# Automated Offensive Language Detection

Omar Elshrief Aya Amin Mariam Hazem Mohamed Hatem

**Department of CS, Faculty of Engineering Alexandria**

## Abstract

Nowadays we may say there is a digital world that exists in parallel with our daily activities. It is represented in social media forums. Currently, people highly rely on social media to interact with other people. Given the complicated environment of social media, it has become very difficult for users to avoid encountering offensive content from time to time. In this paper we shall introduce our approach to tackle the problem of offensive language in online social media.

## Introduction

Fighting abusive language online is becoming more and more important in a world where online social media plays a significant role in shaping people's minds, especially when nowadays there are no restrictions for the ages that are using Social media, it becomes a necessity to fight and stop abusive and offensive languages in order to keep our children safe.

## Data

Webeganwith'Offenseval-training-set1.tsv' and added some more Data from Twitter and Facebook, beside some lexicons classified as offensive by Facebook and GOOGLE, all together was about 28k Tweets..

Pre-processing: First thing is cleaning the data. This is done by removing all unnecessary tokens like @USER and all the mention stuff using regex. We removed all HTML Tags and links using HTML Parser and regex, also removed all the Emoji patterns from the data using regex. We also removed all punctuations and special characters as well as extra spaces using regex and string manipulations. Second phase is Normalizing and Standardizing the data, we started by setting all characters to lower-case, then we expanded all contractions using a predefined dictionary like "isn't=>isnot".

Afterwards, we kept the data like that (for a later use) and took a copy from is at the current state and continued the operation.

We then we used NLTK tokenizer to tokenize the data, using the produced tokens. We removed all the stop words using NLTK stop words English list which reduced the dimensionality of the data, after that we stemmed the data using NLTK stemmer…..

Going back to our original copy of data, we used Part of speech for capturing the syntactic structure, we used NLTK to construct Penn Part-of-Speech (POS) tag for each token of the data, and then we padded each token with its POS tag and reconstructed the document. After that we removed the stop words of the document by NLTK and REGEX to first separate each token from its tag and compare with stop words. Finally, we lemmatized the data using NLTK Lemmatizer.

Till now we have 2 copies of the dataset: Tagged and Untagged.

## Features

Since we have 2copiesof data as mentioned above, the copy that was stemmed (untagged) is used to generate a count-Vector 1-gram and 2- grams weighed by TF-IDF of each token, For the second copy of the dataset (the tagged one), we generated a count-Vector (1-gram) with a different customized token pattern to regard the Word/Tag as a token in order to capture the syntactic structure. We used both the feature vectors to train our models in an ensemble classifier which will be illustrated next..

## Model

First we coded 7 Different models: Logistic regression, Linear SVM, Random-Forests, Naïve Bayes, Stochastic Gradient Descent classifier (SGDC), K-neighbors classifier and Ensemble Vote Classifier. For each model, we used Grid-Search and Cross validation for tuning the parameters for each model, the results of the grid-search:-

```
Logistic Regression: (penalty='l1', C=0.3,
max_iter=15000, solver='liblinear')

Naïve Bayes:-
(alpha=0.001,fit_prior=False)

SGDC: (penalty='l1',  random_state=42,
loss='log', l1_ratio=0.1,alpha=0.0001,
learning_rate='adaptive', tol=None,
max_iter=3000, n_jobs=-1, eta0=0.001)

Linear SVC: (penalty='l1', C=0.2,
max_iter=10000, random_state=45,
loss='squared_hinge', dual=False)
```

Random-Forests: for random-forests classifier we noticed that when increasing the number of estimators we may get better training score but the optimization is lowered so we get high percentage of overfitting. So, best parameters for our Random forests model

```
(n_estimators=20,
max_features=8000,random_state =46 )
```

**K-neighbours**: To get the best fit K, we trained the model with the data 25 times each with different K, eventually, Figure-11 shows value of K vs Accuracy of the Model, best results found at K = 19

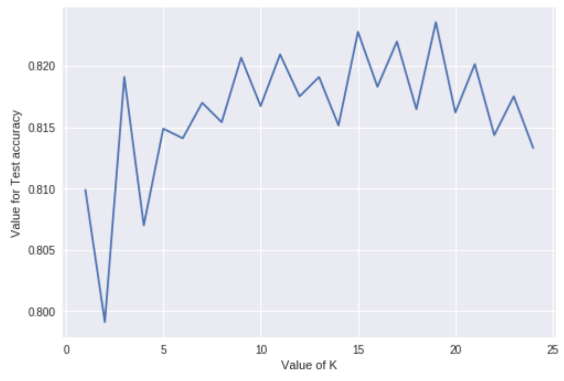


*Figure 11-K-NN best K*

We trained each model separately each time using different vector type, and compared the results. We tested each model using 5-fold cross-validation holding out 10% of the sample for evaluation to help prevent over-fitting.

