Abusive Language Detection

Mohammad Rajiur Rahman
UHID: 1407264
COSC6336 (Spring 2020)
mrahman13@uh.edu

Abstract

This report presents implementation of recurrent neural networks (RNN) based deep neural classifier that can identify aggressive messages in online posts. Models are designed using different algorithms implemented by pytorch[1] library. The best performing model achieved 48.75% f1-score (weighted average) over the given development data set.

1 Introduction

Abusive language can be defined as verbal expressions, that may or may not contains abusive words or phrases, with an intention to hurt any individual or any group of people (ethnicity, community, religious group etc). With the advent of social media abusive language has faster and longer reach to wide audience. Detecting abusive language at personal level is very important, but it is more critical and important at national and international levels as well. In recent past, we witnessed communal violence in many countries fumed from only one abusive tweet or post on social media (Blandfort et al., 2019). In addition, social media data is growing so faster that even referencing any statistics would be outdated by the time writing this sentence is completed. It makes manual moderation of online content impracticable. Detecting abusive language is further complicated by other factors like multi-lingual, social or regional context etc. The same word or group of words can be non-abusive or overtly/covertly abusive depending on rapidly changing contexts. Wiegand (Wiegand et al., 2019) examined the impact of data bias on abusive language detection and showed that this problem is closely related to how data have been sampled. Automatic detection of abusive language is getting more importance in recent days. A number of works have tackled this problem, or related ones, in the literature. Recent works focus on different deep learning approaches. In this work, we aim to analyze the data to extract features and experiment different RNN based models for the task.

[1] https://pytorch.org/

2 Dataset

The data for the task comes from Facebook. The dataset is divided in three sets. Training, development and test dataset have 12000, 2000 and 1001 records respectively. The training ahd development dataset have labels but test dataset has no labels associated.

2.1 Classes

There are three different classes[2] in the 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.

Figure 1 shows the number of records in different classes in training and development datasets.

The datasets are **skewed** but training and development dataset are skewed following similar distribution of number of records in each class labels.

[2] class definitions & examples copied from homework-2 description

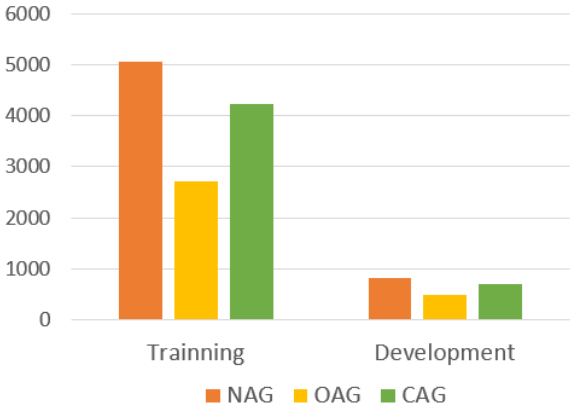

Figure 1: Class Distribution of Training and Development Dataset

2.2 Comment Lengths

Length of comments are analyzed for three datasets. Following observations are made for all of the sets, unless specifically mentioned:

- Mean comments length is in the range from 23 to 25.
- Some comments are of 0 length.
- Most of the reviews less than 100 as shown in Table 1
- There are quite a few comments that are extremely long which are less than 2.3% for all the datasets (Table 11).

Table 1: Lengths of Comments in Datasets

	Train	Dev	Test
count	12000	2000	1001
mean	24.57	23.79	23.04
std	36.64	27.62	28.26
min	0	0	0
25%	11	11	10
50%	16	15	15
75%	27	26	26
max	1094	312	411

2.3 Comment Language

The datasets have words from English and at least one other language. It is not possible to determine if there are more than one non-English language. Many of the comments have mixed words from English and non-English language. Dictionary from Pyenchant library [3] is used to determine if any word is English. Percentages of non-English words in each class of training and development class along with total in each of the three sets (including test set) are analyzed. It is observed that the ratio of comments with 0 (NIL), 0-25%, 25-50%, 50-75% and 75-100% non-English words are same across three different sets (Figure 2).

