Toxic Comment Classification - Report

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

March 28, 2018

1 Problem statement

We have chosen the toxic comment classification challenge on Kaggle. For this challenge we have to classify about 160000 comment. There are 6 different classes and each comment can be labeled with any of these classes. This means our problem is a multilabel classification problem.

2 Dataset

The dataset that is provided by Kaggle consists of some 160000 comments with their respective class labels. They also provide a test set of about the same size, without the labels. Our task is to predict the labels of the test set.

One important characteristic of our data is that the set is very imbalanced. This posed many challenges to us.

3 Feature based models

3.1 Feature Extraction

Our first approach was to look at the data and try to extract some features by hand. These features include things like:

- 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.
- Amount of unique words

In total we produced about ten features.

We used these features to train various other models on. We will evaluate each model seperately, but in the end, none of these feature based models managed to produce convincing results.

3.2 Models

3.3 Feature Extraction Part 2

After a meeting with our supervisor, we thought that a problem with our feature extraction was that we might be using to little features. Since we are trying to predict 6 classes seperately, and we are using a quite 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.
- More features??

4 Neural Networks

Having implemented these feature based approaches, we decided it might be better to use a neural network model. In particular, we decided to use the LSTM (Long Short Term Memory), since LSTM's are recurrent neural networks, so it can learn the context of words in sentences. First we tried to use the most simple LSTM we could think of.

After that, we found that on Kaggle there was a very well performing LSTM approach called the bidirectional LSTM. This means that we feed the

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

Figure 1: An example of how the Bidirectional layer would feed the data to the LSTM

sentence twice to the LSTM, once normally, from front to back, and once flipped.

We implemented two different LSTM.

We implemented a 1 dimensional convolutional neural network.

5 Validation