Alex Ginglen

CS410

October 18, 2020

Technology Review – NaturalNode and Brain.js for Natural Language Classification

As the world of technology grows, so does the amount of natural language data and the need to classify that data. Classifying natural language data is relatively simple for a human being, but a human cannot read and analyze millions of documents of text per day. Now more than ever, computers are needed to classify natural language. Two modern tools that can be used to classify natural language are NaturalNode and Brain.js. However, information technology professionals need to choose the best tool for their uses. For this technology review, I tried to determine whether NaturalNode or Brain.js was better at natural language classification. I created a test to determine how accurate and efficient the various algorithms of NaturalNode and Brain.js are at determining topics of a simple natural language dataset.

NaturalNode is a Node.js package for text parsing, analysis, and classification. For the purpose of this review, I only focused only on the classification features. To classify text using NaturalNode, I only needed to pick a classifier algorithm, add training data, and provide data to be classified. NaturalNode offers two different classifier algorithms, a Bayesian classifier and a logistic regression classifier. Once I had my training data, I was able to execute the function classifier.train() to generate the trained model in json that can be reused for natural language classification. Saving and loading the json used for classification can save a lot of processing time, especially if the training dataset is quite large. NaturalNode can also be run on any

platform that Node.js can be run such as on websites, in web services, and in serverless architectures like AWS Lambda.

Brain.js is a GPU accelerated neural network package written in javascript. Brain.js is an incredibly powerful tool because it abstracts users from the complexity of neural networks and allows them to write powerful forecasts and predictions using only a few lines of code. To use Brain.js, I instantiated a neural network with brain.NeuralNetwork(). Then I provided a set of training data, which was a set of inputs and outputs. Once the data was trained, I simply had to give the neural network data to classify. Similar to NaturalNode, the training data could be saved as json for later reuse. Additionally, Brain.js provided multiple options for tweaking the neural network such as the number of iterations through the training data, the acceptable error percentage from training data, and the learning rate. For the sake of this review, I used the default settings. Brain.js also offered additional features for training multiple neural networks in parallel.

I created a simple test to determine the accuracy and efficiency of NaturalNode and Brain.js for classifying natural language. To do this, I ran exactly the same test multiple times on each tool using different algorithms. I measured the time to train the data and make a classification. Overall, I had three instantiations of the same test: one using Bayesian classification with NaturalNode, one using logistical regression with NaturalNode, and one using the default neural network with brain.js. After several executions, all three algorithms were able to successfully pass the classification tests. The Bayesian classifier was the fastest with an average time of 9ms. The logistical regression classifier was the second fastest with an average

time of 54ms. And the neural network was the least fast with an average execution time of 112ms.

In conclusion, I was able to determine that the Bayesian classifier with NaturalNode was the fastest way to classify topics of a small natural language dataset. Although my test was simple and naïve, I was able to show which approach was most likely the best option for a use case similar to my test data. In the future, I would be interested to test each tool and algorithm with a larger and more complicated dataset. I would also like to understand the neural network algorithm better so that I could develop a test to optimize the neural network settings. Text classification is becoming a common task all over the information technology industry. I hope to take this knowledge and use it to make a difference in my career.