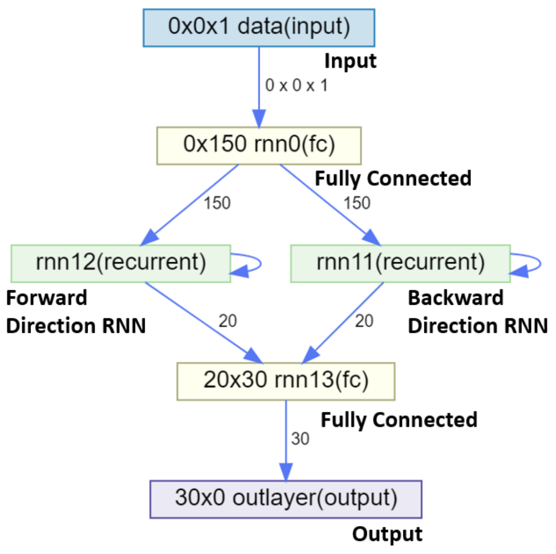


Figure 3: RNN model architecture showing layer dimensions

4.1 Evaluation Metric

Each model's performance was evaluated using an F1 score, as described in (Martino et al., 2019). If the system is represented by A_i, the return set from the system is $S^t_{A_i}$, t represents a post or comment, and G represents the ground truth annotations for post t, then F1 score of a system is defined as:

$$F^t_1\left(A_i, G\right) = \frac{2 \cdot P^t\left(A_i, G\right) \cdot R^t\left(A_i, G\right)}{P^t\left(A_i, G\right) + R^t\left(A_i, G\right)} \quad (1)$$

$$P^t\left(A_i, G\right) = \frac{|S^t_{A_i} \cap S^t_G|}{S^t_{A_i}} \quad (2)$$

$$R^t(A_i, G) = \frac{|S^t_{A_i} \cup S^t_G|}{S^t_{A_i}} \quad (3)$$

If $S^t_G = 0$, an instance where there are no toxic spans present, then $F^t_1(A_i, G) = 1$ if no toxic spans are predicted, and $F^t_1(A_i, G) = 0$ otherwise. To obtain the final F1 score for a particular system, the F1 scores are averaged over all posts t.

4.2 Augmented Data

Table 1 shows comparisons between the use of augmented and non-augmented data when training both the GRU and LSTM. We can see in the various hyperparameter settings, the augmented data actually performs worse than the original data. This is interesting considering generally the more data points used, the better the model. It is possible that the method in which we augmented the data, only changing one word within a comment, did not help the model learn any of the connections between the text but rather memorize more toxic words, many of which are often repeated.

Table 1 also shows that the learning rate does not affect the augmented data quite as drastically as it does the non-augmented data, with the non-augmented data showing much lower F1 scores when a lower learning rate is used, such as 0.001. It may be interesting to further explore if larger datasets are less affected by changes to the learning rate. The augmented dataset is almost three times larger than the original dataset, so it is also much more time consuming to train. With much longer training times and worse performance, the original dataset proves to be the more effective option. Table 3 shows that across both dev and test datasets, results from augmented dataset training were worse than their non-augmented counterparts.

We also compared results from only comments with a single toxic word present to find if these performed better for the augmented data, since only comments with a single toxic word were augmented. Again, the original dataset outperformed the augmented data. The differences between the single toxic word scores and scores when all comments are factored in are also very similar between augmented and non-augmented data results, showing the augmented data did not make much of a difference in predicting single toxic word comments. Table 2 shows these results.

4.3 GRU vs LSTM

After tuning the hyperparameters for both models, we found that the GRU model performs slightly better than LSTM across most hyperparameter settings. We were able to get the highest F1 score using GRU as well. Even between the scores using augmented data, GRU still performs slightly better than LSTM. Table 1 shows these results.

4.4 Results and analysis

Table 3 shows F1 scores using the best hyperparameter settings across the four methods - GRU, GRU trained with augmented data, LSTM, and LSTM trained with augmented data. The F1 test column shows scores for the evaluation data given, which was to be used for submitting a team's results. Our team submitted results for each of these methods in hopes of securing a better spot on the leaderboard, but ultimately ended up with a very low score, the

Max Epochs	Learn Rate	Mini-batch Size	GRU	GRU Augmented	LSTM	LSTM Augmented
10	0.1	5	0.515	0.504	0.514	0.501
10	0.05	5	0.514	0.503	0.499	0.495
10	0.001	5	0.287	0.406	0.324	0.393
10	0.1	10	0.473	0.459	0.463	0.469
50	0.1	5	0.518	0.478	0.512	0.460
20	0.1	5	0.519	0.494	0.504	0.464
20	0.05	5	0.369	0.433	0.396	0.513
20	0.001	5	0.521	0.500	0.505	0.501

Table 1: F1 scores during hyperparameter tuning for GRU and LSTM models with both original and augmented data

