Stat 154 Final Project

Group 5: Max Eddy (23658703), Julio Soldevilla (24614087), Samba, Njie (23075185), Matthew Weglicki (24585950)

Introduction

In response to the well documented controversy of 2015, when it was revealed that Hillary Clinton used personal email accounts on private servers while she was US Secretary of State, the State Department released thousands of pages containing these emails. Our project consists of having access to 3505 of these emails as our training set from Clinton's top 5 correspondents among the entire set of released emails. The purpose of this project is to utilize the machine learning techniques learned throughout the semester to find the technique that classifies the emails to the actual sender most accurately. The techniques used include Random Forests, Support Vector Machines, and K-Means Cluster Analysis and were used alongside feature selection in order to achieve the highest possible classification accuracy. We built our classification models in R and took advantage of the natural language processing packages tm and SnowballC.

Data Description and Feature Filtering

We began with a training set consisting of 3505 emails from 5 senders, which had the following sender distribution:

Sender	Number of Emails	Proportion of Training Set
1	685	19.54%
2	1023	29.19%
3	1241	35.41%
4	275	7.85%
5	281	8.02%

We initially processed the data by making use of the tm R package to convert the emails into lowercase, remove punctuation and numbers, and strip excess whitespace. Next, we proceeded to remove stop words, favoring the SMART information retrieval system over "en", since SMART contains 571 commonly occurring words vs the 173 found in en. This reduced the number of features we had in our raw dataset from 74020 terms to 36063. Next we implemented a stemming correction to the dataset, further reducing the number of terms to 26103. This concluded our initial processing of the training set, reduced our total data size from 19.6Mb to 16.9Mb and is summarized below:

Step	Total # of Features
Raw	74020
Stop Words	36063
Stemming	26103

Being aware that leaving a full word feature matrix that is too large could leave us susceptible to overfitting our supervised classifiers we decided to take one more step in filtering our features. We concluded this process by only including words in our full word feature matrix if they could be found in both the training and test sets of emails. In this way we narrowed down our number of features from 26103 to 8863.

Power Features

Along with optimizing the number of features we chose to train our models on, we decided to include some additional power features that allow us to identify the sender of the email. In most cases these power features were phrases that occurred a disproportionately high number of times for a particular sender. The way we identified these were by physically inspecting a sample of emails from each sender in the training set, choosing unusual phrases from those and plotting a histogram to display how often the chosen phrase appears for each sender. Phrases that produced a spike in the histogram for a particular sender but appeared a negligible amount of times for others were included as power features. The threshold for whether a spike was large enough or not was decided heuristically after observing hundreds of histogram distributions and set at 25. In such a way we were able to identify 14 power feature phrases that ultimately decreased our training error even after implementing feature selection for both our best N tree RF model, and our best SVM model.

Additionally, we chose to use email character length as an identifying power feature. With the median character length for all the emails being 904, the legitimacy of this power feature can be seen when we notice that the median character length for sender 4 emails is 587, and 4183 for sender 5 (the median character lengths of emails for senders 1, 2 and 3 were 859, 1278, and 701, respectively). Therefore, it is reasonable to assume that this power feature could be particularly effective in identifying emails from senders 4 and 5.

A summary of the power features we used is found below:

Power Features	Description
"peace talks", "private residence", "situation room", "en route", "conference room", "mini ", "house select benghazi comm", "condfidential reason", "human rights", "thank you", "prime minister", "white house", "gordon brown", "tony blair"	Phrases found to be particularly effective in identifying specific senders
email_chars	Number of characters in each email

