Makena Kong

Whitney Larsen

CSC 466 - Khosmood

Project 2 Report

Part 1

This portion of the project is a K-means clustering algorithm on the text fields of the digital democracy files with the purpose of discovering the main topics of each field and seeing if they corresponded to to committees.

Implementing the K-Means algorithm on text was confusing at first. We (as in Makena) didn't know what each "point" was supposed to be - were they supposed to be words or sentences or phrases (bigrams/trigrams). Then, we realized that each point is the set or dictionary of features. After this was cleared up, we had to figure out the best way to implement the algorithm.

We decided that the best data structure to hold our data is a Pandas Dataframe. Each row is a point, each column is a feature name and each cell is the value of the feature. Using dataframes made calculating the distances between rows (points) very easy since they allow for matrix / vector operations.

To get the features we tokenize the words and remove the stop words, then lemmatize them so that words with the same root would be counted as equal. Then we take check

if that lemma already exists in the dictionary. If it does not, we create a new entry in the dictionary with the lemma as the key and 1 as the value. If the lemma does exist in the dictionary, we add one to the value at that key. This way the word agriculture, for example, being said hundreds of times by the Agricultural Committee counts more than the word agriculture being said by some other, largely unrelated committee.

First, we read in the tsv file into a DataFrame where each row is an utterance and the text column contains the text field for each utterance. Then, we applied the getFeatures method to each record in the Data Frame, which was then converted into a dataframe where each key in the feature dictionary is a column and each row is an utterance. This was then split into the testing and training sets - where the training set was passed into our k-means algorithm implementation. Our k-means algorithm followed Algorithm 13.1 in the Zaki textbook. First, random k centroids were established and clustified - each point was assigned to its nearest centroid. Then, new centroids were found by taking the average record of each cluster. We repeated the previous step updated our clusters and centroids until our centroids no longer moved. Then, we returned our clusters and tried our model against our test data set.

After hours of fine-tuning our getFeatures method, we were able to get our program to generate clusters that went from 20% to 70% similar to SciKit Learn's K-means clusters. We passed in the same training set and used Euclidean distance for both, so this reassured us that our model was somewhat accurate.

After running our clustering algorithm multiple times, our highest accuracy was about 30% and our lowest was about 22%. We didn't do this many times because our algorithm took around 15 minutes to complete - which we are aware is over the limit. We didn't want to change our getFeatures method because it was producing reasonable clusters. Anyways, we believe that our program is so inefficient because of all the loops we had to use - which was terrible with the size of the committee utterances file.

Part 2

This portion of the project is a Decision Tree algorithm on the text fields of the Digital Democracy files with the purpose of discovering the committees or hearings depending on the input.

We are processing the text similar to how we did in part 1. The main difference is that all of the features should be binary - 1 and 0 representing True and False. Our decision tree is represented as a Python Binary Tree, where each Node contains and decision and each leaf contains an id.

To create our model, we use the training set to build the decision tree recursively. Then, we predict our results on a test set by traversing the tree with our test data to find which leaf it should end up as.

When writing this algorithm, our biggest issue was figuring out Entropy. To get the Information Gain when creating a new node, you need to subtract the entropy of the attribute with respect to the target from the entropy of the target. Getting the entropy of the target is easy, but figuring out how to get the entropy of an attribute was confusing. In fact, we are still confused on how to get this. The calculated split entropies often seemed to be larger than the target entropy - which didn't adhere to the information gain equation.

Overall, we did not fully complete this function. The decision tree seemed to only output one node - the root node. Therefore, there were only about 4 possible outcomes of our tree and there are certainly more committees and hearings we should be predicting. We will continue to work on this and turn it in tomorrow.

Part 3

We are still working on this portion of the project.

:)