Capstone Project Proposal

Pradosh Priyadarshan

Machine Learning Engineer Nanodegree

May 19, 2020

Toxic comment classifier

1. Domain Background

Discussion in online community has a lot of benefits - it allows to share opinions freely and anonymously, express yourself and exchange ideas with people all over the world. However, this also gives a chance for offencive behaviour like abusement or harassment to arise. It discourages people from using online platforms to share their ideas and forces communities to limit of turn off commenting option overall [1]. To help fight this problem, the fast, automatic way of identifying toxic comments is needed. In this capstone project I will present the solution for classifying if the comment is toxic by using machine learning models.

2. Problem Statement

This is a multilabel classification problem. For each comment in training dataset the algorithm should automatically predict which of the 6 toxicity classes the comment should be assigned to. Comment can be assigned to none, one or more than one class. If comment is not assigned to any of the classes, that means that the comment is predicted as not toxic. Toxicity classes are defined in section 3 and are encoded as one-hot encoded vector, thus the output for each of the comments should be a 1x6 vector with predicted score for each class. Predictions should be normalized to be between 0 and 1.

3. Datasets and inputs

For this task I will use dataset made from wikipedia comments. They have been checked by humans and labeled by categories:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

Comment can belong to multiple classes.

Comments in training dataset has highly uneven classes distribution (Figure 1.), this imbalance will be addressed when training the models.

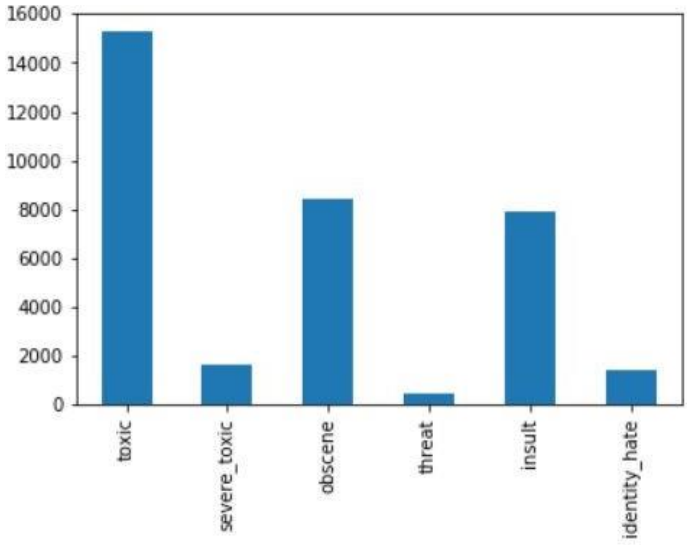

Figure 1. Number of comments assigned to each class in training dataset

As provided comments were not processed beforehand, text preprocessing such as removing non ASCII characters and stopwords, lowercasing, tokenizing will be required.

Dataset was provided in Kaggle's "Toxic Comment Classification Challenge" competition [2]. It is already split into training (~160k comments), and testing (~153k comments) files. Labels are provided in binary form for each class.

Dataset is under CC0 [3], with the underlying comment text being governed by Wikipedia's CC-SA-3.0 [4] license.

4. Solution statement

Provided solution will be an API that will be hosted in AWS. As an input it will take the comment and return the predicted value for each of the toxicity class. The predictions will be normalized to be between 0 and 1 for each class. The trained model will be evaluated on the test dataset. Using the evaluation metric as defined in section 6. the solution should reach the score of 0.95.

5. Benchmark Model

As this data was provided for Kaggle competition, the benchmark will be based on the evaluation metric that was used in the mentioned competition. Model should achieve the AUC score equal or higher than 0.95. The top approaches provided by the participants achieved higher scores, however this benchmark should be enough to get a reasonably well performing model, that could be further used in other applications. Models submitted to the competition and their performance can be viewed in leaderboards of the challenge [2].

6. Evaluation Metrics

To evaluate the model and compare to benchmark models I will use the same metrics that were used in the competition: “the average of the individual AUCs of each predicted column”.

7. Project Design

Project workflow:

1. Exploratory data analysis - reading comments, drawing and plotting insights. Checking common or correlated words in each class.
2. Data cleaning - removing unnecessary characters (non ASCII), stopwords, or texts that will be known after step 1
3. Choosing features and machine learning model - for the baseline model, I will start with a simple approach, that is based on word counts. Vectorize comments using TF-IDF and use it as an input vector to a simple 2 layer NN classifier with sigmoid activations in the output layer. Evaluate this model, check and plot the results to see where the model could be improved. If the task requires more complex solution, I would investigate word embeddings (word2vec, glove, bert), to better represent sentence meaning and make a classifier with these embeddings as an input.
4. Training and evaluating the model - evaluate the model and compare to benchmark.
5. If needed repeating previous steps until satisfactory model performance is achieved
6. Deploying the model
7. Exposing API

References

[1] https://www.wired.com/story/twitter-let-users-hide-replies-fight-toxic-comments/

[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview

[3] https://creativecommons.org/share-your-work/public-domain/cc0/

[4] https://creativecommons.org/licenses/by-sa/3.0/