Machine Learning Engineer Nanodegree

Identify and classify toxic comments online

Kirupakaran Harikrishnan
August 9, 2018

Proposal

Domain Background

The idea of online forums and discussion sections is to promote meaningful conversations. Nowadays every website has a comment section and it is very difficult to deal with toxic comments. Every comment needs to be moderated which takes a lot of time. The emergence of massive social networking websites like Facebook and Reddit lead to millions of people posting content and commenting on them. People including children interact in the internet and there is a need to control the content and encourage a healthy environment and facilitate conversations, and ensure that they don't give up on stop expressing themselves. Many news websites now have started disabling comments sections.[1] Most of these websites have human moderators behind them monitoring them constantly which is very inefficient.

There is a need for tools to automatically monitor, categorize and flag comments. Also, different websites may need to control different kinds of content. There is a need to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate.

Personally, I participate in many Massive Multiplayer Online games like dota, csgo, pubg and there is a lot of obscenity, racism and identity based hate speech which turns down many new players.[2] We need better tools for such platforms where many children interact everyday.

I've taken up the Kaggle competition "Toxic comment classification" by Jigsaw[3] for the data and benchmarks. The kaggle competition is based on the Perspective models developed by Conversation AI team, a research initiative by Jigsaw and Google[4]. The predictive API can predict whether a given text is hate speech or not but can't classify them.

Problem Statement

The problem comes under Natural Language Processing, more specifically a sentiment analysis multi classification problem. A model will be developed that, given a text, can identify and classify it as toxic, severely toxic, obscene, threat, insult, identity hate with some probability. Note that, a comment may have high probability for all of the classification problems.

The Kaggle submission is evaluated on the average of the individual AUCs of each predicted column (type of toxicity).

Datasets and Inputs

The primary data is wikipedia talk comments provided by Kaggle labeled by human raters for toxic comments. The training data contains 160k rows and the test data contains 153k rows.

The training data contains the following columns:
Id, Comment_text, Toxic, Severe_toxic, Obscene, Threat, Insult, identity_hate

Comment_text is the text input and the columns following that are the output labels with probability values

The test will contain only the id, comment_text and the output will be the probabilities.

Additional data[5][6] will be used to augment the original data (if time permits).

Both of the above data will be partly used for testing the data. They contain labels but they don't have the prediction values. These datasets augment the original dataset because they come from a wider demographic section.

Solution Statement

The problem can be solved by using an algorithm that takes comment as input and outputs a list of probabilities whether it is toxic and the type of toxicity, i.e.,
Given a comment C -> algorithm -> r [toxic, Severe_toxic, Obscene, Threat, Insult, identity_hate]

Each result r is a probability ranging from (0, 1).

A simple algorithm that works reasonably well is Naive Bayes - Support Vector Machine which will be used as the baseline model.

Long Short Term Memory network[7], a form of Recurrent Neural Network algorithm is an algorithm that is designed for natural language processing and one that is proven to works well and that'll be the basis of the solution. The data will be preprocessed using word embeddings that turns text into a numerical vector representation that can be feeded to neural networks. As part of the solution, several word embedding approaches like Word2Vec, Glove, FastText and pretrained embeddings will be evaluated.

Other algorithms like Convolutional Neural Networks and Gated Recurrent Networks will also be evaluated (if time permits).

Benchmark Model

Support Vector Machines are one of the most used algorithms for text classification and that can be used as the Benchmark model. An algorithm based on SVM and Naive Bayes is SVMNB[8] which gives better performance than traditional SVM is the recommended benchmark in the kaggle competition and that will be used as the benchmark.

Evaluation Metrics

The Kaggle competition suggests uses Column wise average of the Area under the ROC curve for each predicted column and that'll be used as the evaluation metric. ROC curve is a graph plotted using True Positive Rate and False Positive Rate at different classification thresholds and Area Under the ROC (AOC) simplifies it by aggregating the performance for all thresholds.

The data will be divided into training, test and cross validation dataset in the ratio 50:30:20.

Also Bias, Variance, Precision, Recall and F1 score will also be used as evaluation metrics to check overfitting and underfitting.

Project Design

The project will follow the following steps:
Data Exploration
Data Preprocessing
Model design
Model evaluation

The initial step is to explore the data. Create visualizations to check the number of non-toxic comments and the number of toxic comments and the distribution of the different types of toxic comments. Also, create word cloud visualizations to understand the frequent words in each category. It's also important to understand the unique words, commonly occuring words, filler words, etc. which can help in preprocessing the data.

The next step is to preprocess the data and clean all the null values. During the preprocessing, all the unneeded data needs to be removed. This includes spam data that may have random letters or words, non-text data, usernames, etc.

Before the data is feeded to the network, the entire data needs to be parsed and tokenized into individual unique words and each word will be encoded with its index. Then each comment will be represented using the index values where each word is replaced with its index. Each of this index value will be a parameter or feature to the network. This is called word embedding and there are several word embedding approaches like word2vec, GloVe, FastText, etc., and there are pre trained word embeddings which will be evaluated.

I'm going to use Recurrent Neural Networks, a form of neural networks with feedbacks in them (memory) so they can deal with sequential data. Since we're dealing with text where every word depends on the other words, RNNs will be effective. Traditional RNN has a long-term dependency problem where it cannot remember long-term contexts where the gap between relevant information grows. To solve this problem, Long Short Term Memory networks[1] are created which are a special form of RNN that has LSTM cells which are 4 neural network layers and 3 gates (input, output and forget). These layers and gates helps the network to remember relevant information and forget irrelevant information.

The network requires fixed number of parameters. So once the tokenizing and indexing is done, feature length needs to be fixed which can be taken as the mean of the distribution of number of words and the input comments needs to be converted to consistent length data. Padding needs to be added to shorter comments and longer comments needs to be trimmed. Padding can be done by appending zeros. For longer comments, trimming should be done which will lead to some loss of information. Then the data will be passed to the LSTM network for training.

Once the training is done, data will be cross-validated using the cross validation set to tune various parameters and then tested using the test dataset and evaluated using the described evaluation metric (AUC) .

Keras, TensorFlow, Scikit learn will be primary python libraries that will be used for implementation.

References:

1. https://www.theguardian.com/science/brain-flapping/2014/sep/12/comment-sections-toxic-moderation
2. https://massivelyop.com/2015/07/23/massively-overthinking-dealing-with-toxicity-in-mmo-communities
3. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
4. https://www.perspectiveapi.com/
5. https://data.world/crowdflower/hate-speech-identification/workspace/project-summary
6. https://www.kaggle.com/datasnaek/youtube
7. https://www.researchgate.net/profile/Sepp_Hochreiter/publication/13853244_Long_Short-term_Memory/links/5700e75608aea6b7746a0624/Long-Short-term-Memory.pdf
8. https://nlp.stanford.edu/pubs/sidaw12_simple_sentiment.pdf