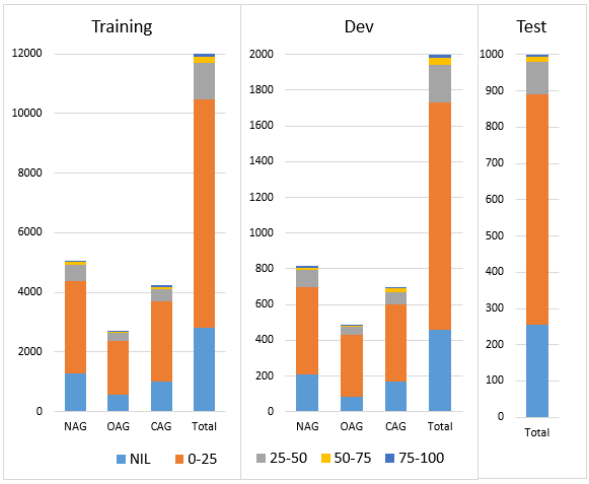

Figure 2: Non-English Words

3 Methodology

3.1 Pre-processing

Following steps are included in the preprocessing of the datasets:

Checking missing values. : Training and development datasets are checked for missing values i.e. if any comments has no labels. No such cases is found.

Removal of unwanted chracters : Following groups of characters are removed from comments in the datasets:

- URLs
- Newline and carriage returns
- Punctuation
- Numbers
- Non-ASCII characters

Lower case conversion : All comments are converted to lower-case.

Tokenization : The comments are tokenized using TweetTokenizer from nltk[4] library.

[3] https://pyenchant.github.io/pyenchant/

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

Class Label to Index : Class-label is changed to indexes as shown next to class label:

- CAG: 0
- NAG: 1
- OAG: 2

3.2 Vocabulary

Extracted tokens from training and development set are used to create the vocabulary. It contains 22,138 types.

3.3 Feature Extraction

Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) is used to extract features. Glove pre-trained word vectors are taken from its "Common Crawl (42B tokens, 1.9M vocab, uncased, 300d) dataset". Embedded word-list has 300 dimensional features for 19,447 types. Which means 2,691 types (1.22% of total types in vocabulary) are marked as unknown in the training and development dataset. These types are not removed from the datasets because of its insignificance. Table 2 shows number of tokens not in Glove for training, development and test sets. This occurrence has similar ratios in each of the datasets (around 1% only).

Table 2: Total Tokens & Tokens not in Glove

Sets	Total Tokens	Not in Glove (%)
Train	294843	3210 (1.09%)
Dev	47589	538 (1.13%)
Test	23063	235 (1.02%)

3.4 Models

Simple RNN and two variations of GRU from pytorch library are implemented. It is difficult to capture long-term dependencies through RNN because of gradients vanishing or exploding problems (Bengio et al., 1994). Long-short term memory (LSTM) is introduced by (Hochreiter and Schmidhuber, 1997) to capture long-term dependencies. GRU is a variation of LSTM. Chung showed some advantages of GRU over LSTM (Chung et al., 2014). List of models implemented for this experiment are discussed below:

Model-1: Simple Recurrent Neural Network (RNN) RNN module (Pytorch library) is used that applies a multi-layer Elman RNN. Model-1 uses following paramters:

- Number of Layers: 2
- Hidden Size: 128
- Non-linearity: tanh
- Bias: True
- Batch size: 64
- Dropout: 0.3
- Bidirectional: False

In this model output of RNN is sent to linear classifiers. Softmax output of the classifier is taken and argmax is used to determine the class label having highest probability in the classifier's output.

Model-2: Bidirectional Gated Recurrent Unit (BiGRU) GRU module (Pytorch library) is used that applies a multi-layer gated recurrent unit (GRU) RNN to an input sequence. Model-2 uses following paramters:

- Number of Layers: 3
- Hidden Size: 128
- Bias: True
- Batch size: 64
- Dropout: 0.1
- Bidirectional: True

In this model the directions of the GRU is separated. Then last hidden layer per direction is picked up and concatenated as a new hidden layer. This new hidden layer is passed to a linear classifier and softmax of the classifier's output is taken as final output of the model. Argmax function is used to determine the class label with highest probability.

Model-3: Bidirectional Gated Recurrent Unit (MulBiGRU) GRU module (Pytorch library) is used that applies a multi-layer Elman RNN with tanh or ReLU non-linearity to an input sequence. Model-1 uses following paramters:

- Number of Layers: 2
- Hidden Size: 128
- Bias: True
- Batch size: 64

- Dropout: 0.3
- Bidirectional: True

In this model the directions of the GRU is separated. Then last hidden layer per direction is picked up and concatenated as a last hidden layer. Sigmoid of this hidden layer is taken and then passed to a linear classifier and softmax of the classifier's output is taken as final output of the model. Argmax function is used to determine the class label with highest probability.

