Toxic Comment Classification challenge on Kaggle.

René Ángeles
Udacity Capstone - Machine Learning Engineering

November 22, 2019

1 Definition

1.1 Overview

In this project we work on the Kaggle toxic comment classification challenge, aiming to improve online conversation. In a nutshell, the challenge consist in build an ML model to classify Wikipedia, talk edits page, comments as being toxic, severe toxic, obscene, treat or identity hate.

1.2 Problem Statement

Our aim is to explore the performance of three well models for the analysis of this data set: CNN, GRU and LSTM on top of the GLOVE pretrained word embedding. We not only aimed for a model that performs well to train and deploy this model on Sagemaker.

1.3 Metrics, goals and reasults

To gauge our work we take as benchmark for comparison is the most voted kernel of the competition: a Naive Bayes - Support Vector Machine model, it reaches 0.9772 auc roc, while the current leader team reaches 0.98856. We manage to top such benchmark, but the main results of the work consist in exploring the performance three popular model for NLP: CNN, LSTM and GRU. When combined with a given preprocessing, and using the Glove pretrained word embedding our model tops our benchmark model under 10 mins, on a laptop.

In addition, we deploy the CNN model to an interactive website via SageMaker and on the road we create some utilities to be able to use Torchtext, our preprocessing tool, to train on Sagemaker. Our simple tools allow people to use the befits of tochtext for prepossessing and the befits of SageMaker for training on GPU. After presenting our analysis we discuss performance and fure directions.

2 Analysis

The project overview,

3 Problem statement

4 Datasets and inputs

Jigsaw provides a training dataset of 160K comments, with binary labels corresponding to the following classes: Toxic, Severe toxic, Obscene, Threat, Insult or identity hate. Additionally, a test dataset with over 153K unlabelled test comments is provided to make a submission to the competition. Both label and unlabelled distribution were taken Wikipedia's talk page edits. Additionally, we shall use a pretrained word embedding that is included via Torchtext and we will further trained this, and other model layers, as required.

5 Solution statement

There is several competitive models developed in this competition. Rather than focusing on creating a model to reach the accuracy of top competitors, which may involve ensemble techniques as well as vast computational resources, we shall focus on the task of creating developing models that can be train, deployed and updated on SageMaker, emulating a production setting. To this end, I propose to use three well tested deep learning strategies on natural Language processing:

- Long short-term memory
- Gated recurrent units
- Convolutional neural networks

Of course, on the road we shall tackle text preprocessing and a choice of pretrained word embedding. We will use Pytorch as model build library as it is easy to train locally and it facilitates deployment on SageMaker.

6 Benchmark model

The metric of this multi-label Kaggle classification challenge uses the average of the Area Under the Receiver Operating Characteristic Curve (ROC AUC) to classify solutions. Currently, the team leading this competition scored team 0.98856%.

As explained above, we aim to emulate a production environment not only for this particular problem but applied to other similar text classification settings. Hence it

is reasonable to demand our model reach the modest 97% ROC AUC, but creating code that is easily trained, deployed and updated via Sagemaker. Furthermore, our models should be versatile enough that further improvements are easily achivable and generalisable to other classification settings, e g. modifying the network architecture and hyper-parameters must be easy.

7 A set of evaluation metrics

Our performance evaluation metric will be the class averaged, ROC AUC. Roughtly speaking, it measures how a model's probabilities separate the positive classes from the negative classes and, crucially, is a metric that asses a model regardless of the threshold chosen to map probabilities into categories. In addition, our aim is that the model is compact enough to perform on laptop cpu in around one hour (without hyperparameter tuning). Our non-quantitative metric is to create a metric that can be deployed with SageMaker, which can be easily improvable and updated by further work.

8 An outline of the project design

We shall use the contents of the Udacity nanodegree contents [**?**], the course on Sequence Models by Andrew NG [**?**], and the Ben Trevett tutorial series on sentiment analysis [**?**] to follow the following program:

- Data Exploration (Pandas)
- Text preprocessing (Spacy [**?**])
- Choosing an appropriate word-embedding (Torchtext)
- Try different well established models such as LSTM, GRU, Convolutional Networks (Pytorch)
- Deployment via Sagemaker.