project_proposal

April 30, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Stephen O'Kennedy
25 April, 2018

1.2 Proposal: Identify and classify toxic online comments

1.2.1 Domain Background

Since it's inception the internet has allowed people from most parts of the world to freely communicate, debate, and collaborate with each other over a wide range of topics and projects. Platforms like Github, Hackernews, Twitter, Wikipedia, etc. form the foundations for which these interactions can take place. Many of these communities have standards and rules in place to facilitate conversations, and to prevent these communities from being hijacked, or destroyed by toxic behaviour. It is becoming increasingly harder to regulate and enforce these standards. In fact Facebook are currently hiring more and more moderators to sift through questionable content [1].

Conversation AI are working to provide tools to help improve online conversation[2]. One area that they're focusing on is the study of negative online behaviours, like toxic comments[2]. As their Kaggle page states, the current models in use for detecting toxic comments still make errors, and they don't allow users to be able to identify the types of toxicity they're interested in finding. For example some platforms may be fine with comments that contain profanities. Ultimately we want to define a model that can perfrom sentimental analysis on wikipedia comments.

As we can see from Pak's (Pak et al 2010) work there is a significant body of research into sentimental analysis. Thier work focused on sentiment analysis which is fairly similar domain to what we'll be working on. Pak (Pak et al 2010) defined a sound methodology for pre processing text using the bag of words technique. They proceeded to use both SVM and Naive Bayes classifiers to identify if the sentiment of a tweet to was `Postive`, `Neutral`, or `Negative`, and found that Naive Bayes classifier was more accurate. Furthermore, they were abel to boost thier classifier's accuracy by using bigrams when they vectorised their tweets. It would appear that their work could be used as the basis for a baseline model for evaultions [7].

1.2.2 Problem Statement

Given a dataset that contains a large number of Wikipedia comments which have been labelled by human raters for toxic behaviour. We want to create a model that predicts the probability of different types of toxicity for each comment.

1.2.3 Datasets and Inputs

We are provided with a dataset in csv format where we have the following columns:

- `id`
- `comment_text` : [String]
- `toxic` : int
- `severe_toxic` : int
- `obscene` : int
- `threat` : int
- `insult` : int
- `identity_hate` : int

The `comment_text` column is comment that we want to feed into our classifier, and the outputs will be will be a vector containing the probabilities of the comment being one of the `toxic`, `obscene`, etc. The training data set can be found here [3].

The dataset is comprised of two files: - train.csv - test.csv

We will use the data contained in the train.csv as both our training and validation datasets. The main reason for doing this is because the data contained in test.csv is unlabelled, and therefore we'll be unable to validate our model.

Below are some details about the data in train.csv

```
In [32]: import pandas as pd

         df = pd.read_csv("./train.csv")

         data_shape = df.shape
         print("Shape of data: {}".format(data_shape))

         target_labels = df.columns[2:]

         for x in target_labels:
             print("Percentage of {} comments: {:.2%}".format(x, df[df[x] == 1].shape[0]/data_sh
```

```
Shape of data: (159571, 8)
Percentage of toxic comments: 9.58%
Percentage of severe_toxic comments: 1.00%
Percentage of obscene comments: 5.29%
Percentage of threat comments: 0.30%
Percentage of insult comments: 4.94%
Percentage of identity_hate comments: 0.88%
```

As we can see from the above output we have 159,571 rows in our dataset. The dataset is fairly unablanced across the board. Most of the comments in the dataset do not qualify as being toxic.

1.2.4 Solution Statement

The solution will involved the development of deep learning algorithm that uses Keras with TensorFlow being used as the backend. Our aim is to use a multi-class CNN to process the content of the comments and out put a ROC AUC score [4]. Finally, predictions will be made on the test data set and will be evaluated on Kaggle.

1.2.5 Benchmark Model

We aim to have several benchmark models, to evaulate with an additional stretch where we'll compare our model on kaggle. Our initial benchmark model will be to use the bag of words technique and the multinomal naive bayes to output a vector containing the probabilities of the classification by using the `predict_proba` method. This method is defined in sklearn's implementation Multinomial Naive Bayes [8]. The next benchmark we'll have is to use a fully connected MLP and attempt to surpass the previous benchmark. After this we'll use a CNN to again try and surpass the other benchmarks.

Our next batch of benchmarks will be to replace the bag of words pre processing with the word-2-vector approach and use the previously described models.

Our stretch benchmark score will be to achieve a score of `0.982900`. This score was is in the 50th percentile of the public leaderboards[5], and was calculated using ROC AUC metric [4].

1.2.6 Evaluation Metrics

Submissions are evaluated by using the ROC AUC metric. Each comment in the test data set will need to be labeled with the predictions for each type of toxicity appearing in each comment, and will need to be submitted to Kaggle.

ROC is the receiver operating characteristic curve. It is a graphical plot that displays the discrimination threshold of a binary classier, which is what we'll need to build. Our threshold (T), which is used to classify a datapoint as either positive or negative, is by default set 0.5. We take the true positive rate (TPR) and false positive rate (FPR) for all scores and plot a curve. Calculating the AUC (area under the curve) will reduce the curve down to a single value, $1 \geq A < 0$. Where A is the AUC. If A is close to 1.0 we've got a perfect classifier, if However it is 0.5 of lower than our classifier is doing little more than guessing [6].

The formula is:

$$A = \int_{\infty}^{-\infty} TPR(T)FPR'(T)dT = \int_{-\infty}^{\infty}\int_{\infty}^{-\infty} I(T' > T)f_1(T')f_0(T)dT'dT = P(X_1 > X_0)$$

1.2.7 Project Design

As described in the problem statement, we will be analysing comments made on wikipedia. We will first need to perform data analysis and get familiar with the data. There are several questions that come to mind immediately that I want to understand about the data. We may want to look at the most common words found in each class of toxic comments. It may be worth looking to see if some of the comments that are deemed toxic have a strong correlation with each other. There are questions wa may want to answer as such as, are the classes balanced? Is the data set complete? Do some of the comments use non standard characters?

Once we've completed our data analysis we will begin text processing. There are two approaches we'll want to purse when performing sentimental analysis. The first will be the bag of words technique, this is a tried and tested technique for processing text. It'll allow us to process our data for the first round of benchmarks. The benchmark models we'll use will be Multinominal Naive Bayes, SVM, etc. We'll then create a fully connected MLP usng keras with a tensorflow backend, and then a CNN.

The next approach we'll want to do is utilise Google's word-2-vector algorithm to help us cluster words that are similar [9]. Our aim is to use a GloVe, which is an unsupervised learning algorithm for obtaining vector representations for words. There are several pre-trained models to choose from and we'll look to use [10]. What we'll aim to do is to load the vectors from the pre-trained model where can can use the tf-idf weighting scheme and train the previously mentioned algorithms [11].

After we've established our benchmarks we'll begin building our final model using transfer learning on a pretrained model. Our aim is to create a RNN or CNN that uses Google's DRAGNN or Syntaxnet as our foundational model [12].

1.3 References

- [1] http://fortune.com/2018/03/22/human-moderators-facebook-youtube-twitter/
- [2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
- [3] https://www.kaggle.com/c/8076/download/train.csv.zip
- [4] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
- [5] https://www.kaggle.com/c/8076/publicleaderboarddata.zip
- [6] https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve
- [7] http://crowdsourcing-class.org/assignments/downloads/pak-paroubek.pdf
- [8] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.
- [9] https://code.google.com/archive/p/word2vec/
- [10] https://nlp.stanford.edu/projects/glove/
- [11] http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/
- [12] https://github.com/tensorflow/models/tree/master/research/syntaxnet