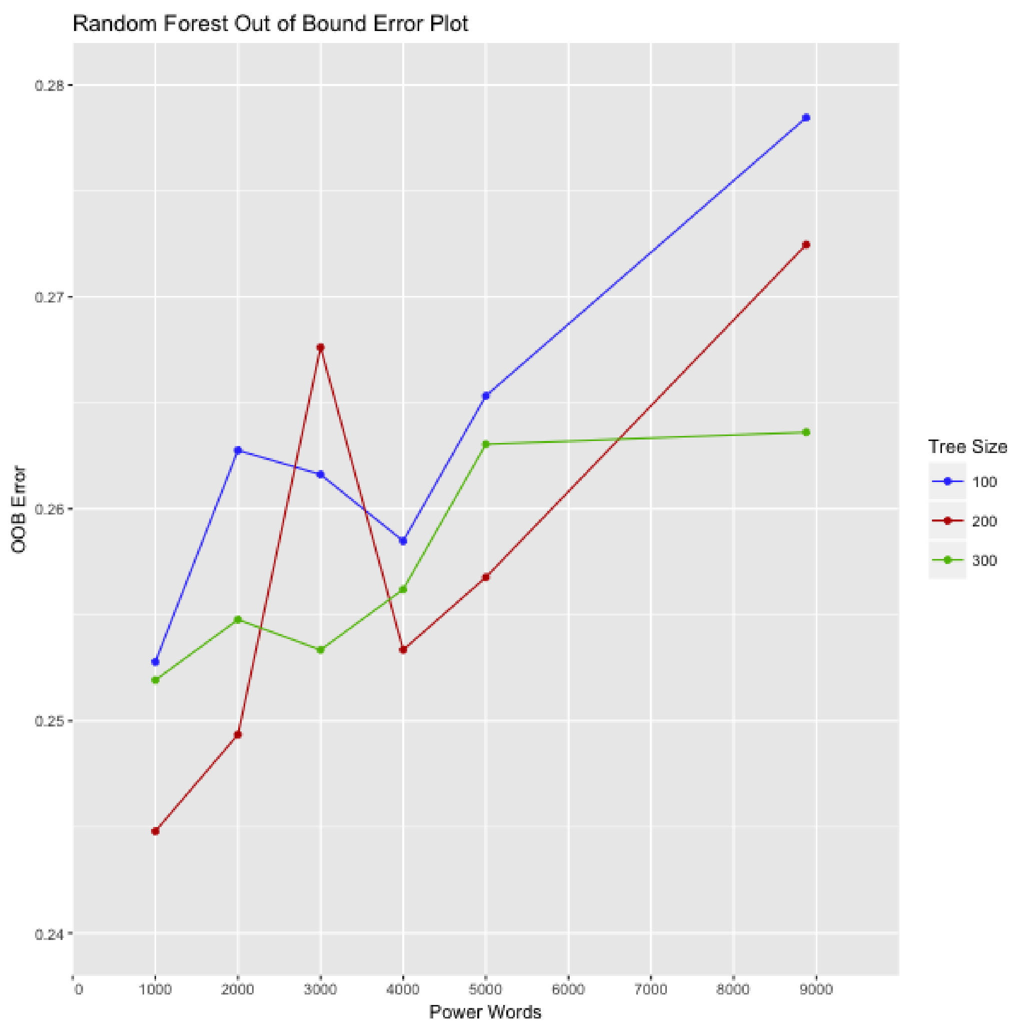
Random Forest Classifier

After finalizing our final full feature matrix including power features (now containing a total of 8878 features), we proceeded to train a random forest classifier, our first supervised learning method used to predict the senders for each email. The benefits of this tree-based method are that results are easily interpretable and the method can manage qualitative predictors more easily than regression methods.

The random forests algorithm takes in a training set, and samples B bootstrap samples as prescribed. From each of these samples, we generate a decision tree by using a subset of our total predictors in each bootstrapped sample at each branch to classify which side of the branch an observation falls on. In practice, this subset is p/3 for a regression algorithm and square root of p for a classification algorithm, as is the case for this project. We use a subset of predictors at each branch because this allows us to decorrelate the B trees produced by bootstrapping, generating a more group of trees. Once we have a “forest” of trees, we train a method on each *b*th bootstrapped tree, and average all the predictions.

