COMP30018 Project 2 Report

1. Introduction

Throughout the second half of the semester, we have learned various algorithms that can help us detect patterns within datasets. In this report, I will examine the usefulness, of a decision tree for detecting patterns between words in a tweet and where that tweet came from.

2. Hypothesis

With the problem at hand, I hypothesized that a decision tree may have decent results because it might be able to create a path between correlated attributes, such as 'Obama' and 'Washington', to get us to the correct prediction. Decision trees have also proven to withstand noise and useless attributes pretty well. But given that I'm using the best 35 attributes (as opposed to using 100 or 400+ attributes), useless attributes may be quite hindering. Precisely why I assume it will be a decent model, but not the optimal model.

2.1 J48 Tree vs REP Tree

WEKA provides us with multiple trees and I decided to compare the performance between J48 and REPTree because of their differing branch splitting criteria. J48 uses Gain Ratio as the splitting criteria while REP uses Information Gain. With this slight difference, I imagine that J48 will provide better results because the gain ratio gives more balanced results when trying to find the attribute with the least amount of entropy.

2.2 Performance Measures

I will measure the performance of my models with accuracy and then look at the precision and recall for individual classes to see if it was particularly good at classifying a specific class.

3. Setting the Baseline

It is important to choose a baseline that is both informative of the problem at hand and moderately easy to beat. I was between 0-R and 1-R. 0-R decided to classify every instance as San Diego because it was the most prevalent class in the training set. 0-R managed to obtain an accuracy of 26.16% when tested on the development set. 1-R, on the other hand, became a random classifier because it preferred the TweetID attribute. It's rather low score of 23.01% accuracy tempted me to choose 0-R as the more informative and accurate baseline. It wasn't until I realized that I was overfitting this simple model to the nature of the training set and development set that I opted out for 1-R. Though 0-R may have had better results, our dataset has a pretty balanced distribution of classes which makes 1-R more illustrative of the data.

4. Training the Decision Trees

I trained my models using the best35 arrf file because I believed it would allow for the optimal generalization.

5. Validation Step for J48

Given that we are using the holdout method, I loaded dev-best35 into the classify tab and started the classification process. To my disappointment, J48 received an accuracy score of 28.96%. It was definitely better than 1-R but whether a 5% increase in accuracy mattered was arguable. From there, I visualized the tree in WEKA and found out that the tree provided nodes for an unnecessarily large amount of TweetID's. The URL attributes were also deemed as very important, but from a human perspective, it's not possible to tell if they really are important.

6. REP Tree

The REP Tree had a similar story. It obtained a lower accuracy score of 27.40%. But, interestingly, it deemed the URL's as rather unimportant. Unfortunately, it set TweetID's as nodes pretty early in the tree which probably contributed to its lower score.

7. Feature Selection & Re-Validation

The fact that the REP Tree decided that the URL's were largely unimportant encouraged me to remove them from the set of attributes and retrain both the trees. I didn't remove words such as 'health' and 'lol' because both J48 and REP placed them towards the bottom of the tree, which hints that not much would change if I removed them.

7.1. Re-Validating J48

From hindsight, removing the ID's and URL's proved to be effective as it greatly reduced the depth and complexity of the tree. This, in turn, allowed for better generalization and faster decision making, which gave us an accuracy of 30.67%.
However, these were deceiving results. With J48, San Diego had the lowest precision (.276) and a really high Recall (.972). I found that the tree classified a large amount of tweets as San Diego, which was rather odd because J48 placed the attributes 'seattle' and 'houston' above 'diego'. So I would have expected to see a higher Recall for the classes Se and H.

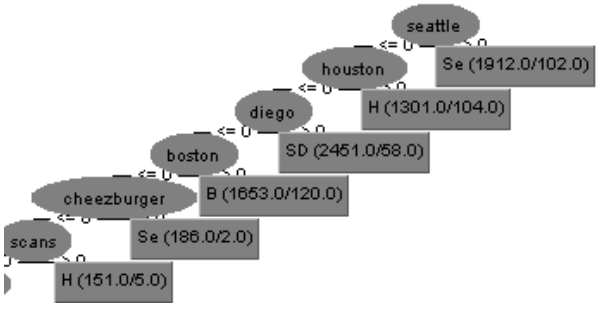

Figure 1Visualization for J48 Tree after feature selection.

The best explanation for this behaviour is that I probably overfit the model to the training and development set when going through the feature selection process. I caused the tree to choose the most prevalent class in the data set. In fact, the precision and recall for SD in 0-R (Precision: .262, Recall: 1.000) was not that far from the results J48 produced, which further confirms the idea that I overfit the model.

7.2. Re-validating REP Tree

Unfortunately, the REP Tree has the exact same story. San Diego, once again, stands out among the classes with a recall of .972 and a precision of .276. Though the nodes are placed in a different order in this tree, it produces nearly the exact same results as the J48 tree in the confusion matrix which is a consequence of overfitting.

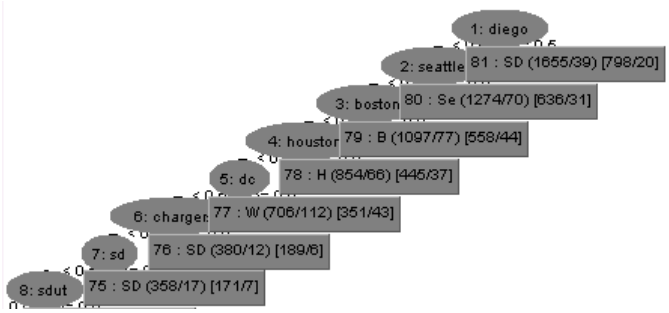

Figure 2: Visualization for REP Tree after Feature Selection. Notice that this tree also isn't very broad and instead deepens quickly in one direction. This is a very hindering quality for its decision making.

8. Conclusion

Given the set of attributes, the decision tree was largely incapable of predicting where a specific tweet was coming from. Initially running J48, I found that it had a very disappointing accuracy that only slightly beat a random classifier. I attempted to increase its accuracy by getting rid of attributes that, to the human eye, looked irrelevant, but instead I caused an increase of bias towards SD in both the trees.
I was correct in assuming that J48 would provide better results than REP because it used Gain Ratio to split the branches. And it did cause a different ordering in the nodes, but the trees still performed at similar levels.
I believe that the largest problem came from the fact that decision trees rely on closely related attributes that help narrow down a decision. Our attributes, though very indicative of a certain location, were rather unrelated,

which caused both trees to make senseless decision paths such as the one shown in Figure 2. And though I mentioned that a tree with more attributes may behave differently, I believe the fact that it has to start with a single node at the top is what will hold back the decision tree from performing well in this problem.

References

Cieslak, David A., and Nitesh V. Chawla. "Learning decision trees for unbalanced data." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2008.

Abernethy, Michael. "Classification and Clustering." *Data Mining with WEKA, Part 2*. IBM, n.d. Web. 7 Oct. 2016.