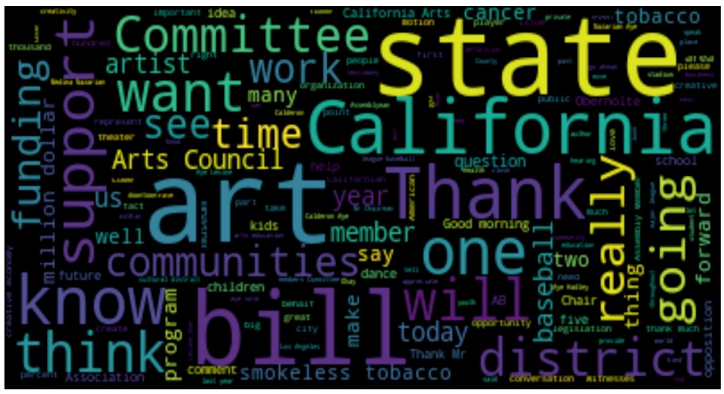
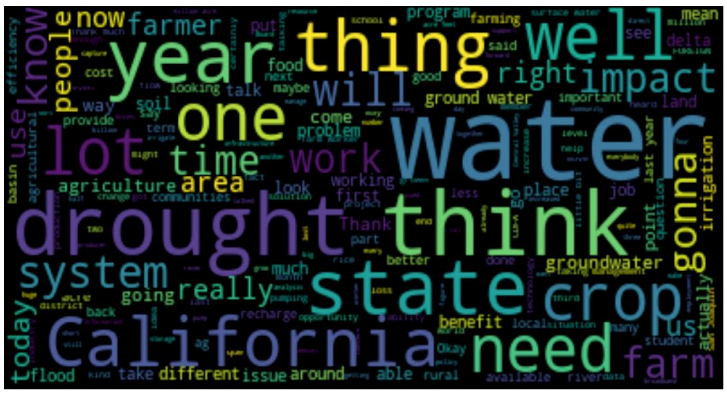
Markelle Kelly, Josh Abel, Daniel DeFoe
CSC 466
Professor Khosmood
5/12/19

Project 2 Report

Introduction: What had to be overcome

In general, speakers in the California Senate tend to repeat many of the same words throughout their hearings. Some of these appear more or less equally across committees, including words related to the Senate proceedings, such as “bill” or “state”, and stop words. However, since different committees focus on different topics of interest, there are also patterns of repeated words which are more committee-specific, such as agricultural speakers frequently mentioning “drought” or “water”. Although it is second nature for a human to recognize a difference in topics when listening to a discussion, it is not a simple task to transfer this common sense to a computer. Precise features must be picked by asking the proper questions that might help a human decipher who says what. The following word clouds will help explain the problem at hand.





Which committee should be predicted to have said the words from the image on the left and the right? The words in the left image belong to the arts, entertainment, sports, tourism, and internet media committee, while the words on the right belong to the agricultural committee. It is definitely easy to tell that the words from the right belong to the agricultural committee because the words “water”, “drought”, and “crop” are popping right out of the image. It also may be easy to tell that the words from the image on the left belong to the art committee because the word “art” is very prominent. However, there are also a lot of words that have nothing to do with art which stand out, such as “state” and “bill”, which may confuse a computer and make it think that the words for the left image belong to, for example, the judiciary committee. Therefore,

careful feature engineering must be performed to aid a computer in correctly deciding that the words from the left image belong to the art committee.

