Toxic Comment Classification - Report

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

April 17, 2018

1 Problem statement

We have chosen the toxic comment classification challenge on Kaggle. This challenge consisted of a dataset of around ∼160 thousand wikipedia edit comments which had to be classified given six different classes. These classes regard the level of abuse in each comment. Moreover, the classes are not mutually exclusive, which means our problem is a multilabel classification problem.

Data Preprocessing

The data consists of raw Wikipedia edit 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 and correct erroneously spelled words. Though naturally, different algorithms might require different data. While misspelled words might not be good input for some neural networks, the occurrence of misspellings might be a good hand-crafted feature.

2 Dataset

Both the training and test set provided by Kaggle consist of about 160 thousand comments, a comment id and for the training set, the respective class labels.

One important characteristic of our data is that the set is very imbalanced (figure **??**). This posed many challenges to us.

3 Models

Several approaches were taken, starting with feature extraction, followed by Neural Network approaches including language modeling with an LSTM and a convolutional network.

3.1 Feature Extraction

Our first approach included mahual feature extraction. The features were handcrafted, such that they are tangible features (i.e. not generated by a Neural Network).

3.1.1 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, about ten features were generated. These features were used to train various classifiers, which will be described in section 3.1.3. 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.

3.1.2 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 concluded a list of common abuse and spam words, as well as common function words.
- A word-2-vec score per comment is calculated by taking the average word2vec scores of every word in the sentence, normalised by the tf-idf score of each word.

Tweet	**Class**
Who are you tomorrow? Will you make me smile or just bring me sorrow? #HottieOfTheWeek Demi Lovato	0,0,0,0,0,0
The maestro ... the legend Roger Federer king of the backhand game one of his best shots	0,0,0,0,0,0

Table 1: Example Wikipedia edit comments from the Kaggle Toxic Comment classification challenge

Method	AUC
Feature Based Approach v1	0.60
Feature Based Approach v2	0.53
Convolutional Neural Network	0.4902
Vanilla LSTM	0.52
Bidirectional LSTM	0.9256

Table 2: Summary of the achieved results, the Area Under the Curve (AUC) are computed using Kaggle.

These features increased features from 10 to 80+ but only gave a marginal improvement to the mean-averaged area under the curve (AUC) score.

3.1.3 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 6 classis as last layer, each denoting the probability that that one of the six respective was active for the current sample.

We experimented with various configurations of this setup. We varied 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, we have not YET tested it on the big feature set.

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

3.2 Neural Networks

After a small literature study we found a LSTM (Long Short Term Memory) to be the appropriate method in terms of a neural network approach. Because LSTM's are recurrent neural networks, it can learn temporal information, or the context of words in sentences. First we implemented a basic LSTM with a custom reverse dictionary. For this LSTM algorithm we first tried focusing on a top n percentage of common words as we thought that very common words might be influencing the model too much. However, after testing it 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 that, we found a Kaggle kernel for a bidirectional LSTM. This means that we feed the sentence twice to the LSTM, once normally, from front to back, and once flipped. This mode performed better than aforementioned models, so we continued optimizing it.

4 Conclusion Reflection

Here we briefly discuss some things that stood out during this project.

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 ≈ 0.96. This made it hard to train our models as during training time the model quickly returned a small loss.

Ensemble Methods During the first competition, we found out that ensemble models are extremely powerful for machine learning tasks (e.g. during the project presentations some groups used these). 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.

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 work 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] Jagan, *Stop the S@#$ - Toxic Comments EDA*, https://www.kaggle.com/jagangupta/stop-the-s-toxic-comments-eda