Abusive Language Detection

1 Introduction

Anytime we go online, whether on social media or message forums, there is always a high risk that we may be the target of ridicule and even harassment. sentences such as go kill yourself are so common online and can have a very important impact on the user experience. As number of people communicating online is increasing, the need for high quality abusive language classifiers, to catch offensive language and thus remove the comment, becomes essential.

There are so many studies that have been done in this field. Most of them only focus on the targeted message to detect abusive language. For instance (Dinakar et al., 2011)(2) applies this principle to detect cyberbulling, and (Chen et al., 2012)(1) for offensive language. (Pavlopoulos et al., 2017; Mishra et al., 2018)(4)(3), works on training Recursive Neural Network operating on word embeddings and character n-gram features.

The work done under this project employs a neural network solution composed of bidirectional Long-Short-Term-Memory (LSTM) based classifiers to predict Non-aggressive (NAG), Overtly aggressive (OAG), and Covertly aggressive (CAG) labels for the test set.

The main Tasks I followed to achieve the goal of this assignment are: : 1)**Data pre-processor** to transform raw data into a format suitable for input to the model, 2)**Token encoder** to represent each token with a vector of real numbers, 3)**RNN classifier** to assign a label to each input sequence that shows whether it is offensive or not, 4)**Training function** that performs forward and backward propagation, and 5)**Evaluation** function that evaluates the performance of the model in every training epoch.

2 Background

2.1 Difficulties

Abusive language Detection is often more Complicated than one expects for some reasons. the noisiness of the data along with a need for world knowledge not only makes this a inquisitive task to automate but also a hard task for people as well.

Obfuscation of words. The intentional obfuscation of words and phrases makes detection difficult.Obfuscations such as ni99er make it impossible for simple model to be successful.

Abusive language may actually be very fluent and grammatical. Many samples of abusive language on the internet are very noisy that can be very helpful signal for an automated method, there are actually plenty of cases as well where abusive language is very fluent and grammatical.

Sarcasm. There are also some cases where users would post sarcastic comments in the same voice as the people that produce abusive language. It is hard for both human and machines to distinguish between sarcasm and abusive language

2.2 Data

Data for this project comes from Facebook. The format of the data is the same across training, development, and test sets . Each sample includes the document id, comment, and label. There are three different classes in this data:

- Non-aggressive (NAG): There is no aggression in the text. Example: no permanent foes, no permanent friends. interest is permanent!
- Overtly aggressive (OAG): The text is containing either aggressive lexical items or certain syntactic structures. Example: You can not escape from the sin of promoting this useless fellow.

- Covertly aggressive (CAG): The text is containing an indirect attack against that is not explicit, i.e., not using swear words or other direct forms of attack. Example: Absolutely! the deeper you dive the shallower cushion you have.

Table 1 provides information about the data in terms of the number of posts and tokens for training, development and test data sets. And table 2 provides statistics of the labels for both train and development sets.

3 Methodology

In this section, I give some intuition on how to make decisions like finding the right parameters while building a model. I divide this section in following parts: data preprocessing, token encoding, RNN classifier, training model, and Evaluation .

3.1 Data preprocessing

The first step in any natural language processing is to convert the input into a machine-readable one. Our preprocessing efforts are outlined below.

- Text tokenization: tokenizing our samples into pure lowercase English words to avoid capitalized versions of same words being treated as different words By using Natural Language Toolkit.

- Building vocabulary: build our own Vocabulary to map between words existing in both training and development sets and indexes, and vice versa.

3.2 Token encoding

Next step is converting all the words to their unique index. The method we'll be using is the so-called One-Hot Encoding. The length of the word vector is equal to the length of the vocabulary, and each observation is represented by a matrix with rows equal to the length of vocabulary and columns equal to the length of observation, with a value of 1 where the word of vocabulary is present in the observation and a value of zero where it is not.

Then all the unique index in the sentences were mapped to their real valued vectors of Dimensions 100 using Glove. Also, To deal with both short and long Comments, we will pad or truncate our comments to a specific length(seqlen=300). For comments shorter than seqlen, we will pad with 0s. And comments longer than seqlen we will truncate them to the first seqlen words.

3.3 RNN Classifier

In this assignment, We used Bidirectional Gated Recurrent Unit (GRU) using 100 dimensions to represent sequences by fixing the maximum length to 60 . Post padding with 0 was used for shorter sequences as it helps in preserving the information at the borders.

