Jeffrey Chiavetta

Project Report: Autism Classification

This work originally served as a final project for a course on statistical Natural Language Processing. The course was meant as an intensive overview of machine learning methods through the lens of text-based language data. The project was not designed for the class itself, but rather represented ongoing research from a professor at the University of Arizona campus. After this project, I continued to work on it with that professor and a small group. Please note that I am not allowed to provide the actual dataset due to the terms of a NDA.

The data consists of 4,480 records of 8-year old children, each one accompanied by a gold label, 1 if the child was diagnosed with autism and 0 if they were not. This makes it a perfect candidate for supervised learning. The cases are split very evenly between positive and negative diagnoses. The team has stated that with "simple" methods (FF/BP, C5.0) with bag of words, they found a baseline of 86% accuracy on the task. The goal here was to establish a baseline of my own, and then try different methods to see how these affect the accuracy and f1 scores.

This is an important issue for its humanitarian impact. Having accurate machine translation to streamline the process of autism diagnosis goes a long way to help physicians work with this condition, especially because it is so common. Due to its prevalence, such a task benefits from machine learning methods. Rather than having the final word on diagnosis, this software could classify every case document and recommend the subset with a given confidence interval to the physician for final review.

The data is formatted in a spreadsheet, where each of the rows are an individual case, and each of the 450~ columns include relevant information. Some of these were numbers, incoding the diagnosis, test dates, etc. while others had simple strings for names of physicians, schools, etc. Most importantly, a large number of columns had detailed prose reports by the physicians, detailing aspects of the patients' behavior as well as test scores, personal opinions, and recommendations. The contents of these columns became the vocabulary used to construct feature vectors, one for each word. This resulted in vectors of size 40,000~.

My approach was to establish a baseline and use the sci-kit learn package to experiment with different classifier models to compare their performance. In addition, I wanted to try putting in custom features on top of the bag of words to see if accuracy could be increased by taking advantage of unique properties in the data.

To import the data I used the csv and xlrd modules. About 300 of the cases were unable to be used as we didn't have gold labels, so I filtered those out, leaving 4480 cases. Of these, 500 were set aside as a test partition, with the remaining 3980 as training. My goal was to test different classifiers rather than explore hyperparameters for one in particular, so I opted not to use a development set (especially because the Naive Bayes and Logistic Regression models trained in only a few seconds).

Next, I set up lookup tables and lists for the gold labels, for both the training and dev sets. In the same pass over the data, I accumulated the data from the cells of each row in order to later create a vocabulary.

My primary contribution aside from testing sci-kit model performance was to see whether custom features are of any use in this classification task. To test this, I looked through the data to see if any columns in particular could be good indicators of autism. I found a series of columns that contained information on previous diagnoses for the children. I noticed 5 in particular that showed up in many of the cells: Down's Syndrome, ADD, Microcephaly, OCD, and a history of seizures. I decided it was a reasonable hypothesis that prior diagnoses like these could be an indicator of later autism diagnosis.

The purpose of the add_features function is, then, to encode the information from these prior diagnoses into the model. I set up a feature for each of the 5 diagnoses and ran regex searches on the most common orthographical forms. Given a match, the feature was set as 1. At the end of the iteration,

these 5 features were added onto the end of the vectors. In other words, the output for the function is a matrix of vectors, where each vector contains 5 more dimensions than it did before. Testing this function on a case file that included two of the prior diagnoses yielded [0, 1, 0, 0, 1] for the last five indices, which matches up correctly to the order in which those features were appended onto the list.

The next two blocks iterated over the case documents, joining every cell together into one large string, so that sci-kit could work with it.

Next, a count vectorizer was set up that build the vocabulary and transformed the case strings into unigram vectors, train_X and test_X. Then a tf-idf transformer took those inputs and created train_X_tfidf and test_X_tfidf. Then, my add_features function was applied, creating train_X_tfidf_custom and test_X_tfidf_custom.

The rest of the code contains the instantiations, training, and testing for the three types of classifier, and the print statements that result in the tables below. These models test using unigrams by default, but can be used with bigrams if the vectors are built beforehand using a count_vectorizer object with bigrams set as a parameter.

