Hate Speech and Offensive Language Detection

Dimitrios Patiniotis Spyropoulos
Natural Language Processing

July 1, 2022

1 Introduction

Hate speech and offensive language detection is a relatively simple yet challenging task that has very important real-world applications. It is challenging due to the fact that both hate speech and offensive language can take many different forms and are context related and important since algorithms that carry out this task regulate the content of most (if not all) major social media platforms.

The purpose of this project was to experiment with different machine and deep learning algorithms to tackle this multi-class classification problem. In the *Theoretical Background* section we will take a brief overview of the algorithms used in the experiment. In the *Dataset Description* we will analyse the dataset chosen as well as ways to deal with imbalanced datasets. Finally, we will go through our experiments and their results and we will compare them in the *Experiments* section.

2 Theoretical Background

2.1 Traditional Machine Learning

Before Long Short-Term Memory (LSTM) networks became a standard when dealing with text, traditional machine learning algorithms were the most popular method to deal with NLP tasks. For the purposes of this project, we will compare the performance of five of the most popular shallow algorithms with LSTMs and Transformers.

2.1.1 Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) is one of the most frequent (if not the most frequent) implementations of the Naive Bayes (NB) algorithm when dealing with text. The NB algorithms make the *naive* assumption that every feature is conditionally independent with every other feature given the value of the class variable. In MNB the distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ where each parameter θ_{yi} is estimated by a smooth version of maximum likelihood $\theta_{yi} = \frac{N_{yi}+\alpha}{N_y+\alpha n}$.

2.1.2 Logistic Regression

Logistic Regression (or Logit Regression) is a linear model for classification. The possible outcomes of a prediction by the model are modeled by a logistic function. Since we have more than two classes we will use the multinomial logistic regression.

2.1.3 Support Vector Machine

A Support Vector Machine (SVM) constructs hyper-planes in a dimensional space, the size of which is determined by the number of features. With slight modifications SVMs can be used for classification and regression tasks.

2.1.4 Random Forest

A random forest is an ensemble estimator that fits many decision tree classifiers and uses averaging to improve the accuracy and prevent over-fitting. Each of these trees is built using a portion of the training set. Randomness is also introduced in random forests when splitting each node of each dicision trees, were we can use a random subset of features. This randomness is purposeful and aims to solve a major flaw of decision trees, over-fitting, by reducing the variance.

2.1.5 AdaBoost

The main idea of AdaBoost is fitting a sequence of weak shallow learners, such as small decision trees, on repeatedly modified versions of the data and make predicitons based on the weighted majority vote of these models. In each boosting iteration the data modifications consist of applying weights$w_1, w_2, ..., w_n$ to each sample. At each step, if the training sample is correctly predicted its' weights are decreased and vice versa if the boosted model fails to predict correctly.

2.2 Long Short-Term Memory

An LSTM network is a type of recurrent neural network that can learn long-term dependencies between time steps of sequence data. LSTM has a chain structure that contains four different neural networks and memory blocks (also called *cells*). Cells are responsible for retaining information whilst memory manipulations can be done by the three available *gates*:

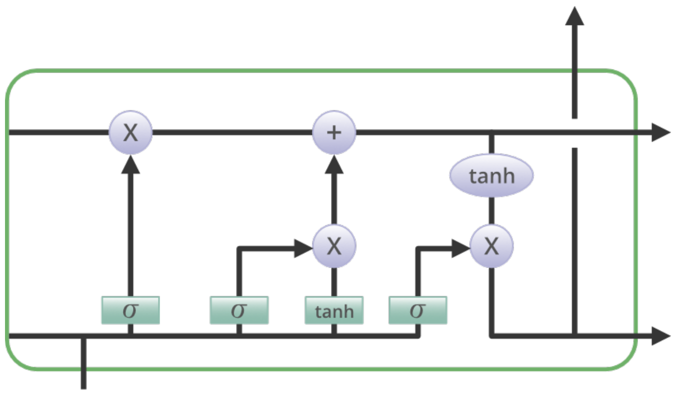

Figure 1: LSTM Architecture

2.2.1 Forget Gate

The forget gate removes the information that is no longer useful. Given two inputs (current and previous cell output) the forget gate returns a binary output that determines weather a piece of information is forgotten or not.

2.2.2 Input gate

The input gate is responsible for adding useful information to the cell state. The values to be remembered are regulated by a sigmoid function. A vector is then created using the tanh fuction that outputs values in range of -1 and +1 that then get multiplied with the output of the sigmoid function to obtain the useful information.

2.2.3 Output gate

The output gate is tasked with extracting the information stored by the input gate. After generating a vector via the application of a tanh function to the cell and multiplying it with the output of a sigmoid function that has two inputs, similar to the one used in the forget gate, we get the LSTM output and input to the next cell.

