Neural Content Moderation

Project Report

Isabella Garcia-Camargo
bellagc@stanford.edu

Guy Wuollet
gwuollet@stanford.edu

Martin Amethier
amethier@stanford.edu

Abstract

This project attempts to address the problem of online content moderation, as communities struggle to remove improper content, and avoid harming human moderators. The project aims to develop a series of models that detect different types of inappropriate content in online community comments. The project incorporates data collected from Wikipedia and Reddit comments, and uses several deep learning techniques to classify these comments. Measured by classification accuracy, the project was able to achieve a high degree of success. The project also found that the layered nature of deep learning model architectures was able to improve on the classification of simpler models. The project suggests that there is a future for automated content moderation, attempting to solve a major problem for contemporary online platforms.

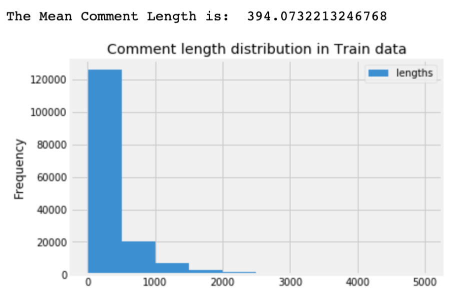
1 Introduction

Our project is attempting to solve the growing issue of content moderation on social networks, by developing a model that would be able to automate this process based on previously moderated content. As the use of social networks, digital news media, and other forms of online content continues to become more widespread, many platforms are currently grappling with the issue of quickly identifying and removing content that is inappropriate, not relevant to the platform itself, or fails to meet the guidelines of the community. In many situations, the platforms are struggling to perform these tasks quickly enough before the damage has been done. Furthermore, as some content can be highly graphic and disturbing, the harm caused to humans who are employed as moderators is significant[1], and should be a real concern for these platforms. An algorithm that can accurately classify content by how inappropriate it is, and by the specific nature of the impropriety, would therefore be extremely valuable.

2 Data

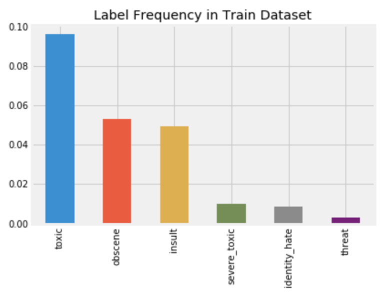
2.1 Collection

For data, we leveraged the dataset from the Kaggle competition *Toxic Comment Classification Challenge* for identifying toxic Wikipedia comments, then also investigated a larger Reddit data set which we have compiled. The Kaggle dataset's comments have been rated by humans for different forms of toxic behaviour: "toxic", "severely toxic", "obscene", "threat", "insult", or "identity hate". The aim is to correctly classify comments into these types of toxicity, and it should be noted that one comment can be classed as several different forms of toxicity. Below, a couple of graphics give a sense of the contents of the Kaggle dataset today:



```
The Mean Comment Length is:  394.0732213246768
```

(a) Figure 1: Wikipedia Comment Length Distribution



(b) Figure 2: Wikipedia Toxicity Label Frequency

We have also aggregated all Reddit posts and comments from inception of Reddit through August of this year, made possible because of the files stored at files.pushshift.io. Pushshift is a community on Reddit that stores a copy of all Reddit posts and comments ever for the purposes of computational social science research. We originally planned to take advantage of r/undelete and r/longtail, two subreddits that keep track of moderated posts from the top 1000 subreddits by popularity. Unfortunately, once we explored this data further, we realized the majority of posts were either external links or images. Less than 5% of posts were purely text. We also had trouble because there is no good way to track moderated comments from the historical Pushshift database.

Fortunately, the paper 'Hybrid Approaches to Detect Comments Violating Macro Norms on Reddit' open sourced a dataset of 2 million moderated comments from 2016-2017[2]. Their data attempted to segment moderated comments into different categories, primarily using clustering techniques. Since we already had all unmoderated Reddit comments ever posted, we sampled 2M unmoderated comments from the same list of subreddits. While we could have included more unmoderated comments, we struggled to handle a large volume of data when training models. We chose to balance the moderated and unmoderated class to both solve class imbalance and make training our models financially feasible. It's important to note that the Reddit data is only labeled moderated/unmoderated. It is not segmented into several reasons for moderated like the Kaggle data.

2.2 Pre-processing

Our baseline ran simply on the imported comment text values from the dataset. To improve our model's performance, created a preprocessing script with the following transformations: to lowercase, expanded contractions (don't do not), nonalphabetic character removal, and stopword removal (based on NLTK's dictionary of english stopwords. From here, used the WordNetLemmatizer, which considers the context of a term and and converts it to its meaningful base form.