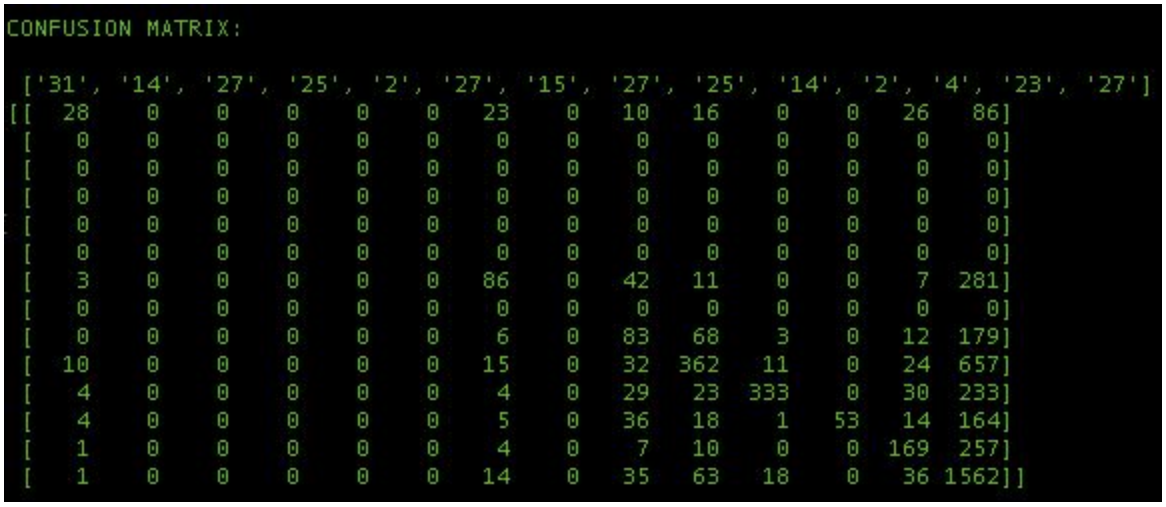
Part 1: K-Means

In the K-Means clustering algorithm, a getFeatures function takes a single text field and creates our features, which counts words corresponding to overarching topics of discussion. The topics of discussion, such as “Agriculture”, “Health”, and “Transportation”, did not directly correspond to the data’s “c_name” values, but represented all of the general topics. Due to the manually created myKM’s non-trivial computation time, limiting the number of features was of reasonable importance. To take the actual words being spoken into consideration, we created lists of words that were relevant to each overarching topic covered by the committees and kept counts corresponding to the total occurrences of any word in each list for each utterance. Therefore, each utterance was assigned a feature of word counts for these topics. Proportions of proper nouns, improper nouns, adjectives, and verbs also helped correctly categorize the committee ID. Several other features, including the number of sentences, the number of words, and the number of characters in each utterance were considered, but did very little in helping predict the correct committee ID. Using the actual proper nouns themselves as features was thought to have potential to help, but decreased the accuracy of the classifier, and naturally added a great number of features, slowing the process of the manually created myKM function. Finally, synsets for a subset of the important words in the utterance also added a great number of features, significantly slowing the algorithm, though they did have potential to help in correctly separating the topics of committee discussions. Because most of the features used were overarching themes of discussion, each of the K means at the end of the algorithm was formed by combined influences of multiple themes. We considered a total of 12 topics.

The manually created K-Means function myKM works with the same functionality as the scikit-learn K-Means algorithm. The scikit learn K-Means algorithm is certainly efficiency-optimized as it can run nearly instantaneously, while the myKM function requires non-trivial time to complete. The final results of our K-Means algorithm have some variance due to the randomness of the initial centroids. When we specified the exact initial centroids for both our algorithm and the scikit-learn algorithm, the resulting clusters were exactly the same. However, in a sample of 10 trials with randomly initialized centroids, we achieved an average Euclidean distance between our centroids and the scikit-learn centroids of 3.0882 arbitrary units. The myKM function returns three objects, the first being the list of final centroid means, the second being the actual

clusters of the input data, and the third being the counts of points classified in each clusters.

Because the centroids are initiated randomly, there is some volatility in resulting F measures, but because the program runs quickly, it is able to go through 7 iterations of myKM and get the set of clusters with the greatest accuracy. The number of clusters, which is expressed as k in the myKM function call, are determined by the number of unique classes of the target variable. When run on random subsets of 7,715 observations (about 25% of the total committee_utterances.tsv dataset), where there were 14 unique committee ID's, in five calls of the overall function the average best accuracy for each call the overall function was 0.2452.



The output confusion matrix when run on the pre-mentioned subset of 7,715 observations containing 14 unique committee IDs shows that many predicted classifications dominated all of the points. Most of the points in an iteration run producing the figure above were predicted to be in committee 27, which is the transportation committee, but only 1562 were actually correctly placed there. This could be because the list of words in the transportation feature was slightly larger than some others, and perhaps some of the words were slightly more general than other categories. For the committee IDs that had no points classified to them, this is because the clusters which may have contained most of those points contained more misclassified points of some other committee, meaning that no points would ever be classified as as this less populous category.

Part 2: Decision Trees

Our decision trees function, DT, takes a similar input of features to the myKM function. Again, to save time, we limited features by focusing on lists of primary topic

words rather than checking for presence of individual words. Due to the nature of the decision trees algorithm, it was not practical to use features such as the exact number of words in an utterance (as this resulted in a very large number of midpoints to check each time), and an investigation of binning these values indicated that they were still not beneficial to the accuracy of the algorithm. It is reasonable to infer by this that particular committees do not have dramatically different lengths of utterances, possibly due to the nature of longer speeches being split up into several utterances. In comparison to K-Means, the speed of our decision trees algorithm was affected much more dramatically by the addition of new features, so we had to prioritize the most important features carefully. Many interesting features we considered, especially those related to synonyms and parts of speech, that improved accuracy but slowed the algorithm significantly. By grouping words by topic rather than counting individually, we were able to dramatically decrease the number of features to allow the program to run in a reasonable amount of time.

To store our results, our decision trees algorithm employs a Leaf class, saving the rule, subset of the data, and predicted class (where applicable) in each node. This allows us to run our testing data down the tree to compute metrics assessing classification. We split the dataset into subsets of 70% for training and 30% for testing, using scikit-learn's train_test_split function.