Bidirectional GRU's are a type of bidirectional recurrent neural networks with only the input and forget gates. It allows for the use of information from both previous time steps and later time steps to make predictions about the current state. Figure 1 displays the typical structure of a bidirectional recurrent neural network.

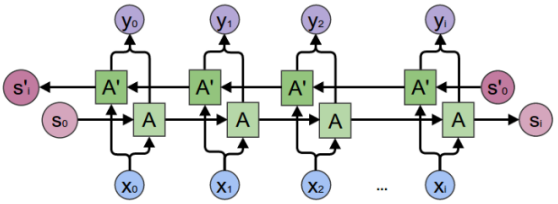

Figure 1: General Structure of Bidirectional Recurrent Neural Networks

After getting desired hidden representation from 2 layers it is passed to the Fully Connected layers followed by softmax Classifier for getting probability distribution among classes.

3.4 Training model

After Setting up our Bigru layer and the RNN classifier, we need to train our model. In order to do that first, we create an optimizer to update the parameters of the module. In this assignment, we use stochastic gradient descent (SGD). Next, we define our loss function. The loss function here is binary cross entropy. This function first feeds the predictions through a softmax layer, fixing the values between 0 and 1, then assign 1 to max probability and zero to others.

The train function iterates over all samples, one batch at a time.For each batch, we first zero the gradients.

We then feed the batch of sentences, batch.text, into the model and calculate the loss and F1_score using the predictions and actual labels. To update

	Train	Development	Test
Total number of samples	12000	2000	1001
Total number of tokens	345538	16261	55464
Average number of tokens per sample	29	28	26

Table 1: Number of samples and tokens for train, development, and test data

Label	Train Count	Development Count
NAG	5052(42.1%)	815(40.7%)
OAG	2708(22.6%)	485(24.3%)
CAG	4240(35.3%)	700(35%)

Table 2: Statistics of the labels for both train and development sets

the parameters, we use the gradients and optimizer algorithm with optimizer.step(). And finally, we return the model

3.5 Evaluation

evaluate is similar to training process except that we don't want to update the parameters when evaluating.So, with the removal of optimizer.zero_grad(), loss.backward() and optimizer.step(), we do not update the model's parameters when evaluating. No gradients are calculated on PyTorch operations inside the with no_grad() block. This causes less memory to be used and speeds up computation.

Then, we train the model through multiple epochs, At each epoch, if the validation loss is lowest among other epochs , we'll save the parameters of the model and then after training has finished we'll use that model on the test set.

4 Experiments and Results

I have trained different models in order to compare them and get the best classifier. The first attempt was implementing the basic Recurrent neural networks. Second, I trained bidirectional GRU model With only 5 epochs And the result were almost the same as RNN model. Finally, I tried to improve the results by increasing the number of epochs to 10 and it improved the result of the best model by around 5%. Table 3 Shows loss and weighted F1 core For both training and development sets.

Model	Loss	F1_Score
RNN	0.0056	38.98
BiGRU(5 epochs)	0.005	39.12
BiGRU(10 epochs)	0.0048	0.45

Table 3: Performance of different models

The model is not good enough to beat the baseline because of the simplicity. When I experimented with the simplified neural network, it turned out that it could not generalize as good as the more complicated model.

5 Conclusion

Although the built model performs fairly well and can be useful for the task of abusive language detection but it is not enough to win the competition. further adjustments such as parameter tuning and assembling with other models shall be considered to help improve the result.

References

[1] Chen, Y., Zhou, Y., Zhu, S., and Xu, H. *"Detecting offensive language in social media to protect adolescent online safety,"* in International Conference on Privacy, Security, Risk and Trust and International Conference on Social Computing (Amsterdam: IEEE), 71–80, 2012.

[2] Dinakar, K., Reichart, R., and Lieberman, H. *"Modeling the detection of textual cyberbullying,"*in 5th International AAAI Conference on Weblogs and Social Media / Workshop on the Social Mobile Web (Barcelona: AAAI), 11–17,2011.

[3] Mishra, P., Yannakoudakis, H., and Shutova, E. *"Neural character-based composition models for abuse detection,"* in 2nd Workshop on Abusive Language Online (Brussels: Association for Computational Linguistics), 1–10, 2018.

[4] Pavlopoulos, J., Malakasiotis, P., and Androutsopoulos, I. *"Deep learning for user comment moderation,"* in 1st Workshop on Abusive Language Online (Vancouver, BC: ACL), 25–35. doi: 10.18653/v1/W17-3004, 2017.