After using a grid-search to iterate over the models and parameters to find best model with best feature vector, we find that the Logistic Regression and Linear SVM tended to perform significantly better than other models using the count vectors (1-Grams and 2-grams). We decided to use a **Logistic regression with L1 regularization, Linear SVM and Random forests with the Ensemble voting classifier with hard voting as a final model with count-Vectors.**

## Results

We used 90% of the data as training and 10% as validation data.
The best performing model out of the 7 was Logistic Regression with testing f1_score = 0.88 and accuracy 0.88, weighted average = 0.89.
Shown in figure 1 is the Confusion matrix of Logistic Regression results. Also Figure 3 shows the classification report. Figure 2 shows the ROC curve the logistic regression.

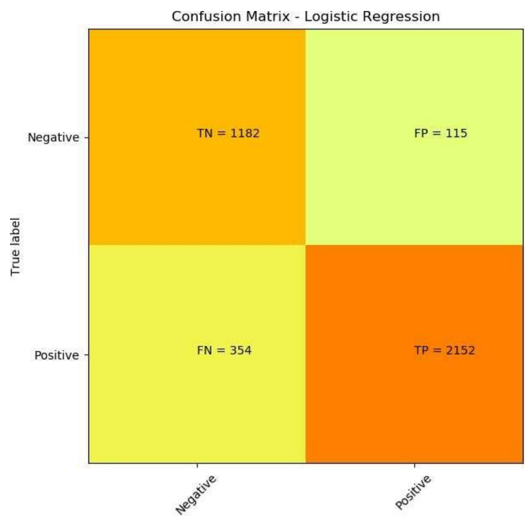


Figure 1-Logistic Regression Confusion Matrix

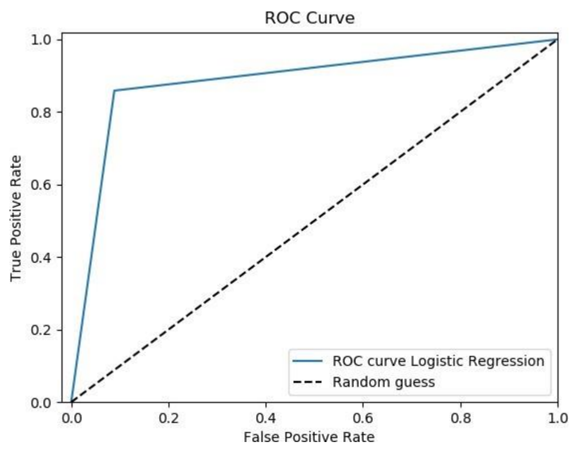
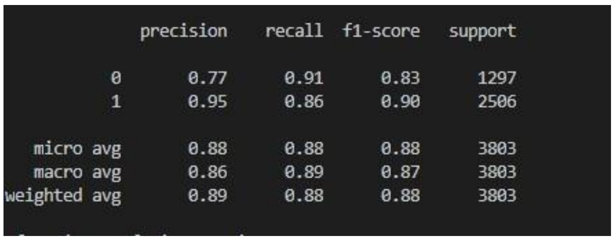


Figure 2-ROC Logistic regression



```
              precision    recall  f1-score   support

           0       0.77      0.91      0.83      1297
           1       0.95      0.86      0.90      2506

   micro avg       0.88      0.88      0.88      3803
   macro avg       0.86      0.89      0.87      3803
weighted avg       0.89      0.88      0.88      3803
```

Figure 3-Logistic Regression Classification report

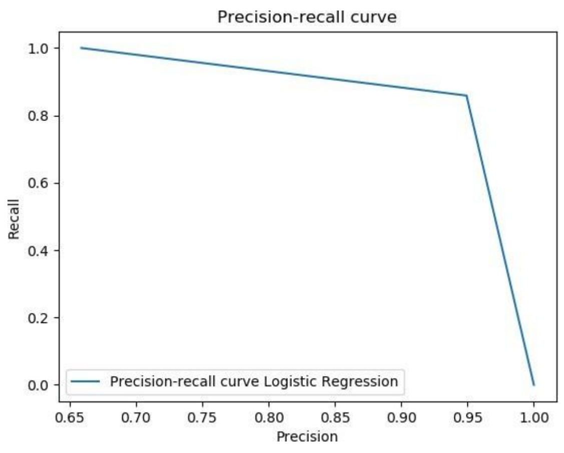
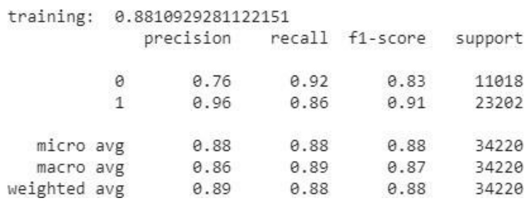


Figure 4-Logistic Regression Precision- recall

Although logistic regression was performing very good, but our final model was Ensemble Classifier with Logistic regression. For Linear SVM and Random forests with hard voting performance was slightly better than Logistic regression. Our model was more accurate than Logistic regression. The model has an overall precision 0.89, recall of 0.87, and F1 score of 0.88, the model was best-fit one as training accuracy and score almost was the same as testing score. Figure5 shows the results of training the model. Figure 6 shows the score of testing the model.