2.3 Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers (BERT) is an open source machine learning framework that was introduced by Google in 2018 and is widely used in natural language processing tasks. It enables computers to better understand ambiguous language by establishing context based on the surrounding text.

BERT is based on Transformers, a deep learning model in which every output is connected to every every input element. The weighting of the input data is done by a mechanism called *self-attention*. Unlike sequential models that can only read text input from left to right or from right to left, BERT is designed to read in both directions at once. Due to this functionality BERT is pre-trained on two different NLP tasks: Next Sentence Prediction and Masked Language modeling.

3 Dataset Description

3.1 Datasets used

The dataset we will be using was compiled by Thomas Davidson et al. (2017). It contains 25.000 tweets that contain samples from a hate speech lexicon that was created by hatebase.org. Annotators classified these tweets as one of the following three classes: *hate speech*, *offensive language* or *neither*. The annotators did not only take into account the words that appeared in each tweet, but also the context in which they were being used. The inter-annotator agreement was 92%. After removing some tweets that did not fit the criteria of neither of these three classes, we are left with a dataset of 24.802 tweets.

The immediate observation when dealing with this dataset is that it is very imbalanced, with the dominant class representing 76% of all tweets.

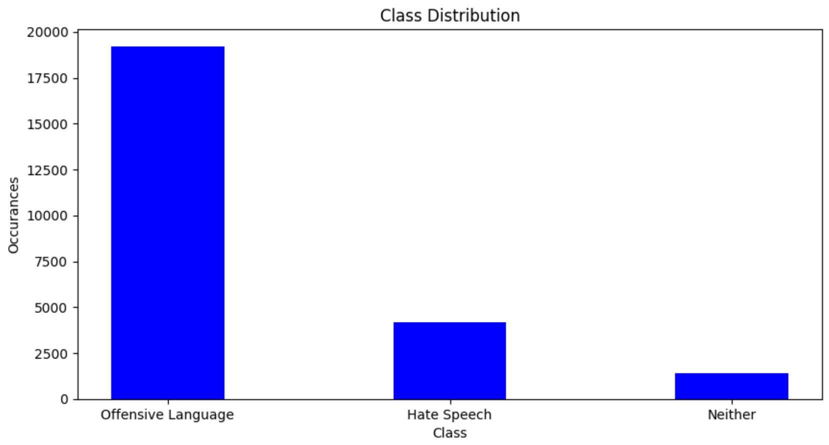

Figure 2: Original class distribution

This imbalance may (and as we will see) leads to flawed models. There are many different ways to deal with imbalanced datasets, and the one used in this report is random under-sampling of the dominant classes. This results in a balanced dataset where each class is represented 1430 times each, and a total of 4.290 tweets, almost 5 times smaller than the original one. In the experiment section we will train and test each model twice, once with the original dataset and once with the balanced one.

3.2 Data Reprocessing

Shallow ML algorithms and LSTMs require different data reprocessing procedures. As a common first step we clean each tweet: After loading the dataset tweets, we remove special characters and hashtags. We then use Porter stemmer to stem each tweet, excluding all English stop words, and remove all links. We then split the data, holding out a test set so that we can test the performance of each classifier when dealing with never seen before data. For ML algorithms we use a simple Count Vectorizer to turn the tweets into vectors that the models will use as inputs. As for the data used in our Deep Learning models, we tokenized and padded the tweets.

4 Experiments

4.1 Traditional Machine Learning Algorithms

Shallow machine learning algorithms perform well on both the original dataset and the balanced one, considering that the SVM model has an f1 score of 0.91 and logistic regression one reaches 0.90. For reference Davidson et al. (2017) also achieve an f1 score of 0.91.

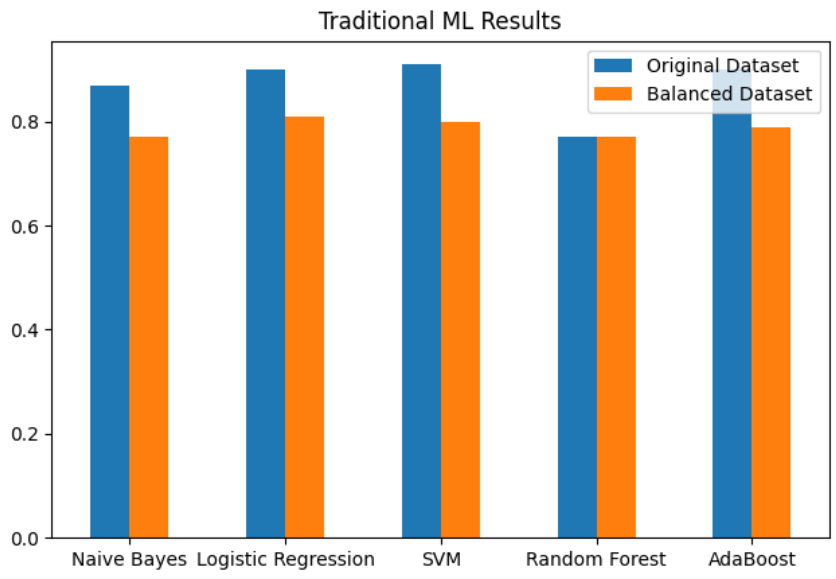

Figure 3: Traditional ML F1 Scores

It's important to note that the random forest classifier shows no decline in performance (f1 score of 0.77) when dealing with the reduced dataset, although its size is 5 times smaller than the original one. When taking a closer look at the confusion matrices of the 2 models we can observe that the model trained on the original dataset only predicts the dominant class (offensive language) (Figure 4).

