Galen Byrd

11/12/18

Midterm Writeup

My Jarvis is reporting 96% accuracy on the testing data I have given it, using linearSVC. This accuracy is defined as the total number of correctly classified phrases out of the total number of phrases given. I ended up trying k nearest neighbors, decision tree, random forest, multinomialNB, and linearSVC and went forward with linearSVC as it was returning the best results. As expected, the statistical analysis of the training data shows that the classifier is very well trained, other than using a decision tree.

For my statistical analysis, I decided to use scikit learn's metrics module to report precision, recall, f1-score, support, and the confusion matrix. Support is simply the number of observations of a particular class. Precision is the number of true positives (phrases the classifier correctly classified) divided by the true positives plus false positives (phrases the classifier incorrectly classified); which translates to; out of all the observations of one class given by our classifier how many were correct? Recall is the true positives divided by the true positives plus the false negatives (phrases were actually the correct class, but our classifier grouped the phrases incorrectly), which gives us; out of all the phrases actually given in a class, how many were predicted correctly? The f1-score is a combination of these statistics, 2 x (precision x recall)/(precision + recall), which gives us a better understanding of the classifier than either precision or recall alone. I also reported the confusion matrix, which indicates to us which actions the classifier is having trouble separating. The rows represent the actual class of an observation and the columns represent the classifier's predicted class. This means that along the diagonal will be represented how many observations the classifier classified correctly.

Given this understanding of the reported statistics, we can see that the classifier has a tough time classifying phrases relating to TIME as this can be a very obscure concept and may not be able to be understood as completely by a computer just yet. It would typically get confused between TIME, WEATHER and PIZZA because those three actions all relate to time and/or temperature somehow. When using words like “today” or “currently” or “hot” or “cold” the classifier gets confused as those words can be signals for different classes. Using linearSVC, we end up getting .91 as the F1-score for PIZZA and .93 for WEATHER, which is significantly higher than using other classifiers.

One way to improve jarvis would be to give it more specific classifiers than the general actions it is currently classifying. If we were able to look for more specific language, that would make it easier to differentiate between classes. Obviously, we could also add more training data. Theoretically, there does not exist an upper bound for how much data is needed to make the classifier better, but with only 61 training observations and 31 tests, there is not a huge corpus for the classifier to train from. Another potential way to improve jarvis would be to tune the parameters to values that would help the classifier be better, although I was not able to find these values when I tried.

I reported statistics on both the testing and training data to check if the classifiers were too biased or had too much variance. In order to save the testing data with the correct action, I decided to structure my testing mode as only able to run one action per testing time. This is so that I can write to the testing data table with the correct action as opposed to the predicted action. I could have done it where jarvis will take in the correct action for each phrase in testing time but that seemed less clean code-wise.

This project was very overwhelming to start, but once I understood the starter code and felt confident I was able to complete the actual code of jarvis in just a few hours. The

conversation logic for jarvis I implemented is pretty basic, as it will only respond to the three phrases it must, but it could be expanded to respond to many more phrases pretty easily. We could also get Jarvis to respond to the phrase with another phrase as opposed to a prediction, but I am not sure how I could make it respond different phrases for one particular class. For example, I could get it to respond “I am ordering a pizza” every time it classified a phrase as pizza.

One thing that I need to get better at is looking at other peoples code/starter code and figuring out the logic the original coder was trying to follow. When I see code I am unfamiliar with I let it overwhelm me even though I know I can understand most of it. I am also not well versed in object-oriented coding in python so that was another small hurdle to learn. The biggest issues I ran into were dealing with the classifier and the pipeline. I had started without a pipeline, simply trying to input a list of strings into the classifier and it would not run. I shortly figured out I needed the pipeline to process the input text. With the pipeline in place, everything began to run smoothly. I then proceeded to test the different classifiers. I tried to tune some of the parameters of the classifiers but I found that the default parameters were giving better results than when I was adding parameters myself.