Machine Learning Engineer Nanodegree

Capstone Proposal

Yoong Kang Lim August 1, 2018

Proposal

The proposed project is the Jigsaw Toxic Comment Classification challenge on Kaggle [1].

Domain Background

There is a lot of discourse on the internet, and it is one of the things that keep people engaged on the internet. Comments can be found on various content such as news articles, blog posts, and forums. However, there can be instances of abuse and harrassment on these platforms. The threat of this may lead to people becoming less inclined to express their opinions on the internet [2].

Manual moderation of comments – one done by a human being – often works when the number of comments is small. Another approach is "crowd sourcing", which can be seen in platforms like Reddit which have upvotes and downvotes, where downvoted comments are hidden or deleted. However, this has some problems. With more and more people engaging in discussions on the internet, this can be too much for human beings to manage. As a result, more work is being done to automatically detect and classify toxic comments using machine learning [3].

This project is an attempt to improve upon such machine learning models, particularly those provided by the Perspective API [4]. While these models are useful, they are not perfect and are subject to several flaws. One of the flaws is that an intentional misspelling can affect the toxicity score significantly [5]. It also does not allow users to find the "type" of toxicity they are interested in. For example, some platforms may allow profanity but a binary classification system will mark a comment with profanity as a toxic comment. On the other hand, a comment without profanity may look civil on the surface, but is just as harmful, (e.g "I feel sorry for your children, as they are doomed to live a life raised by a mother with clearly deficient mental capacity").

Problem Statement

The goal of the project is to predict the probability that a comment falls under a set of "toxicity" classes. The classes are `toxic`, `severe_toxic`, `obscene`, `threat`, and `identity_hate`. The model should tell us the probability that it falls under

each class as a value between 0.0 and 1.0. For example it can be 0.6 toxic and 0.2 threat, etc.

Datasets and Inputs

The datasets are provided by Jigsaw on the Kaggle competition website [1].

The input is in CSV format with the following columns:

```
"id",
"comment_text",
"toxic",
"severe_toxic",
"obscene",
"threat",
"insult",
"identity_hate"
```

Jigsaw and Kaggle have also provided the test set. As this is a completed competition, they have also provided one with labels, so that submissions are unnecessary to find out the score.

Solution Statement

The output will be in the format expected by Kaggle, which is a CSV file with the following columns:

```
"id",
"toxic",
"severe_toxic",
"obscene",
"threat",
"insult",
"identity_hate"
```

Note that this is the same as the training input, except without `comment_text`.

Apart from the column `id`, each column should contain the model's prediction that it falls under this class. This will be a value between 0.0 and 1.0, where the higher value means more likely. Thus, this is a "multi-class" regression problem.

The solution is quantifiable by taking the difference between these probabilities with the ground truth. This score will be evaluated under the metric that will be described later in this proposal.

This will be provided as Jupyter notebook(s), so that it is replicable.

Benchmark Model

I plan to use a number of benchmark models to compare my model with. Firstly, we will look at "weak" baselines:

- Randomly generated probabilities
- `0.5` for each column (equal probability for all)
- Naive Bayes Classifier (without ngrams)

I will also use one strong baseline. This is the top voted kernel on Kaggle for this competition:

- NB-SVM strong baseline (https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline)

Evaluation Metrics

The evaluation metric will be the same as the Kaggle competition's metric, which is mean column-wise ROC (receiver operating characteristic) AUC (area-under-curve). In other words, the score is the average of the individual AUCs of each predicted column.

Project Design

These are the phases that I will use to find a solution for the given problem:

Exploratory data analysis

I will look at the data to look at what the training input looks like. I will identify things that require some data cleaning or feature engineering. For example, it was not specified if all the comments are in English – and if there are Japanese comments in an encoding like Shift-JIS, I will need to deal with this.

I will also collect some simple statistics, like obtaining the most common words associated with a certain class, and to see if there is any overlap. This is to see which words are important for analysis, which can help inform what feature engineering I do.

Preprocessing and feature engineering

It is likely that this project will need feature engineering. Comments are likely not of the same length, so I will need to find a way to make that matter less. For example, a naive way would be to simply repeat the comment until it is the same size as the longest comment. I will take a small sample to see if this improves predictions.

Some other things I will explore is to use TF-IDF, as well as ngrams. Another thing I might do is to preprocess the comments to fix simple spelling mistakes.

The EDA phase should inform what feature engineering might be required.

Model and hyperparameter selection

Kaggle has given us a training set and a test set. I will break the training set into training + validation sets. This is to tune hyperparameters. If needed, I will split the training set into three (training + validation + test), which will avoid accidentally overfitting to the test set (e.g by submitting and refining the model).

As we seem to have enough data, I might skip cross validation techniques, but I am not ruling it out right now.

The models I will look at will be NB-SVM (as described in the Benchmark Model section) with my own improvements, as well as deep neural networks.

Model refinement

I will make further refinements to the model. For example, for deep neural networks I may experiment with dropout, early stopping, LSTM, and various activation functions.

Report

I will then describe all my steps in a report and provide Jupyter notebook(s) that document the steps I took.

References

[1] Kaggle, "Jigsaw toxic comment classification challenge.".

[2] M. Duggan, *Online harassment*. Pew Research Center, 2014.

[3] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," *CoRR*, vol. abs/1610.08914, 2016 [Online]. Available: http://arxiv.org/abs/1610.08914

[4] "Perspective api.".

[5] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, "Deceiving google's perspective API built for detecting toxic comments," *CoRR*, vol. abs/1702.08138, 2017 [Online]. Available: http://arxiv.org/abs/1702.08138