To streamline the process of obtaining the OOB error and prediction matrices for the random forest method, we created a function that when given the body of our training emails, the senders associated with those emails and our full word feature matrix will output these values directly for any specified size of the forest. When we obtained these accuracy metrics we then created a variable that lists the features for that given size of forest in order of most to least important. By indexing the first k elements of this new variable we were able to rerun our user defined function and obtain OOB errors and prediction matrices for subsets of our full word feature matrix composed of the k most important features. We did this in 1000 word increments stopping at a word feature matrix length of 5000, noticing that increasing the number of word features beyond this point was likely to only increase our error rate. We ran this process over forest sizes of 100, 200, and 300 (stopping there since 200 trees appeared to yield a significantly smaller OOB error rate than 300).

Our findings for determining the best random forest classifier are displayed graphically below:



We can see that for most of our data points a 200 tree size yields the best OOB error rates and therefore we conclude that our best general RF model trained on a full word feature matrix is a random forest consisting of size 200 trees. Furthermore, we see that our overall lowest error rate occurs within the forest containing 200 trees when we use a word feature matrix composed of the 1000 most important features. Therefore, it is clear that our 1000 feature selected RF model is our best overall RF model.

It should be noted that the out-of-bag error is similar to cross-validation in that there exist observations from the training set which are traditionally not chosen by the bootstrapped samples that were used to generate decision trees, and these out-of-bag observations can be used to predict a response for the *i*th observation using trees in which that observation was out of bag. This is essentially similar to leave-one-out cross-validation, and since the OOB errors were used to fit observations that were not fit on the bootstrap samples, we are essentially using this to tune our models as we would with a test set. One may say that 5-fold cross-validation may produce an error rate more indicative of the test error, but we favor the OOB error due to the computational infeasibility of the CV error as compared to the OOB. Therefore, from a cost-benefit perspective using OOB to predict the observations maximizes computational efficiency, accuracy trade off.

The numerical values for the total accuracy and accuracy per sender obtained by running our general and overall best RF models can be seen in the table below:

Step	Total # Features Used	Total Accuracy	Accuracy per sender class (sender1, sender 2, sender3, sender4, sender5)
RF	8878	72.75%	37.37%, 80.94%, 85.50%, 64.63%, 81.14%
RF Feature Selection	1000	75.52%	44.67%, 83.48%, 84.77%, 66.55%, 89.68%

Lastly, our RF analysis showed that the top 10 most important features were: “sid”, email_chars, “cdm”, “fyi”, “subject”, “cdstategov”, “part”, “unclassified”, “state”, and “date”.

Support Vector Machine Classifier

Besides doing Random Forest classification, we applied the method of Support Vector Machine (SVM) to classify the emails to the corresponding senders. In comparison to the Random Forest approach, with SVM we have several models we can use, for instance we can use different kernels: linear, radial and polynomial and within each kernel we can alter some parameter. This freedom to alter our parameters allows us to explore more models to decide which one is giving the lowest test error.

We trained different SVM classifiers varying the parameters each time. We started training SVM classifiers with a linear kernel and made changes on the cost parameter, using costs 0.000001, 0.00001, 0.0001, 0.00011, 0.00099 for example. We decided to use these costs because if we used higher ones, then the margins would be too narrow and we would have few supporting vectors on or violating the margin. This would result in error rates that are extremely small, suggesting that the issue lay in having few supports. Additionally, when we used the costs 0.000001 and 0.00001, we saw error rates larger than 100%. We believe these error rates were generated by the fact that with such small costs, the margins would be too wide and thus most of our relevant points would be between the margins. This would imply that the computational machine would consider most of the points we are interested as being errors and thus generate the error rates larger than 100%.

