Project 3 - Useful YouTube Comments

Clay Negen

4/9/2020

Intro

In the world of computer science, YouTube can be one of the most useful sources of information. In addition to the video, the comment section can be a source of information as well. They could tell you whether the video is not credible, they could suggest other ways of doing things, or they could even provide more insights into smaller details of the video. But most comments aren't a useful source of information, they're usually just responses like: "nice video" or "this was crap". My project is a JavaScript application that can import comments from a YouTube video about software development and classifies them as "useful" or "not useful" using machine learning.

In short, it is a text classification program that uses artificial neural networks to perform machine learning on a dataset specific to youtube videos. The library I used is called Brain.js, it provides a javascript wrapper for GPU accelerated Artificial Neural Networks. It actually uses python dependencies, but with my lack of programming skills in python, the wrapper was perfect for me.

Application

I built a react application that can import data from the youtube data api, transform it into usable JSON objects, and write them to a database for use. I utilized FireBase (Google's free noSQL database) in order to store, visuallize, and update my data live. My initial plan was to manually label a training dataset by finding legitimate, useful comments on youtube and create a database. This was not practical because the comments vary so much on how they are worded, so retrieving comments became too time consuming. Next I tried importing comments from a video, assigning them all as "useful" then going through the dataset and manually assigning the ones I found to be "not useful". After using this data to train my ANN, I still was not getting the results I wanted and realized I needed a much larger data set

Data

For my training dataset, I imported 100 random comments from 10 different youtube videos that taught you how to write an Iphone Application. If they had likes or replies I marked them as "useful" and if they didnt I marked them as not useful. For my testing data I used all of the comments off of a different Iphone App tutorial video, and ran them against my trained network. The biggest challenge of this project has been the data. I highly underestimated how much data was needed to train an artificial neural network, and my idea of manually assigning values was not possible.

Testing

I built the main portion of my project in JavaScript, using the library Brain.JS. I used Node.JS for my javascript compilation and python 2.7 for the artificial neural network. I imported my data to local files, mapped my training data to inputs and outputs a neural network can understand (it uses json objects or key value pairs for this). I then instantiated a new recurrent neural network:

```
const net = new brain.recurrent.LSTM();
```

Brain.js recurrent.LSTM means its using forward and back propagation, as well as Long Short Term Memory for string parsing. You can specify how many hidden layers are in the network and how many nodes per layer, but by default it comes with 1 hidden layer equal to the number of inputs. I messed around with the amount of hidden layers, but found the default to work the best.

I then trained my 'net' with the mapped data for X iterations (or epochs) and logged my training error every iteration. This is where my small dataset really came to bite me in the butt. After about only 20 iterations of training, the model started to overfit to the training data and would produce odd results. So I continually produced new datasets and attempted to train my model for different periods of time. The final result as shown in the video was 728 pieces of classified data ran for 20 epochs on our network. While the results were not exactly what I was looking for, I think it achieved something close and this was a really good start. I think in order to really get this project to succeed, I would need a much larger, and cleaner data set.

What Other work has been done?

Text classification, and sentiment analysis have already been performed on a dataset of youtube comments, an example would be classifying spam comments.
However, I have found no work done on classifying the usefulness of comments.

Why?

I did this project because most of my learning these days is on youtube, and I often look to the comment section for help. However some of the most useful comments slip through the cracks of youtube's algorithms and get lost in the non useful comments. So I did this project to surface those comments and extract useful information from the data that is youtube comments.