Ryan Bui

4/28/2023

Data Mining Report

This is a kaggle challenge created in 2019 that is a response to a previous kaggle challenge which also set out to detect toxicity. However, they noticed that many of the models from that challenge incorrectly learned to associate the names of frequently attacked identities as toxic. So, my goal is to create a model that accurately detects toxicity while also minimizing this unintended bias.

In this Kaggle challenge we are given a large dataset of 2 million public comments from various news websites, and we are tasked to develop an NLP model to accurately predict how toxic these comments are. The comments have already been labeled on how toxic they are, so this will be a supervised task. This data set was collected by a plugin called "Civil Comments" which was a way for people to peer review comments posted on independent news websites. When they shut down in 2017 they released their archive of comments to the public. The comments in the data set have already been labeled. According to the Kaggle challenge page it was labeled by 10 people who each labeled the comments either: Not Toxic, Hard to Say, Toxic, or Very Toxic

For the Data Analysis and Preprocessing I made some simple histograms to have a look at the distribution of the target variables. And then I made it so that if the target variable was greater than 0.5 then it was 1 and 0 if otherwise. After doing this transformation it was clear that the data was very unbalanced with only 144,000 toxic examples and over 1 million non-toxic

examples. In order to combat this I decided to use downsampling in order to make the data set more balanced. It did however bring down our dataset to about only 248,000 training examples. I could have up-sampled the data but this would end up increasing our training examples to over 3 million which would have a drastic effect on training times so I decided to downsample.
We then moved on to cleaning the text up in order to make it easier for the model to read. Then I tokenized the text using each model's respective tokenizer and created the attention masks.

The first model I used was BERT for text classification from the hugging face transformer module. I used pytorch to finetune this model and it was becoming very clear that training this specific model would take a very long time as each epoch was taking 4 hours to get through. So I decided that I wanted to use TPU's to increase training speeds. However pytorch does not natively support TPU's so I had to switch my model's to use Tensorflow. In my repository I have posted both the pytorch version and the Tensorflow of the original BERT model. However while training with tensorflow versions of both GPT2 and BERT I realized that the model was over-fitting a lot. So I tried adding regularization, increasing and decreasing the number of epochs. But nothing seemed to work, so for the graphs and classification report I used the pytorch version of BERT. I would definitely like to try and resolve this issue in the future. With the pytorch version of BERT I was able to achieve a precision of 0.92 and an F1 score of 0.89.

The last model I used was Naive Bayes' Algorithm. Naive Bayes is a probabilistic classification algorithm. Based on strong independent assumptions, it uses Bayes' Theorem as a basis for the model. There is only a small impact on performance even though the assumptions may be wrong. This is why the algorithm is called naive. This model was able to get an F1 score of 0.84. Which is very good performance when considering how easy this model is to run. It requires very few lines of code and it runs in basically no time.