Machine Learning Engineer Nanodegree Capstone Proposal

Domain Background

Toxicity is a common problem in online forums and comment sections. Many people have had the experience of opening the comment section of an article only to be met with insults, obscenity, and hatred. Extensive human moderation is one solution, but at large scale it's quite costly, and it can be a harrowing task citep:verge-2019-facebook-moderation. The ability to augment human moderation with machine learning models to filter out the worst and most obvious toxic comments is critical.

Problem Statement

This problem was originally posed in the Toxic Comment Classification Kaggle competition citep:kaggle-toxic-comments. The objective is to build a model that is able to classify comments as containing different types of toxicity like threats, obscenity, insults, and identity-based hate.

Datasets and Inputs

The data originally set comes from Wikipedia comments which have been labeled by human reviewers. Each comment has 6 binary labels, one per type of toxicity: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, and `identity_hate`. The training set contains 159,571 labeled examples; the test set contains 153,164 examples, of which 63,978 have labels and 89,186 do not. Because this project isn't going to be submitted to Kaggle, the 89k without labels will not be used for this project, and the remaining examples will be combined into a single dataset of 223,549 examples which will be shuffled and split into 80/20 train/test.

The final dataset contains comments ranging from 1 to 5,000 characters in length, with an average of 391 characters. Most of the comments are fairly short; only 8.2% are over 1,000 characters long. The classes are also highly

imbalanced: 9.6% of comments are labeled `toxic`, 0.88% `severe_toxic`, 5.4% `obscene`, 0.31% `threat`, 5.1% `insult`, and 0.95% `identity_hate`. 90% of the examples are not toxic at all.

Solution Statement

The solution is to create a machine learning model that takes in the text of a comment and outputs a score for each of the toxicity labels, denoting the probability that the comment falls under one of those categories of toxicity. The scores would be used to flag comments for manual inspection by human moderators.

Benchmark Model

Jigsaw/Conversation AI have their own model citep:jigsaw-model-project, a convolutional neural network trained with GloVe word embeddings. Their model achieved an ROC AUC of 0.96857.

Evaluation Metrics

The main evaluation metric will be the mean column-wise ROC AUC:

$$\frac{1}{L}\sum_{l=1}^{L} AUC_l \tag{1}$$

where L is the number of labels (in this case 6). Additionally, the AUC for each individual label will be analyzed; it's likely that certain labels will be easier to classify than others.

Project Design

The project will start off with some basic pre-processing of the text: just splitting the comments into individual words. Then some analysis will be

done on the correlations of words to particular labels. In addition, I hypothesize that certain types of toxicity, like insults, will be more likely to have lots of special characters (e.g., exclamation points), repeated letters in words (e.g., the f-word but with several "u"s), or excessive capitalization. I will do some analysis on this, and if any of those appear to be correlated with the labels, I will include them as features somehow.

After deciding on the "special character" features, I will continue with a more complete pre-processing step. This will include removing capitalization and special characters, then splitting into words and stemming. The output of the text pre-processing will be converted into a bag of words encoding, possibly concatenated with the special character features identified in the previous step. Some analysis will be done on the size of the vocabulary. Ideally the vocabulary will be as small as possible, but it's possible that certain long-tail words will be important in identifying toxicity. I will look at some of the long-tail words to see whether an additional regex matching step to group misspellings, repeated letters, or other types of differences would be useful. Finally, at this point the dataset will be split into 80/20 train/test.

For the model itself, the bag of words vector will be fed first into an embedding step, then into an LSTM, then finally into a 6 element sigmoid activation function to get the class probabilities (remember that the classes are not mutually exclusive, so softmax is not appropriate here). Depending on the performance of this model, I may decide to use pre-trained embeddings in place of the first step.

References