As our final step, we pad all sequences to a length of 150 terms, which was the median length of comments in this dataset. In order to input this data to the sequential models we have built, all of our inputs must be of the same length. Shorter comments were padded with 'None' values, while longer comments were truncated.

3 Model Architectures

3.1 Baselines

We implemented the most upvoted public notebook on the Kaggle competition[3], which used NBSVM (Naive Bayes - Support Vector Machine), as a baseline, viewing this as a benchmark to then try to improve upon as we implement our model. We then implemented a second baseline. It featurized using TF-IDF and then fed the featurized data into logistic regression with k fold cross validation. We used 5 folds and L2 regularization.

3.2 TF-IDF NN

To build upon that the first model we implemented a deep neural network using the TF-IDF embeddings since that baseline performed best. The model had 3 hidden layers with 128, 64, and 32 units respectively. It used a tanh activation function for all layers except the last which used a sigmoid activation function. It also used an adam optimizer and binary crossentropy loss. We split the data 90-10 between train and test and then used 20% of the training data for validation. Because the embedding vectors and thus the TF-IDF vocabulary were about 159,000 cells long, we only kept the top 10000 must common terms in the vocabulary.

This model attempted to serve as a more advanced baseline, comparing a non-recurrent neural network against the LSTM and GRU models. As such, this model did not preprocess the data in the same fashion as the RNN models and was faster to train. Because of its speed and peak training accuracy, we decided to train this model with the Reddit data to see whether comments are more or less difficult to moderate on different sites. Further work on cross community moderation is important and discussed in the Further Work section.

3.3 LSTM

3.3.1 One Layer LSTM Model

An LSTM is an augmentation of an RNN to solve sequence prediction problems. LSTM models often perform well in sentiment classification problems as they take are able to model the long term relationship between different terms.

In our preprocessing, we took care of ensuring all the input data would be of the same length, 150 terms. This input layer is fed to a WordNet Embedding layer which will represent each term by its word embedding vector of dimensions [100 x 1]. The embedding layer is then fed to the LSTM layer, which is then fed to a 6 logit Dense layer, to represent the 6 independent labels of the classification. We trained the model with binary crossentropy loss and sigmoid activation.

3.3.2 Two and Three Layer LSTM Models

After creating our initial LSTM Model, we created Two and Three layer LSTM variation models. These models simply had two or three hidden LSTM layers which fed into eachother instead of just one layer. The hiearchy introduced due to several layers could improve the performance, as each layer could then process some part of the task we wish to solve, allowing for more nuance in the network. In terms of our comment classification, this could result in one layer taking care of identifying explicit terms while another layer identified threats.

3.3.3 BiDirectional LSTM

Our final LSTM model was a Bidirectional LSTM (BLSTM). A BLSTM works as an LSTM, but connects two hidden layers of opposite directions to the same output. While an LSTM can only compute the relationship between earlier terms to the later terms, a BLSTM can compute both the forward and backward relationship between terms.

3.4 GRU and Two Layer GRU Models

The key difference between a GRU and an LSTM is the number of gates used to compute the relationship between words: while a GRU has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output and forget gates). This makes the LSTM particularily good at longer sequences, but certain studies have found increased effectiveness for GRU models on shorter sequences. The GRU is also more efficient due to the less complex structure.

Thus, since we have on average shorter sequences, we included both a GRU and a two layer GRU model in our training. The benefits of the two layer GRU is parallel to the benefits which a two layer LSTM may provide.

3.5 HyperParameters

We trained all models for 3 epochs with a batch size of 20 and an adam optimizer. We conducted a grid search to identify possible parameter optimizations per model, and while there were very slight variations, this combination seemed to produce the best accuracy most often.

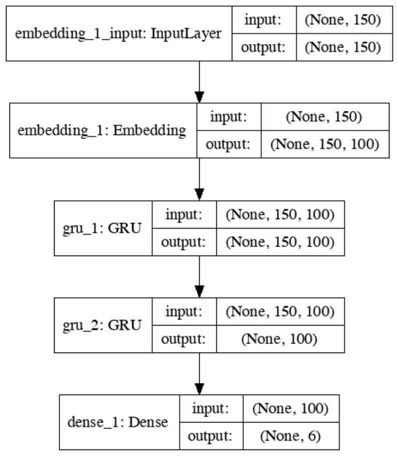
We also conducted test variations on length of input sequence, however, we found that the 150 length input sequence which did reflect the median performed the best.