4.2 Deep Learning Algorithms

Deep learning algorithms performed considerably worse than their shallow counterparts, as shown in Figure 5. The single LSTM to a dense layer model reached an f1 score of 0.84 on the original dataset and 0.72 on the balanced one. BERT followed by a dense layer had f1 scores of 0.83 and 0.67 repsectively. BERT followed by an LSTM layer and then a feed forward layer proved to be a notable improvement, achieving f1 score 0.86 on the original set and 0.71 on the balanced one.

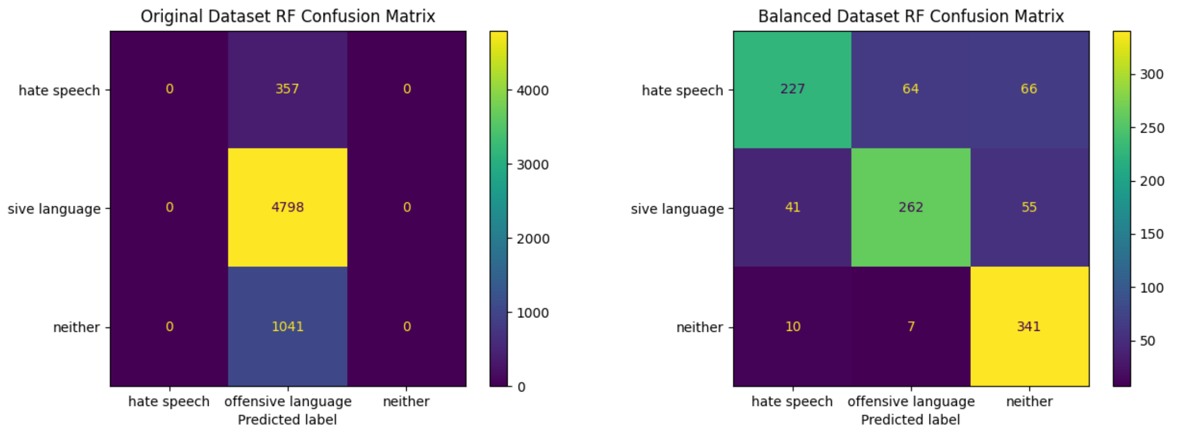

Figure 4: Random Forest Confusion Matrix

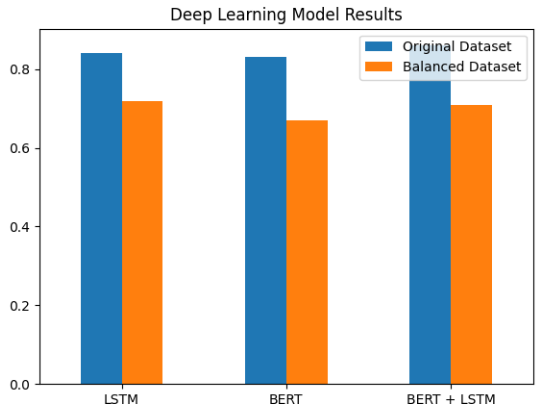

Figure 5: Deep Learning F1 Scores

It does not come as a surprise that traditional machine learning algorithms outperform LSTMs and BERT considering the relatively small size of both datasets. Furthermore, the standard version of BERT, which was used for these experiments, was not trained to understand context in tweets, where informal abbreviations and permutation of words are common.

Another thing to note is that in all models when dealing with the balanced dataset the *neither* class had considerably higher precision than the other two classes (can also be observed in Figure 4, Balanced Dataset RF Confusion Matrix). Most of the miss-classifications of the models that were tested were between the hate-speech and offensive language classes. This makes sense, and is actually being mentioned in Davidson et al. (2017) when describing the challenges of the annotation process.

5 Conclusion

Through our tests we reconfirmed that shallow machine learning models still have a place in the tool arsenal of an NLP engineer, especially when dealing with smaller datasets or/and niche cases. We also explored how a reduced but balanced dataset leads to a bit worse but less unbiased models. As a next step to this study we could augment the dataset to achieve a balanced set as well as stacking LSTM layers and/or using bidirectional LSTMs.

The complete codebase of this project can be found here.