The following is data gathered from my testing:

Multinomial Naive Bayes

	Unigrams	Bigrams	tf-idf	tf-idf + features
Accuracy (ROC)	80.29%	81.52%	89.15%	89.13%
Precision	76.06%	60.26%	44.78%	44.78%
Recall	69.41%	89.81%	100%(?)	100%(?)
f1	72.59%	72.12%	61.86%	61.86%

Logistic Regression

	Unigrams	Bigrams	tf-idf	tf-idf + features
Accuracy (ROC)	89.12%	84.63%	90.50%	90.57%
Precision	82.20%	72.20%	92.59%	90.58%
Recall	76.21%	71.84%	60.68%	60.68%
f1	79.09%	72.02%	73.31%	72.67%

SVC

	Unigrams	Bigrams	tf-idf	tf-idf + features
Accuracy (ROC)	88.37%	87.75%	92.72%	92.74%
Precision	86.05%	82.32%	87.5%	86.36%
Recall	71.84%	72.33%	74.75%	73.79%
f1	78.31%	77.00%	80.63%	79.58%

For evaluation measures, I decided to use both an overall accuracy calculation and f1 scores, to better represent the results. Overall the accuracy was good, often surpassing the baseline 86% that the researchers had previously gotten on the project. The working hypothesis was that each column would

be more accurate than the previous; in other words, I expected tf-idf + custom features > tf-idf > bigrams > unigrams. This was borne out in some ways but failed in others. Note first the alternation between unigrams and bigrams. Surprisingly, bigrams ended up performing worse than unigrams in most cases. Some of these were dramatic, including a full 10% drop in precision for the LR classifier. Clearly unigrams were more effective here; one possible explanation is that two-word phrases were not very helpful because these were more likely to be observational or opinionated, whereas the important data were single words, like “disorder”, “autism”, or even digit test scores.

The next important point is between count unigrams and tf-idf unigrams. The tf-idf scores were much better, and this was definitely the most important conclusion to be drawn from the data. The increase in accuracy ranged from 1% for LR to 4% for SVC and all the way up to an 8% increase for NB. Using tf-idf scores for the features turns out to be very important for this task, likely due to important words that appear in only a small subset of the documents, thereby giving them higher weights.

Last is a look at the regular tf-idf unigrams and those augmented by the custom features based on previous diagnoses. The results here were more of a mixed bag, with no real jump in either direction. This is disappointing given the hypothesis that autism may occur more frequently when coupled with certain other conditions. The take-away of the results is that the best results are gained by using simple unigrams transformed into tf-idf counts, on an SVC classifier. It is also worth noting, however, that due to the SVC’s quadratic run-time, it takes much longer to train (5 mins, 45 seconds on 4GB memory) than the other two, which both take only a few seconds. It may in some cases be more efficient to use Logistic Regression, which offers the second-best results and also avoids some strange results found in the ‘tf-idf’ and ‘tf-idf + features’ columns of the Naive Bayes.

There is still much that could be done on this project in the future. I would like expand on the custom features idea, as 5 is a small sample size. Encoding the information in specific columns as features still seems promising, as each column corresponds to a specific finding in the physician’s report. These can be easily compared between case documents, and are much less vague than just picking unigrams out of the vocabulary.

To give an example, encoding “OCD” in a feature based on the column specifically meant for previous diagnoses means that string is definitely relevant to the case, as opposed to matching it in any random cell, which often contain unhelpful strings of text like “The child was once tested for OCD” or “OCD was not considered”. Therefore it is my opinion that using a greater sample size of custom features may still be a valid way to improve performance, despite lackluster results here.

Another domain for future research would be to clean up the data, which has errors. Specifically, there are an enormous number of typos and ungrammaticality, as in “comunicaiton skills are limited both both receptiv ane expressive language but is echoing words”, which is very flawed. I would like to more fully understand sci-kit learn’s count vectorizer to see how it formats the data, as I suspect that improvements could be made. I assume there is a filter to include only words that appear a given number of times, which would eliminate typos. Indeed, the vocabulary created by the count vectorizer does not include these typos.

However, because of the sheer number of misspellings, it seems like it might be worth trying to include this data by matching correct spellings to the typos at the character level so that it can be included in the counts.

This concludes the analysis of the autism classification task. This task involved taking data involving autism diagnoses and attempting to train a model to classify a test set successfully. The results were mixed but overall positive. The differences in some regards were inconclusive, but there were two important points. First was the establishment of a baseline classification accuracy that resembled that given by the research group. The second point was establishing that using tf-idf counts shows a marked increase in accuracy compared to basic word counts.