Machine Learning Engineer Nanodegree
Capstone Proposal

Yiyan CHEN
June 21th, 2018

Toxic Comment Classification Challenge

Domain Background

The topic comes from a kaggle competition [1]. The Conversation AI team, a research initiative founded by Jigsaw and Google (both a part of Alphabet) are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). So far they've built a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don't allow users to select which types of toxicity they're interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).

I have been working on natural language processing before taking the udacity course. When I was learning CNN, I came up with an idea that using CNN for natural language processing can explore the hierarchical structure of the sentences. After making some research, I found that CNN has already been used to make text classification[5]. So I want to try this idea in my capstone project by solving a real world problem proposed by one of the recent kaggle competitions.

Problem Statement

The problem can be seen as a multi-labeled text classification problem. The goal is to build a multi-headed model that's capable of detecting the existence of different types of toxicity in a given online comments like threats, obscenity, insults, and identity-based hate. We can use accuracy or the mean column-wise ROC-AUC on the testing set to measure the performance of the model. The nature of the problem is multi-label classification, linear classifiers do not share parameters among features and classes, especially in a multi-label setting [2]. So I would prefer the deep neural network to be a potential solution for this problem and it will be interesting to compare CNN with linear models that predict each label independently.

Datasets and Inputs

We are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

The only feature we have is the text comment and there are 6 targeted binary variables. The median of the length of one comment is around 200 chars and the mean is around 400 chars. For the toxic label, there are 144277 non toxic comments and 15294 toxic comments in the training set, so the distribution is not balanced and this is true for all 5 other labels. Finally, there are 153164 comments to be labeled in the test set.

File descriptions [6]
train.csv - the training set, contains comments(text and ID) with their binary labels
test.csv - the test set, you must predict the toxicity probabilities for these comments.
test_labels.csv - labels for the test data; value of -1 indicates it was not used for scoring;

Solution Statement

I will use word embedding such as google word2vec to transform the words in the comments into vectors, so the entire comment will looks like a 2D matrix. then I will design, train and tune a CNN model on the transformed data, the output layer of the CNN will return independently 6 values between 0 and 1, each describes the probability of the presence of a toxic type in the input comment. During the validation step, I can use the average of the binary cross entropy loss to measure the CNN's performance. For the testing, I will use the mean column-wise ROC AUC as required by the Kaggle platform.

Benchmark Model

Since this problem comes from an Kaggle competition, many benchmark models will be available. The best model on the leaderboard gains a ROC-AUC score of 98.85 %. As a beginner, my goal is not to beat the best score but to explore the advantages of the CNN architecture for multi-label text classification tasks. Thus, another interesting benchmark model is more interesting for me, this model predicts each label separately using logistic regression. The score is surprisingly not bad: 97.88% [3]. Given this context, it will be more interesting to see how much improvement we can make by sharing parameters for different labels using DNN.

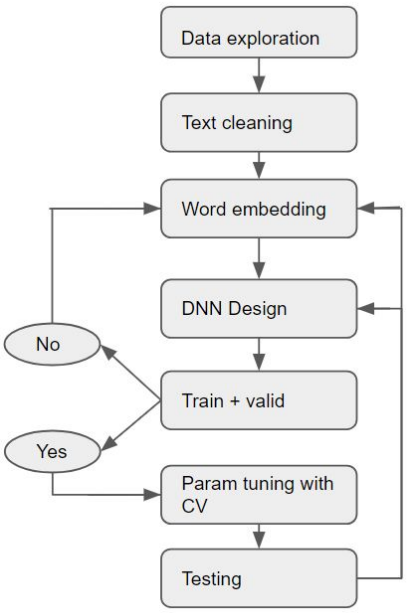
Evaluation Metrics

In the submission file, we need to give a probability for the presence of each of the 6 types of toxicity in each test comment. So the input of the predictor will be a raw text comment. The output of the predictor will be a vector of 6 dimensions. One dimension is the probability of the comment having a toxicity type. The overall Accuracy of each single label binary prediction(if >0.5 : 1 else 0) is not fitted for this model, because of the unbalancy of each class. For example of the "toxic" label, if we predict all are not toxic, we can get an accuracy of 90%.

I can think of two possible evaluation metrics:
1. The mean column-wise F2 score giving more importance on recall
2. As proposed by kaggle: the mean column-wise ROC-AUC
To better understand the metric 2, a column is a toxicity type, the predictor will associate each positive and negative sample with a score(probability of being positive), then the column-wise ROC-AUC is that if we randomly select a positive and a negative sample, the probability of the former having lower score than the later:

$$P\Big(\text{score}(x^+) > \text{score}(x^-)\Big)$$

Project Design



1.Data exploration

2.Text cleaning:
First, I will keep the same process here as the benchmarked model does. For later improvement, I may try to reduce the large comments composed by one repeated expression.

3.Word embedding
I will try out some of the following technique: tf-idf, GloVe, word2vec, fasttext... trained with other dataset or only on the toxic data set.

4.Design a basic DNN according to [4]

5.Split the training set into validation set and train set, improve the validation score by improving the structure of the DNN designed in step 4 by:

- Adding more convolutional layers
- Adding dropout and batch normalization
- Adding attention mechanism

- Using a pre-trained CNN with it's pre-trained embedding layer ...

6.Once the final DNN's structure is determined, Use K-fold cross-validation and automatic hyperparameter tuning to find the best parameters that maximize the CV score.

7.Test the resulted model on testing set, compare the result with benchmark models.

References:

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
[2] https://www.depends-on-the-definition.com/classify-toxic-comments-on-wikipedia/
[3] https://www.kaggle.com/thousandvoices/logistic-regression-with-words-and-char-n-grams
[4]http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/
[5]https://www.researchgate.net/profile/Manikandan_Ravikiran/publication/325451936_Toxic_Comment_Classification-An_Empirical_Study/links/5b0ec999aca2725783f3f72f/Toxic-Comment-Classification-An-Empirical-Study.pdf
[6] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data