Using the supervised decision trees algorithm to classify the data was a definitive improvement over unsupervised K-Means. Running on random samples of 7,715 observations, the mean overall accuracy of 5 runs was 0.161 and mean overall F score was 0.107. Running several random samples also revealed that some committees are easier or harder to predict than others: committees 4, 45, 76, and 145, corresponding to "arts, entertainment, sports, and tourism", "agriculture", "California's clean energy economy", and "public health and developmental services" had consistently high F1 scores. These correspond to relatively distinct categories, compared to committees 52, 64, and 131, which had consistently low F1 scores, and represented "civic engagement", "rules", and "energy, utilities, and communications", committees that might not use as unique of words.

There also were some commonalities in the types of rules that tended to show up early in the tree. Government, a list of words such as "government", "voter", "election", and "senate", consistently showed up on top, splitting by averages of 1-1.5 government words per utterance, and family, which included words like "youth", "parent", and "foster" was also generally at the top of the tree, splitting at 0.5 - that is, whether the utterance contained any family-related words or not.

The program also looks for the '-h' flag as the potential second command line argument. If detected, the program then drops all of the committee IDs which only correspond to one hearing ID. For the remaining committee IDs, which are represented in multiple hearings, at least one full hearing was put into our training set and one full hearing put into the testing set, ensuring that each committee could be predicted, while keeping the split as close to 3:1 as possible given this constraint. When using these long strings representing entire hearings instead of shorter strings corresponding to single utterances, we found that the small sample size (often as small as one observation for a committee in the training set) did not allow the tree to fully develop. Overall accuracy when run on the entire dataset was only 0.2, overall F score was 0.15, and many terminal nodes of the decision tree were never reached by the test data.

Part 3: Neural Network

In predicting the name of the speaker, the model chosen was a neural network using the scikit-learn package MLPClassifier. Many combinations of features were run on the neural network algorithm, and the combination with the best accuracy scores was chosen as the set of features. The chosen features are the following: Term-Frequency-Inverse-Document-Frequency Vectorizer (TF-IDF vectorizer), average word length per utterance, adjective counts per utterance, adverb counts per utterance, and "uhhhh" interjection counts per utterance. A scikit-learn TFIDF vectorizer was used to create feature vectors from each utterance in the training data, looking for the important words used most commonly by each speaker and then weighing the words most identifiable with the speaker accordingly. Then the nltk package was used to gather all the words and categorize them by their parts of speech (i.e. adjective, adverb, and "uhhhh" interjections). Following this, a parameter grid was used to pick the best hyperparameters to use as the building blocks of the neural network. The grid-search used 3-fold cross-validation to pick the best solver for weight optimization and the number of neurons in each hidden layer of the network. The best combination of parameters that the grid search found used the 'lbfgs' solver, which is an optimizer in the family of quasi-Newton methods. Second, the grid search chose two hidden layers, the first hidden layer consisting of 7 neurons and the second hidden layer consisting of 3 neurons. The number of possible classifications or number of individual speakers was determined by a parameter N that subsetted the data, where N determined the number of speakers who spoke the most words in the dataset.

Given the input features, the accuracy of the neural network classifier was 0.26345 when the N was 50 speakers, 0.23882 when N was 75 speakers, and 0.2064 when N was 100 speakers. This decrease in accuracy is to be expected with an

increase in N because when the number of possible classifications increases, there is still only one classification which satisfies ground truth, but the number of incorrect classes increases. With an accuracy decrease of only 0.02463 between N=50 and N=75, and a decrease of 0.0496 between N = 75 and N=100 the model seems to handle increases of potential classes reasonably well. This is because the increases in potential classes grows at a greater factor than the accuracy rating decreases.

The following is a table depicting the accuracy, precision, recall, and F1-score for N=50, N=75 and N=100.

	ACCURACY	PRECISION	RECALL	F1-Score
N=50	0.26345	0.26345	0.29905	0.26870
N=75	0.23882	0.23882	0.25794	0.23353
N=100	0.18923	0.18923	0.20638	0.18023

The table shows how our metrics of classification get worse as the number of speakers increases. These scores assume UTF-8 encoding works. If it does not work, the scores will be lower as the encoding ISO-8859-1 will be used.

When the actual parameters of the model are considered, in this case, the number of hidden layers and the numbers of neurons in those layers, we can consider the impact of model structure versus input size. Generally deeper neural networks can have greater accuracy, but in the case of the case of the committee speakers with N = 50, there was a significant decrease in the accuracy of our model with the addition of another hidden layer consisting of two neurons. The accuracy of this model with the extra layer on N= 50 was 0.1553, lower than the optimized model with N= 100. This supports that a well thought out network structure is arguably more important for accuracy than the number of potential classes in a classification problem. While this is wise to consider, good feature selection is what truly helps define a good model.

Conclusion

Throughout this project we balanced time constraints with improving accuracy and witnessed firsthand the challenges of natural language processing. For this set of data, K-Means seemed to be more effective - and certainly faster - in predicting committee. With a looser time constraint and more feature optimization, we could potentially fine-tune our models to attain higher overall accuracy.