

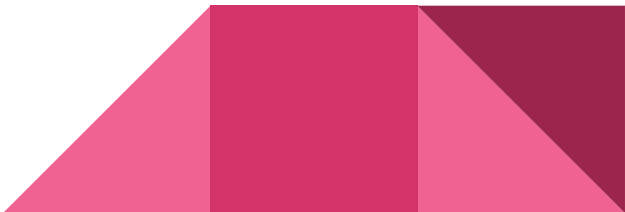
Sentiment Analysis to Classify Online Abuse

Problem Statement: With the increase in online presence, there is an equal increase in responsibility to moderate these kinds of activities online, so as to provide a safer environment for everyone. Automated filters to detect such activities have repeatedly failed to detect these colluded users, which motivates us to develop a learning model which can detect the negativity and classify them. In our model, we find negative comments or intent and try to classify them or group them with other similar comments.

William Scott - MT18026

Srivatsava - MT18054

Subhani - MT18117



Dataset

We are going to use a mixture of datasets as we faced a huge problem in the datasets where it says that few datasets are not labelled.

1. Data generated for the Wikipedia Detox project is available under free licenses on the Wikipedia Talk Corpus
2. 2500 tweets crawled by us and self annotated

Preprocessing

As we are dealing with textual data, there is a lot of preprocessing required • Set all characters to lowercase

• Remove numeric characters, • Remove punctuation, • Remove stop words, • Tokenization technique, • Lemmatization technique • Noise removal technique • Stemming technique



Approaches Used

Baseline

- Logistic Regression
- Multinomial Naive Bayes

Baseline

- Multilayer Perceptron
- Convolutional Neural Network
- LSTM - Recurrent Neural Network



Results - Base Models

Multinomial Naive Bayes

Predicted \ True	None	Sexist Comment	Personal Attack
None	381	83	8
Sexist Comment	28	443	44
Personal Attack	18	117	368

Logistic Regression

Predicted \ True	None	Sexist Comment	Personal Attack
None	383	53	36
Sexist Comment	43	371	101
Personal Attack	13	53	432

Results - MLP

```
f1_score: [0.78066914 0.83937824 0.80178838]
           precision    recall  f1-score   support

    0         0.71         0.86         0.78         366
    1         0.80         0.88         0.84         367
    2         0.96         0.69         0.80         392

 avg / total         0.83         0.81         0.81        1125
```

```
array([[315, 46, 5],
       [ 38, 324, 5],
       [ 88, 35, 269]])
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              nesterovs_momentum=True, power_t=0.5, random_state=None,
              shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
              verbose=False, warm_start=False)
```

Train: 0.997037037037037

Results - LSTM

Validation Score ~67%

click to scroll output; double click to hide	Output Shape	Param #
embedding_1 (Embedding)	(None, 2000, 50)	500000
conv1d_1 (Conv1D)	(None, 2000, 32)	4832
max_pooling1d_1 (MaxPooling1D)	(None, 1000, 32)	0
cu_dnnlstm_1 (CuDNNLSTM)	(None, 18)	3744
dense_1 (Dense)	(None, 3)	57
Total params: 508,633		
Trainable params: 508,633		
Non-trainable params: 0		

[0.	0.	0.52459016]		
	precision	recall	f1-score	support
	0	0.00	0.00	152
	1	0.00	0.00	138
	2	0.36	1.00	160
avg / total	0.13	0.36	0.19	450

```
None
Epoch 1/10
4050/4050 [=====] - 5s 1ms/step - loss: 0.6386 - acc: 0.6667
Epoch 2/10
4050/4050 [=====] - 3s 661us/step - loss: 0.6377 - acc: 0.6667
Epoch 3/10
4050/4050 [=====] - 3s 658us/step - loss: 0.6374 - acc: 0.6667
Epoch 4/10
4050/4050 [=====] - 3s 656us/step - loss: 0.6373 - acc: 0.6667
Epoch 5/10
4050/4050 [=====] - 3s 657us/step - loss: 0.6372 - acc: 0.6667
Epoch 6/10
4050/4050 [=====] - 3s 661us/step - loss: 0.6372 - acc: 0.6667
Epoch 7/10
4050/4050 [=====] - 3s 667us/step - loss: 0.6369 - acc: 0.6667
Epoch 8/10
4050/4050 [=====] - 3s 660us/step - loss: 0.6371 - acc: 0.6667
Epoch 9/10
4050/4050 [=====] - 3s 656us/step - loss: 0.6370 - acc: 0.6667
Epoch 10/10
4050/4050 [=====] - 3s 658us/step - loss: 0.6367 - acc: 0.6667
Accuracy: 66.67%
```

Results - NN

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	41505792
activation_1 (Activation)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
activation_2 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 3)	1539
activation_3 (Activation)	(None, 3)	0

=====
Total params: 41,769,987
Trainable params: 41,769,987
Non-trainable params: 0

Epoch 1/2

3015/3015 [=====] - 6s 2ms/step - loss: 0.7025 - acc: 0.6939

Epoch 2/2

3015/3015 [=====] - 3s 1ms/step - loss: 0.0690 - acc: 0.9794

```
[0.81973817 0.86530612 0.86258776]
              precision    recall  f1-score   support

     0         0.85        0.79        0.82         515
     1         0.88        0.85        0.87         499
     2         0.82        0.91        0.86         471

 avg / total         0.85        0.85        0.85        1485
```

```
array([[407,  43,  65],
       [ 44, 424,  31],
       [ 27,  14, 430]])
```

Test accuracy: 0.849158249198387

Results - CNN

Validation Score ~36%

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 81065)	0	
embedding_1 (Embedding)	(None, 81065, 100)	8106500	input_1[0][0]
reshape_1 (Reshape)	(None, 81065, 100, 1 0)		embedding_1[0][0]
conv2d_1 (Conv2D)	(None, 81061, 1, 100 50100)		reshape_1[0][0]
conv2d_2 (Conv2D)	(None, 81062, 1, 100 40100)		reshape_1[0][0]
conv2d_3 (Conv2D)	(None, 81063, 1, 100 30100)		reshape_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 1, 1, 100)	0	conv2d_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 100)	0	conv2d_2[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 100)	0	conv2d_3[0][0]
concatenate_1 (Concatenate)	(None, 1, 1, 300)	0	max_pooling2d_1[0][0] max_pooling2d_2[0][0] max_pooling2d_3[0][0]
flatten_1 (Flatten)	(None, 300)	0	concatenate_1[0][0]
dropout_1 (Dropout)	(None, 300)	0	flatten_1[0][0]
dense_1 (Dense)	(None, 3)	903	dropout_1[0][0]

Total params: 8,227,703
Trainable params: 8,227,703
Non-trainable params: 0

Epoch 1/2

500/3600 [==>.....] - ETA: 2:00 - loss: 1.0960 - acc: 0.3620

Analysis

- The classes in the data are to a major extent overlapping with each other owing to the fact that abusive slurs share a common vocabulary.
- CNN's were initially used to classify images but recently
- The neural networks were able to detect these but required many epochs to converge with better accuracy
- Models that were trained on one kind of dataset are tested with another geographical dataset, we found that the accuracy reduced a bit and we tuned the feature parameters to improve a little.



Individual Contribution

Each of us have contributed equally for three of the advanced techniques involved which involved designing the architectures of the Neural Networks.

For the base models, each of us split the work equally which involved

- Data collection, cleaning and other preprocessing techniques
- 2 base models *viz.*, Logistic Regression, Multinomial Naive Bayes