Max Epochs	Learn Rate	Mini-Batch Size	Single Toxic Original	All Toxic Original	Single Toxic Augmented	All Toxic Augmented
10	0.1	5	0.622	0.515	0.594	0.504
10	0.05	5	0.623	0.514	0.602	0.503
10	0.001	5	0.351	0.287	0.489	0.406
10	0.1	10	0.542	0.473	0.530	0.459
50	0.1	5	0.610	0.518	0.561	0.478
20	0.1	5	0.623	0.600	0.583	0.494
20	0.05	5	0.451	0.369	0.528	0.433
20	0.001	5	0.616	0.521	0.601	0.500

Table 2: F1 scores during hyperparameter tuning comparing performance for only single toxic comments and all toxic comments using the GRU model

Method	**F1 Dev**	**F1 Test**
GRU	0.519	0.602
GRU Augmented	0.504	0.551
LSTM	0.514	0.600
LSTM Augmented	0.501	0.550

Table 3: Best F1 scores for dev and test sets

reasoning for which is described in the note below. We were able to later calculate the F1 scores for the evaluation data when the entire dataset was released, after the competition ended, and those are the scores shown in Table 3.

It is interesting to note that the test data outperforms the dev data for every method used. This may be due to some kind of overfitting occurring. This is peculiar but is also likely due to the evaluation data containing fewer guessed comments, or comments predicted using common toxic words instead of predicted by the model. Because both the predicted comments and the guessed comments are used in calculating the F1 score, the portion of the entire dataset that is guessed would have an impact on the final score. The dev set had 2.5% of guessed comments and the test set had slightly fewer, with 2% of the entire dataset being comprised of guessed comments, which would explain the better results shown.

One can also see in Table 3 that the GRU model outperforms the LSTM model, even if just slightly, for both the original and augmented datasets and across both dev and test data. Other research has noted that GRU cells may be better for specificity, or finding true negatives, and focusing on less prevalent content, whereas LSTM cells are better for detecting true positives and focusing on highly prevalent content. (Gruber and Jockisch, 2020). Looking into the dev dataset, only 6% of the observations contain no toxic spans, but this model is not predicting whether an entire comment contains any toxic spans, it is predicting if each word is toxic. Out of all of the text, 93% of the words are non-toxic words, or in this case, words to be labelled negative for toxicity. A model that can better predict true negative outcomes would have the best performance with this data.

4.5 Note on Official Results Discrepancy

The official results in task 5 for the SemEval conference show this team to have an F1 score of 0.070. This score is much lower than the values we are presenting here because after adding the ground-truth values for the evaluation data it was found that the best F1 score we achieved was 0.602, as shown in Table 3, and the results were submitted to the competition out of order or formatted incorrectly, producing a very low F1 score.

5 Conclusion

We experimented with several different RNN models and ultimately utilized the results from bi-directional GRU and LSTM models. It was found that the GRU model slightly outperforms the LSTM model for this test case. As was found by Gruber and Jockisch, GRU cells can be better for detecting true negatives and since 93% of the words in the training dataset we used were non-toxic words, it follows that GRU cells may outperform LSTM cells in cases where there are more negative instances than postive in the datasets. It would be interesting to explore if using both a GRU cell and an LSTM cell in a model would further increase performance, with GRU focusing on less prevalent content and true negatives, and LSTM focusing on high prevalent content and true positives.

In the exploration of the use of augmented data, we found that not only did the models trained on augmented data perform worse than those trained on the original dataset, but they were also much slower to train. We compared if the F1 scores for observations with a single toxic word were higher for the models trained with augmented data, since only comments with a single toxic word present were augmented, but still the original datasets outperformed the augmented datasets. Because the augmented data does not improve results, we can continue to use the original dataset to cut down on computation time. We also found that the F1 scores resulting from the augmented data training did not react as greatly to changes in the learning rates as the original datasets did, which may be interesting to explore further.

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555v1.

Cambridge Consultants. 2019. Use of ai in online content moderation.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Publishing.

Nicole Gruber and Alfred Jockisch. 2020. Are gru cells more specific and lstm cells more sensitive in motive classification of text? *Frontiers in Artificial Intelligence*, 3(40).

Shubham Khandelwal, Benjamin Lecouteux, and Laurent Besacier. 2016. Comparing gru and lstm for automatic speech recognition.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cede no, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Join Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Changg. 2016. Abusive language detection in online user contents. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Changg. 2017. Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

John Pavlopoulos, Leo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5:toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

Statista. 2021. Global social networks ranked by number of users 2021.

Ori Tenenboim, Gina M. Masullo, and Shuning Lu. 2019. Attacks in the comment sections: what it means for news sites. Technical report, The University of Texas at Austin Center for Media Engagement.

Ameya Vaidya, Feng Mai, and Yue Ning. 2020. Empirical analysis of multi-task learning for reducing identity bias in toxic comment detection. arXiv:1909.09758.