TOXIC COMMENT DETECTION AND CLASSIFICATION

Prinslou Tare
prinslou@stanford.edu

Motivation

In today's current society, there is a big problem when it comes to online toxicity. This toxicity tends to negatively impact how a lot of people tend to engage in conversation and deters some from engaging in online conversation entirely. As a result online platforms tend to struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

Problem Definition

The goal of this project is to detect and identify toxic comments using both machine learning models and deep learning models. This will help in deterring people from posting toxic comments and thus help to facilitate courteous discussion on online forums.

Features

The original dataset is the jigsaw toxic comment detection dataset that contains comments from Wikipedia's talk page edits. The only feature it includes is the word comment. The comment is then classified into one or more of six classes: toxic, severe_toxic, obscene, threat, insult, idenity_hate.

Feature Encoding

The is a huge variation in the length of the comments with the average length being around 400 words. In encoding the comments, I first got rid of all the empty comments and then encoded the remaining comments using the GloVe 60d model. In encoding I normalized sentence vectors as a whole and got rid of stopwords.

Models/Approaches

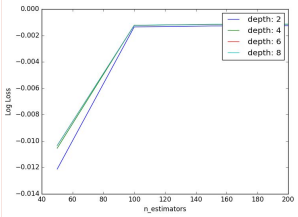
Training and Test Size
Training set: ~ 111699
Test set: ~ 47872

Support Vector Machine(SVM)
The SVM in this particular case made use of a Radial Basis Function kernel

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$$

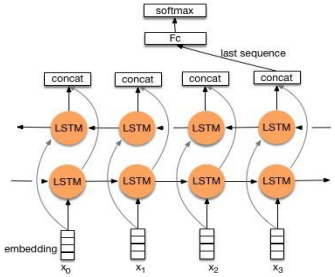
Logistic Regression
For logistic regression I settled for a linear solver and balanced class weights.

XGB Boost
For this implementation I decided to use early stopping after two rounds so as not to overfit the data



XGB Boost Loss Function

LSTM
The bidirectional LSTM was implemented using Keras with zero padding and made use of a maximum of 150 words in a comment and 30000 unique words .



Results/Analysis

Model	Logistic Regression	SVM	XGB Boost	LSTM
Accuracy(CV)	0.89	0.88	0.90	0.96

Discussion

-It was expected that the Logistic regression algorithm and the SVM achieved very similar levels of accuracy.
-Moreover, based on similar projects, I did expect the LSTM to be the best performing model which turned out to be the case
- There was also a challenge in encoding the features in dataset given that they needed to be encoded in a very particular way otherwise they couldn't be fed into the learning algorithms.
-Vectorizing the data and implementing the LSTM required huge computational memory.

Future

- Try to make use of different word vectorizers such as word2vec or TFIDF for the purpose of model vectorization analysis.
- Try to make use of other neural network models such as a CNN to see if I could get better accuracy.

References

[1] "How to Use Word Embedding Layers for Deep Learning with Keras," Machine Learning Mastery, 30-May-2018. [Online]. Available: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/. [Accessed: 15-Dec-2018].
[2]"Linguistic Regularities in Continuous Space Word Representations" Omas Mikolov, Wen-tau Yih, Geoffrey Zweig. Microsoft Research. [Online]. Available: https://www.aclweb.org/anthology/N13- 1090. [Accessed: 15-Dec-2018]