4 Results

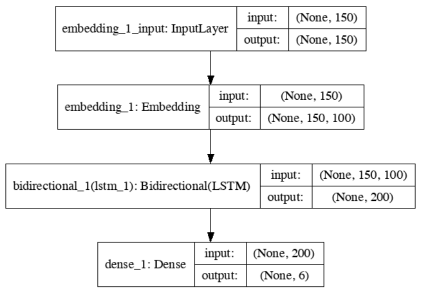
4.1 Best Test Performance: Two Layer GRU

Of all of our models, we achieved the best test performance on the two layer GRU. This performance is reflective of what we knew about our models going in: the simpler GRU seemed to perform better with the shorter sequences, and the complexity from the two layers allowed the model to increase its ability to identify the more difficult labels such as *threat*. The TF-IDF NN did very well in the training accuracy, but it vastly over fit to the data with the highest delta between test and train accuracy values. This is to be expected from the more brittle model, which had over 2M parameters to train.

Overall, all models were able to surpass our best baseline model of *TF-IDF logistic*. This can be attributed both to the preprocessing as well as the complexity we were able to achieve by the sequential models, which parsed the relationship between words which a simple logicstic model could not decipher.



(a) Figure 3: Two-Layer GRU Model Architecture



(b) Figure 4: Bidirectional LSTM Model Architecture

4.2 Accuracy Results Table

Table 1: Wikipedia Comment Results

	Train Accuracy	Test Accuracy	Δ in Accuracy
MODELS			
LSTM	0.9854	0.9810	.0044
Two Layer LSTM	0.9855	0.9807	.0048
Three Layer LSTM	0.9853	0.9795	.0058
GRU	0.9883	0.9814	.0069
Two Layer GRU	0.9869	**0.9816**	.0053
TF-IDF NN	**0.9994**	0.9743	**.0251**
Bidirectional LSTM	0.9864	0.9810	.0054
BASELINES			
TF-IDF Logistic	0.9577	0.9543	.0034
Naive Bayes SVM	0.8899	0.8745	.0154

Table 2: Reddit Comment Results

	Train Accuracy	Test Accuracy	Δ in Accuracy
TF-IDF NN	0.7631	0.6758	0.873

5 Further Work

5.1 Data Augmentation

For the Kaggle dataset, we decided to use several data augmentation strategies to allow the deep learning models to train more effectively. The idea was to preserve the overall meaning and general sentiment of the comments, while expanding the size of the dataset. We leveraged the Snorkel system[4] to implement several transformation functions that allowed us to expand beyond the original text comments. The functions included replacing verbs, nouns, and adjectives with synonyms loaded from the nltk corpus, and then also swapping any random two adjectives in a sentence. We spent a significant amount of time implementing the augmentation, and were almost able to successfully integrate it with our models, but it finally failed due to an ssl certificate failure, and so we were never able to make the individual packet work. Looking forward, this would be useful to allow our models to train more effectively.

5.2 Community-specific moderation

In investigating our project, we were also inspired by the SocialSent[5] paper William Hamilton wrote. In this paper, researchers trained domain-specific sentiment lexicons using Reddit data. For example, the meaning of the word soft is very different in a football subreddit and a fashion subreddit. The benefit to using this data is the structure specific to Reddit with sub-communities that have notably different guidelines and interpretations of unacceptable content. Although the moderation of all forms of content is important, we were especially interested in investigating and automating the different forms of moderation required in different communities. Due to the required size of the dataset, we weren't able to leverage the Reddit data that we had for this goal in this project, however this would be a fascinating next step for the project.

6 Contributions

Martin built the data augementation scripts with Snorkel, created model visualizations, helped build the RNN variant models, and worked on the poster and paper. Isabella built the Naive Bayes model, built the LSTM model and helped with variants, built the preprocessing script, and worked on the poster and paper. Guy built and ran the AWS infrastructure, data loaders, created the Reddit data set, built the TF-IDF models, helped build the RNN variant models and worked on the poster and paper.

7 Code

Project Repo HyperLink

Or plaintext link: https://github.com/guywuolletjr/reddit_content_mod

References

[1] https://www.theverge.com/2019/2/25/18229714/cognizant-facebook-content-moderator-interviews-trauma-working-conditions-arizona

[2] Chandrasekharan, Eshwar, Samory, Mattia, Gilbert, Eric. (2019). Hybrid Approaches to Detect Comments Violating Macro Norms on Reddit (Version 2.0) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.3338698

[3] https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline

[4] https://www.snorkel.org/

[5] Hamilton, William L., et al. (2016) Inducing domain-specific sentiment lexicons from unlabeled corpora. *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Vol. 2016. NIH Public Access.