Training All three models are trained with following parameters:

- Learning rate: 0.2
- Momentum: 0.9
- Epochs: 100
- Optimizer: Stochastic gradient descent (SGD)
- Loss Function: Cross Entropy Loss

4 Results

Following are the performance metrics of the models (determined using classification report from scikit-learn library):

Table 3: Performance Metric of Model-1 (RNN)

	precision	**recall**	**f1-score**
OAG	0.41	0.55	0.47
NAG	0.54	0.56	0.55
CAG	0.39	0.19	0.26
accuracy			0.47
macro avg	0.45	0.43	0.43
weighted avg	0.46	0.47	0.45

Table 4: Performance Metric of Model-2 (BiGRU)

	precision	**recall**	**f1-score**
OAG	0.41	0.53	0.46
NAG	0.55	0.59	0.57
CAG	0.48	0.21	0.3
accuracy			0.48
macro avg	0.48	0.45	0.44
weighted avg	0.48	0.48	0.47

Table 5: Performance Metric of Model-3 (MulBiGRU)

	precision	**recall**	**f1-score**
OAG	0.42	0.54	0.47
NAG	0.56	0.58	0.57
CAG	0.49	0.26	0.34
accuracy			0.49
macro avg	0.49	0.46	0.46
weighted avg	0.49	0.49	0.48

5 Discussions

Table 6 summarizes weighted average F1-score, precision and recall along with total number of misclassified labels by the three models implemented. F1-measure and misclassified labels indicated that Model-3 performs better than other two models on development set.

Table 6: Comparison of Models

	Model-1	**Model-2**	**Model-3**
F1-score	0.45	0.47	0.48
Precision	0.46	0.48	0.49
Recall	0.47	0.48	0.49
Misclassified	1068	1046	1026

The development and training datasets are skewed. However all three models' prediction on development set shows more skew by failing to detect further more comments with OAG label. Figure 3 shows the skewed predictions on development and test sets data for all three models. Since the glove embedding has no contextual information, it was hypothesized that the model would perform better detecting overt aggressive (OAG) labels compared to covert agressive (CAG) labels. Because comments with OAG labels uses aggressive or negative words, where as comments with CAG label do not. However, the hypothesis is proved to be wrong from the output of the models. One probable reason could be skewed dataset used for training.

Table 7, 8, 9 are the confusion matrices of the models (determined using scikit-learn[5] library). The matrices show true labels along the columns and predicted labels along rows of the table.

The baseline model has misclassified 855 labels with weighted average F1-measure 0.57. The confusion matrix of baseline model is given in Table 10. All of our Models (1, 2 and 3) out performs the baseline model for detecting CAG class only, but

[5]https://scikit-learn.org/

Table 7: Confusion Metrics of Model-1 (RNN)

	OAG	CAG	NAG
OAG	93	249	143
CAG	79	383	238
NAG	65	294	456

Table 8: Confusion Metrics of Model-2 (BiGRU)

	OAG	CAG	NAG
OAG	104	246	135
CAG	67	370	263
NAG	46	285	484

Table 9: Confusion Metrics of Model-3 (MulBiGRU)

	OAG	CAG	NAG
OAG	126	234	125
CAG	82	379	239
NAG	48	298	469

Table 10: Confusion Metrics of Baseline

	OAG	CAG	NAG
OAG	261	144	80
CAG	166	352	182
NAG	95	188	532

performs poorly for other two classes, especially for OAG class.

6 Constraints and Limitations

The homework involves understanding concepts for recurrent neural networks & how it can be used to solve the problem. The next steps are analyzing the data and then using pytorch library to implement model(s) to perform the task. It was difficult to test different algorithms and methods from the library and understand their implication on the given problem as training a model would take a long time. It was not possible to implement some of the features as well. All these probable solutions are listed in section 7.1 and 7.2.

7 Future Work

7.1 Hand-crafted features for each Comment

Following Handcrafted features are extracted to be added to the models' RNN output:

- Number of words with all capital
- Number of words with initial capital (title)
- Number of negative words (using profanity package[6])
- Number of exclation marks
- Number of emojis (datasets use latin encoding, it is not possible to indentify unicode emoji characters)

7.2 Experiment with different Algorithms and Methods

Following algorithms and functions from pytorch libray may be explored:

[6]https://pypi.org/project/profanity/

Optimizers Adam, ASGD, Adagrad etc

Loss Function Binary Cross Entropy loss.