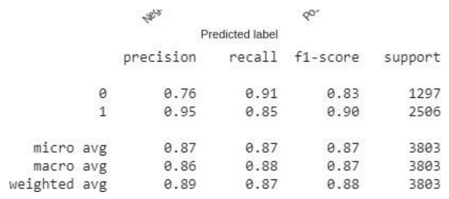


```
training:  0.8810929281122151
              precision    recall  f1-score   support

           0       0.76      0.92      0.83     11018
           1       0.96      0.86      0.91     23202

   micro avg       0.88      0.88      0.88     34220
   macro avg       0.86      0.89      0.87     34220
weighted avg       0.89      0.88      0.88     34220
```

Figure 5- Training score Final model



```
              precision    recall  f1-score   support

           0       0.76      0.91      0.83      1297
           1       0.95      0.85      0.90      2506

   micro avg       0.87      0.87      0.87      3803
   macro avg       0.86      0.88      0.87      3803
weighted avg       0.89      0.87      0.88      3803
```

Figure 6-Testing score Final model

Out of 3883 testing set 2566 was true offensive 1297 was not, our model predicted 2141 as Offensive (miss classifying 365 tweet) and 1183 as Not (miss classifying 114 tweet), Figure 8 shows the confusion matrix of our model.

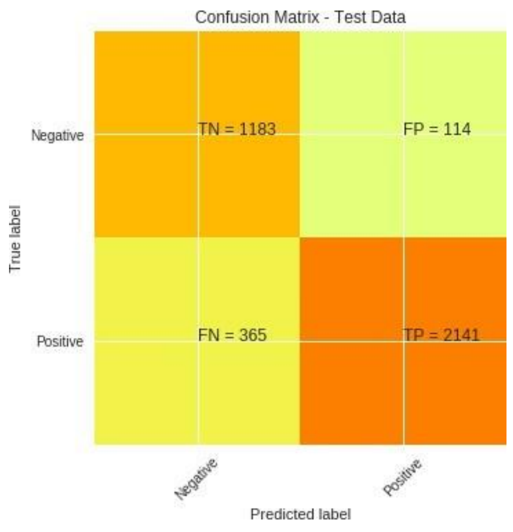


Figure 7-Final model confusion Matrix

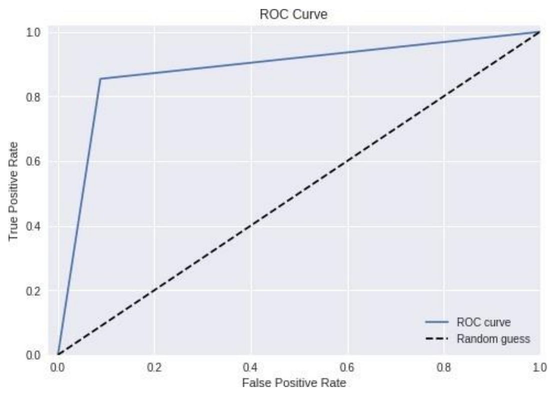


Figure 8-ROC of the Final Model

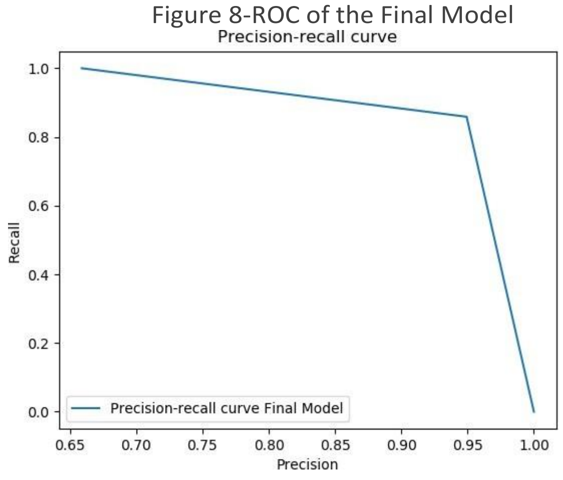


Figure 9-Final model precision recall

## Conclusion

Offensive tweets are not that hard to find. Consistent with previous work, we find that certain terms are particularly useful for distinguishing between offensive speech and normal one.

We explored Ensemble classification which happened to be very useful; especially, when using multi classifier with almost same accuracy, total accuracy and precision of the classification shall be better...