Toxic Comment Classification - Report

Mick van Hulst Dennis Verheijden Roel van der burg Brian Westerweel
Joost Besseling

April 18, 2018

1 Introduction

This report describes the results for the first competition of the course Machine Learning in Practice. The source code for this project can be found by going to the public Github repository.

During the project each of the team members were assigned specific tasks. Do take note that our approach can be considered agile as we switched roles constantly and helped each other out where we could:

- **Mick van Hulst:** Focused on the preprocessing of the data, setting up the (GCP) Google Cloud Platform environment and the final report
- **Dennis Verheijen:** Focused on feature extraction, writing training routines, processing predictions for kaggle, making the code format files for GCP and helped in the report for GCP and for the competition.
- **Roel van der Burg:** Focused on developing the LSTM, bidirectional LSTM network and the final report
- **Brian Weserweel:** Focused on developing the LSTM network and the CNN.
- **Joost Besseling:** Focused on developing the bidirectional LSTM and writing the report.

2 Problem statement

We have chosen the toxic comment classification challenge on Kaggle. This challenge consisted of a dataset of roughly 160 thousand Wikipedia comments which had to be classified given six different classes. The classes are various types of toxicity: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, `identity_hate`. Moreover, these classes are not mutually exclusive, which means that our problem is a multi-label classification problem.

3 Dataset

The training and test set, provided by Kaggle, consists of roughly 160 thousand entries. Each entry has a comment id, the comment and, for the training set, the respective class labels.

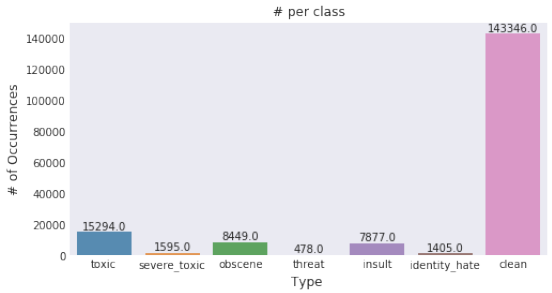

Figure 1: Data distribution

One important characteristic of our data is that the set is very imbalanced (figure 1). This posed many challenges. Over 96 percent of the dataset was clean and thus did not have any label. This made it difficult to train models to classify comments as toxic, opposed to classifying all samples as clean. During the project we thought of training several classifiers for each class, or equivalently having a classifier to first distinguish between toxic and non-toxic comments, and thereafter use a second classifier to classify the toxic comments. However, due to time constraints we could not pursue these options.

To give the reader an example of the dataset, we've included the following comment that is classified as toxic, obscene and as an insult:

'Groin You sad fuck. I'm going to restore my edit, and you are going to leave it, or else I wil knee you in the groin yourself, prick. 91.108.208.101'

Data Preprocessing

The data consists of raw Wikipedia comments, so some pre-processing is necessary before using them for training purposes. These preprocessing steps include tokenizing sentences and subsequently converting individual words to lowercase. We decided not to correct misspelled words as these mistakes themselves may be a sign of toxicity. This, for example, can be used for creating hand-crafted features.

4 Models

Several approaches were taken, starting with feature extraction, followed by Neural Network approaches.

Method	AUC
Feature Based Approach v1	0.60
Feature Based Approach v2	0.56
Convolutional Neural Network	0.49
Vanilla LSTM	0.52
Bidirectional LSTM	0.93

Table 1: Summary of the achieved results, the Area Under the Curve (AUC) scores are computed using Kaggle.

4.1 Feature Extraction

Our first approach included manual feature extraction. The features were handcrafted (i.e. not generated by a Neural Network).

4.1.1 Feature Based Approach v1

In the first iteration of this approach, more generic features were used:

- Ratio of capitals vs total characters
- Ratio of punctuation characters
- Total length in characters, words and in sentences
- Total amount of some special characters: ?, (,), ! and some other characters.
- Ration and amounts of unique words

In total, ten features were generated. These features were used to train various classifiers, which will be described in section 4.1.3. Each model was evaluated separately. However, as may be noted from the results but in the end, none of these feature based models managed to produce convincing results.

4.1.2 Feature Based Approach v2

Since we are trying to predict 6 classes separately, and we are using a complex set, the dimensionality of the set is probably higher than 10. That is why we decided to introduce some extra features.

- For a list of swear words (since we are doing *toxic* comment classification), we added a feature denoting whether that particular word occurred in the comment. This list included a list of common abuse and spam words, as well as common function words.
- A sentence2vec score comment is calculated by taking the average word2vec scores of every word in the sentence, normalised by the tf-idf score of each word.

These features increased count from 10 to 80+ but only gave a marginal improvement to the mean-averaged area under the curve (AUC) score, as may be observed in the results (table 1).

4.1.3 Feature-based Classifiers

Multilayer Perceptrons The multilater perceptron (MLP) is a basic neural network, using only fully connected (or dense) layers. In our case, the input consisted of the total number of features, and the output of the six classes as the last layer, each denoting the probability for one of the six classes. For the MLP, we experimented with various configurations of this setup (i.e. the number of layers and the width of the hidden layers).
Support Vector Machines We only used a linear kernel, but the learning time of the SVM was so high, that we quickly decided not to investigate this approach further.
Random Forests The random forest classifier also didn't get good results on the small feature set, and we decided to abandon this algorithm.
1D Convolutional Network From literature we found convolutional networks to be a good choice for these type of tasks. since the convolutions in essence

Another approach we tried is the 1-dimensional convolutional network. However, because this network showed weak results (merely 0.4902 ROC AUC score), so we decided to drop this approach.

4.2 Neural Networks

After a small literature study we found a LSTM (Long Short Term Memory) to be the, theoretical, optimal method in terms of neural network approaches. Because LSTM's are recurrent neural networks, they can learn temporal information, such as context of words in sentences [1]. First, we implemented a basic LSTM with a custom reverse dictionary. For this approach, we first tried focusing on the top n percentage of common words, as we thought that common words would carry the most information about the content of the comment and as a way the limit the computational complexity. However, after testing on the training set, we noticed that the LSTM performed better when using all of the words. And as such, we abandoned this method.

After testing the regular LSTM network, we found literature about a bidirectional LSTM [1]. A bidirectional LSTM feeds the sentence twice to the LSTM, once from front to back, and once mirrored (figure 2). This model performed better than aforementioned models (table 1), so we continued optimizing it.

```
A forward and backward sentence
sentence backward and forward A
```

Figure 2: An example of how the Bidirectional layer would feed the data to the LSTM

5 Reflection

This section briefly discusses some things that stood out during this project.

5.1 Imbalanced dataset

The one thing that stood out from our dataset is that it was very imbalanced. This made it hard to test if our model was performing correctly as predicting that every comment was non-toxic already resulted in an accuracy of roughly 0.96. This made it hard to train our models as during training time the model quickly returned a small loss.

5.2 Ensemble methods

During the first competition, we found out that ensemble models are extremely powerful for machine learning tasks (during the project presentations, some groups also reported these findings). We have decided that we also want to use ensemble methods in the next competition. For this competition however, we did not have the correct models to have an efficient ensemble.

5.3 Collaboration

After this first project we are more accustomed to each of our group members (i.e. knowing each other's strengths/weaknesses). This makes it easier for us to collaborate together and enables us to divide our tasks accordingly. To efficiently work together, we're going to utilize an existing git strategy, such as git flow. Also, we found great benefit in the weekly meetings and thus we're going to continue doing so.

References

[1] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.