7.3 Drop Outlier Records from Dataset

Comments with length zero can be dropped as keeping these comments do not make any sense for analysis. Comments with lengths more than 100 can be dropped from training set as outliers and model performance can be checked. Table 11 shows number of such records.

Table 11: Number of Comments after removing Comments with more than 150 tokens and 0 tokens

	Train	Dev	Test
total comment	12000	2000	1001
comment length<=100	11718	1954	984
comment lenght = 0	18	6	2
final comments count	11700	1898	982

7.4 Language Models

Language models like BERT (Devlin et al., 2018), ELMo(Gardner et al., 2018) etc., may be explored. Glove is a static word embedding that fails to capture context or high level information, which is very important for abusive language detection. Glove generates fixed embedding for the same word in different contexts. On the contrast, Language Models like BERT captures word semantics in different contexts to address the issue of context-dependent nature of words.

8 Conclusion

This report presents studies and experiments for abusive language detection in social media based on RNN-based deep neural classifiers. Ranking of the best model of this work is 7 with a weighted F1

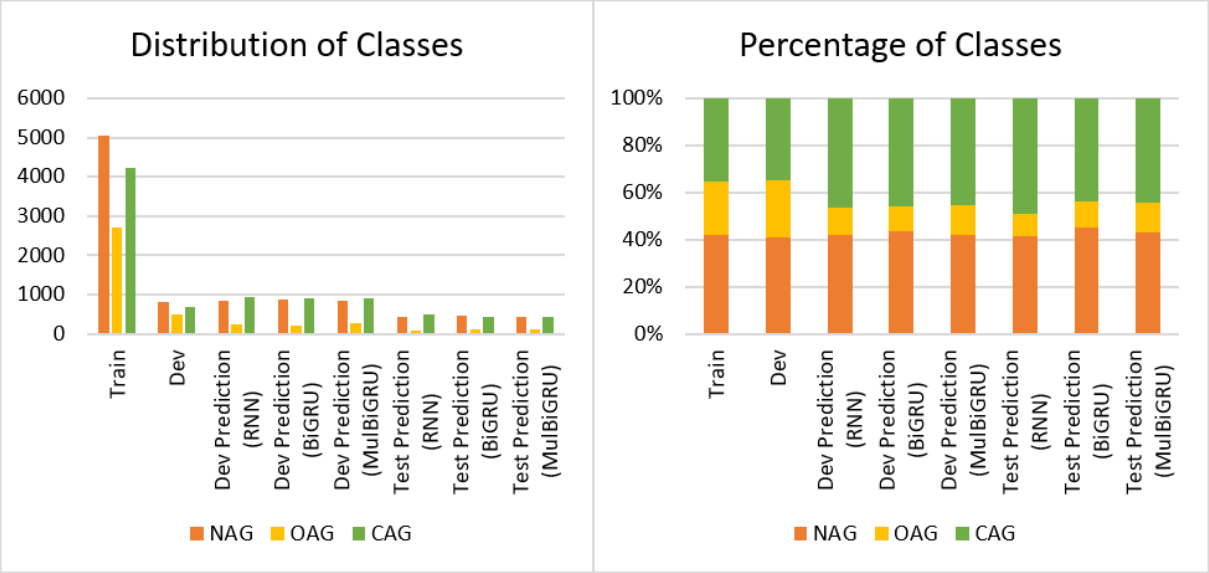

Figure 3: Class Distribution across Training & Developments sets and Predictions by Models on Tests sets

score of 46.81% on the test dataset. The report also analyzes the probable pitfalls of the implemented models and includes possible improvements as future work.

References

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166.

Philipp Blandfort, Desmond U. Patton, William R. Frey, Svebor Karaman, Surabhi Bhargava, Fei-Tzin Lee, Siddharth Varia, Chris Kedzie, Michael B. Gaskell, Rossano Schifanella, Kathleen McKeown, and Shih-Fu Chang. 2019. Multimodal social media analysis for gang violence prevention. *Proceedings of the International AAAI Conference on Web and Social Media*, 13(01):114–124.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: the Problem of Biased Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 602–608, Minneapolis, Minnesota. Association for Computational Linguistics.