Machine Learning Engineer Nanodegree

Capstone Proposal - Toxic Comments Classification

Hui Wang January 25, 2018

Proposal

Domain Background

Natural language processing (NLP) provides methods and tools to analyze, understand and derive meanings from human language. Some well-known use cases for NLP are: automatic summarization of speeches or texts, translation, sentiment analysis, speech recognition, topic segmentation, and etc. NLP is an active research area, and the development in this area is closely related to people's life. For example, a chat bot based on Point-of-Speech tagging can provide customer service at 24/7; a summarizer can extract the most important and essential ideas of text blocks while ignoring irrelavant information; a text-reader application that converts text blocks to audible speech can make it more convenience for people with poor vision. The impact on daily life is also my primary motivation of choosing a specific topic in this area as my capstone project.

The problem I will try to solve in this project is to classify toxic comments into different categories (a Kaggle dataset). Online conversation can be difficult. Many people stop trying to express themselves and give up on seeking different opinions, due to the threat of abuse and harassment online. It has been a request from online platforms to host a healthy environment to foster and encourage people to discuss the topics that they care about. There exist models that identify toxic comments, however, the models still make erorrs, and they don't allow users to specifically select the categories of toxic comments they want to find.

Problem Statement

A training dataset of comments from Wikipedia’s talk page edits will be used. The target labels are severity of toxic, and specific categories: threats, obscenity, insults, and identity-based hate. The task is to predict the toxic severity of a comment and the probability for a given comment to fall into each of the toxic categories. The performance of the model will be evaluated by the average column-wise log loss, where each column represents a severity or a category, and each row represents a comment.

Datasets and Inputs

The input includes the following columns:
"id","comment_text","toxic","severe_toxic","obscene","threat","insult","identity_hate", where the "id" column represents the unique identifier of the comment; the "comment_text" is the content of the comment, in its original format, not processed, and this column will be the primary input of the model; the columns "toxic" and "severe_toxic" indicate differnet toxix severities; and the rest of columns are the labels that the model will be trying to predict on, where 1 indicates the comment is

in this category, 0 otherwise. Note, the categories may or may not be mutually exclusive, i.e., a comment can be classified in more than one category. This needs further clafification.

A sample input row (with header) is shown as below:
"id","comment_text","toxic","severe_toxic","obscene","threat","insult","identity_hate"
"0000997932d777bf","Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27",0,0,0,0,0,0 "000103f0d9cfb60f","D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)",0,0,0,0,0,0i

The dataset has 159571 rows, representing unique comments. However, it does not have balanced classes, specifically, 9.58% records were labeled as "toxic", 0.10% as "severe_toxic", 5.29% as "obscene", 0.30% as "threat", 4.94% as "insult", 0.88% as "identity_hate". Techniques dealing with inbalanced classes, such as downsampling, specifying class weights, should be considered to use.

Solution Statement

I plan to build a multiclass classification model or a combination of several binary classification models (depending on how overlapping the target labels are) with features directly or engineered from the "comment_text" column. There are several tyeps of model that provide multiclass prediction. Some come into my consideration are Decision Trees, Logistic Regression, Convolutional Neural Network, and Recurrent Neural Network. There are also several ways to represent the text in the column of "comment_text", such as word-of-bag(WOB), word vectors, term frequency–inverse document frequency (TFIDF). During the project, I will experiment several combinations of model and text representation, if time allows, and select the best solution.

Benchmark Model

I plan to use a Naive Bayes model with bag-of-words representation as my benchmark model. This model itself is straightforward and easy to understand, and usuualy works descently well for text analysis. The benchmark model's performance will be measured in the same way of using average log loss on a reserved test set.

Evaluation Metrics

The output of the predictive model will be like:

id,toxic,severe_toxic,obscene,threat,insult,identity_hate 00001cee341fdb12,0.5,0.5,0.5,0.5,0.5,0.5 ...

The evaluation metric is the average column-wise log loss, which is a metric on measuring the similarity of predicted labels and true labels.

Project Design

1. Data processing - include closer examining data, understanding data, cleaning data if necessary, and etc. Consider using methods in NLTK and spacy libraries.

2. Feature engineering - brainstorm relavent features that are not directly provided in the input, such as punctuations, capital letters; build word vectors.
3. Model - choose a proper model for the multiclass classfication problem, fit the model, and make prediction.
4. Evaluate model - produce metric as stated in the Evaluation Metrics section.
5. Summarize - conclude the project.

Depending on the model performance, there might be a few iterations of a selection of step 1 to 4.

References and Citations

Kaggle competiiton: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
Algorithmia blog on natural language processing:
https://blog.algorithmia.com/introduction-natural-language-processing-nlp/ Log loss:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html