Additionally, we trained the SVM classifier using polynomial and radial kernels. However, for the values gamma = 1, 2 and 5, and costs = 0.1, 1 and 10 for radial kernel we saw error rates greater than 100% as was also the case when we tried polynomial kernels of degree = 2 and 3 and cost = 0.1, 1 and 10. Once again, we believe that the reason we saw these error rates bigger than 100% was because with the described parameters there were too many data points within the margins causing a numerical error.

In determining our best models, we paid particular attention to varying the parameters of the linear kernel, having discounted other kernels due to the rampant errors observed when testing the classifiers mentioned above. For our general case model we used a data frame that contained our full word feature matrix consisting of 8878 features. Then for our feature selection case, we started using subsets of the full word feature matrix, increments of the 1000 most important features. We noticed that between 1000 and 2000 features we maximized our accuracy without overfitting. Going after further accuracy, we started looking at increments of 100, until we noticed that we obtained the highest accuracy with 1700 features.

Below is a summary of the results of our best SVM models. In both cases we found the best cost parameter to be C=0.0001.

Step	Total # Features Used	Total Accuracy	Accuracy per sender class (sender1, sender 2, sender3, sender4, sender5)
SVM	8878	53.18%	96.35%, 65.40%, 5.96%, 77.45%, 63.35%
SVM Feature Selection	1700	52.95%	96.20%, 64.91%, 5.64%, 77.55%, 62.28%

Unsupervised Classifier – Kmeans Clustering

In order to assess the performance of an unsupervised method in grouping emails from like senders, we fit three k-means clustering models to the data. To fit the model, we selected the top 100 most important features according to the top-performing random forest model, namely the one which utilized 1000 total features and used 200 trees in the model estimation. In order to ensure proper estimation of the clustering model, we set the maximum number of iterations for the model at 50, allowing the observations ample opportunity to settle into a particular cluster. While increasing the number of iterations will not necessarily boost the performance of the clustering model, it will allow the model to come to its best conclusion regarding the cluster assignments given the data. Additionally, we implemented the K-Means clustering algorithm with 50 different initial assignments, in order to identify the clustering that minimizes the within-cluster variation (in terms of the observations' squared Euclidean distances from one another).

In other settings, it is impossible to use the K-Means algorithm to formulate class predictions for a data set. However, because we know the senders from whom each of these emails came, we can formulate such predictions by using the clustering algorithm on the word feature matrix. One issue arises in making these predictions, however: the class to which the clustering algorithm assigns an observation does not necessarily correspond to the sender defined by the same class name (i.e. just because kmeans() assigns an email to Class 1 does not mean that it predicts that the email was sent by Sender 1). To resolve this issue, we implemented Algorithm K-1:

Algorithm K-1

1. Implement the K-Means Clustering algorithm on the 100-feature matrix and attain cluster assignments for each observation

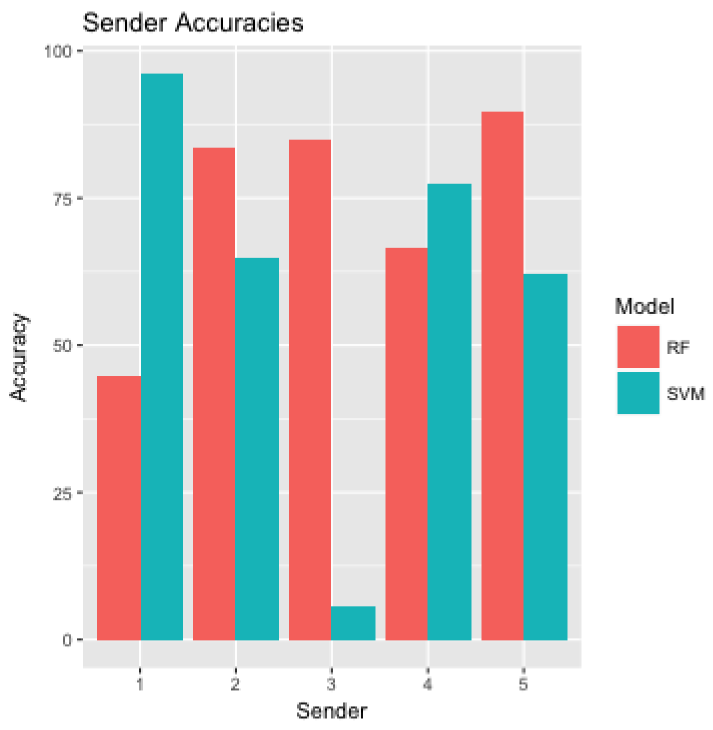
2. Calculate all possible permutations of the 5 classes to which the clustering algorithm assigned observations
3. For each permutation calculated in (2):
    a. Assign Cluster 1 observations to the first element of the permutation, Cluster 2 observations to the second element, etc.
    b. *Assume the new cluster assignments constitute attempts to predict the sender of each observation*, and calculate the sender prediction error rate

The first clustering model simply fit the K-Means model to the unaltered 100-feature matrix. The fit for this model was rather poor: the best configuration of cluster assignment yielded a 59.3% error rate. The primary reason for this is that the observations were primarily assigned to a single cluster - roughly 80% of the observations were classified as Sender 3 emails.

One possible reason for the model's poor performance that the variables in the top 100 random forest features have different levels of scale. In particular, the email size variable used in the feature matrix has much larger values than the other variables in the data set (the median of this variable is 904, compared to the many variables whose median is 0). Because the kmeans() function calculates the distance between two points as the Euclidean distance, variables with generally higher levels may have more leverage in the model. To solve this issue, we removed the email size variable from the list and attempted to standardize the other variables. The first standardization method attempted was to divide each row of the word feature matrix by the number of characters in the corresponding email to prevent emails with more words from dominating the clustering algorithm. The second was to calculate the standardized value for each word in the feature by subtracting the mean appearances per email and dividing by the standard deviation of that value. However, each of these methods produced error rates that were higher than the original. This suggests that the cause of the model's poor performance may have been the inability of the Euclidean Distance to capture distance effectively in high-dimensional space.

Supervised Model Comparisons

Having identified our best RF and SVM models separately we can now see how the two supervised methods compare with each other. Below we have plotted a histogram to make the comparison:



What we can see from this is that although there is not a single model that dominates accuracy over all senders, our RF model wins in the majority of sender categories and experiences less variation in its accuracy across senders. This coupled with the fact that the RF model has a significantly larger total accuracy of 75.52% compared to the best SVM model total accuracy of 53.18% leads us to conclude that our final best model for email classification is comes from the RF method.

Test Predictions

We can now state that our ultimate email classifier is a random forest containing 200 trees based on a word feature matrix of the 1000 most important words according to our analysis. We have provided an R script that produces our test predictions for the test set of emails according to this classifier and we now patiently await the accuracy scores of those predictions.

Step	Total # Features Used	Total Accuracy	Accuracy per sender class (sender1, sender 2, sender3, sender4, sender5)
200 Tree RF	1000		

Conclusion

The final model selected for implementation on our test dataset was ultimately chosen to be a random forest method which had been tuned to contain 200 trees and train on a word feature matrix consisting of 1000 of the most important features for that method. Through this model we were able to attain a training set accuracy of 75.52%. This project was critical in displaying the different applications of supervised and unsupervised learning models in a classification setting. Although support vector machines tend to have higher predictability rates, the random forests model proved to be more accurate in this case, and benefits from being graphically highly interpretable too. Thus, we have found that random forests models is capable of leveraging both accuracy and interpretability. We also learned numerous techniques in optimizing the preprocessing of the data set, and how to obtain power features that are unique to each email sender. We learned that system runtime is also a factor that needs to be optimized with large datasets, which we achieved through brute force, searching for unique power features, and abstracting functions that can search for unique words per sender. Overall, this project was highly informative and taught us how to put into practice various machine learning techniques we have learned about throughout the semester.