University of California, Berkeley

Statistics 154

Modern Statistical Prediction
and Machine Learning

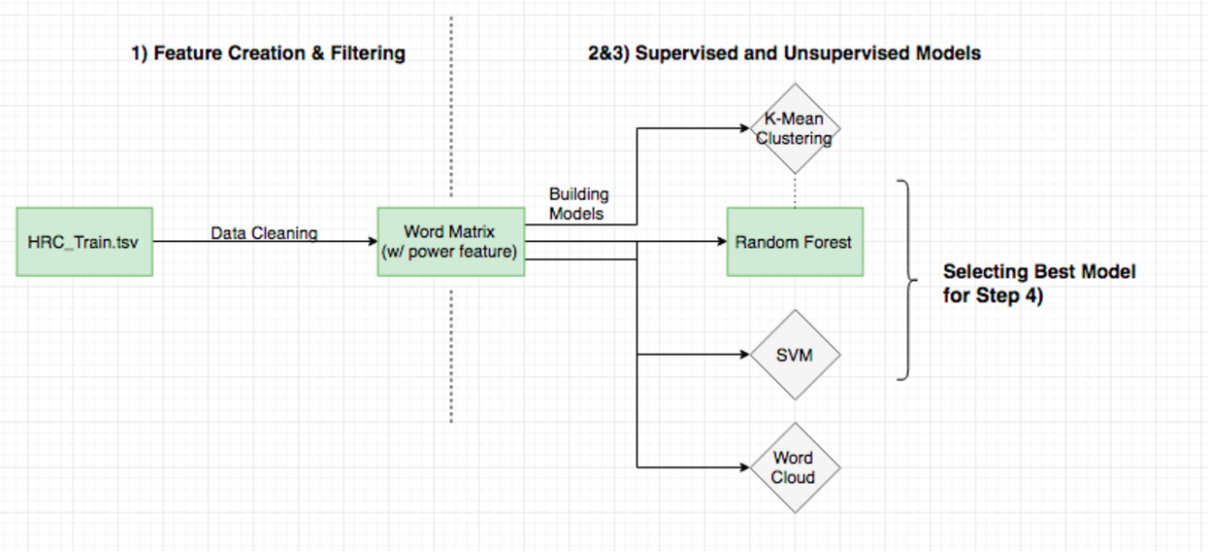
Unclassified Released Emails of HRC

Group 10: Bayes for Days
Pui Fung Lam,
Yishan Han,
Yunshan Guo,
Yvonne Zhou

December 2, 2016

Introduction

In 2015, controversies arose over a leak showing that HRC used her personal email accounts on private servers while she was the Secretary of State. In response to multiple FOI lawsuits, the State Department released thousands pages of redacted HRC emails.We were given the dataset HRC train.csv which contains a training set of 3505 emails from top 5 contacts among the full email's content. Below is the general layout of the project[1]:



1. Feature Creation and Filtering

1.1 Description for Data Set

The given training data set [HRC_train.tsv] is a compilation of 3505 emails and the two columns include the classifier for senders numbered by [1, 2, 3, 4, 5] and the contents for each email. *Table 1* below gives brief summary of the data set:

Class Label	Number of Emails	Percentage by Sender
1	685	19.54%
2	1023	29.19%
3	1241	35.41%
4	275	7.85%
5	281	8.02%

Table 1: Email Files Summary Statistics

[1]∗We are grateful for Professor Rabbee and Graduate Student Instructor Omid Shams Solari's help and support.

In this project we are tasked to create a classifier that classifies emails according to their senders, class variable, using words or phrases, features, used in each email. We will investigate and tune multi-class email classification model using three machine learning algorithms: Random Forest, Support Vector Machine and K-means clustering. We will explore our methods for feature selection and engineering to find the most optimal feature matrix. In building both supervised and unsupervised classifier on the training set, we predict the classes of test set using models that return the lowest error rate. Finally, we will also discuss the difficulties we have encountered and possible improvement methods for our model.

1.2 Text Cleaning and Corpus Processing

After reading the file and storing it in a dataframe, we proceed to extract the contents in R. To preprocess the raw data, we used R **tm package**. First, we converted the data frame into a vector corpus. Using tm_map function, we changed all the words to lowercase, removed whitespace and punctuations in all emails. Furthermore, we removed common stop words given by the generic english stop words list. However, the list was quite limited and basic, so we proceeded to select some new stopwords that were common in all emails, such as "unclassified", "us", "department", "state", "case" etc. Initially, we removed numbers from our corpus. However, after performing power feature selection by manually browsing through various emails, we found out that numbers play a large role in determining the sender. For example, some senders use more numbers than others as they refer to the time more. Therefore, we decided to include numbers in our corpus. We will provide all the stop words that we removed in the Appendix A. We also performed word stemming as our word feature was very large, so by stemming, we were able to decrease the dimensions of the word feature and increase the word feature overlap rate between training set and test set. After initial text cleaning process, we created a Document-Term matrix with about 1854 word features.

Since the word feature matrix was very large, computationally expensive to run, we decided to decrease the number of features based on a cut-off of frequency less than 50. *Figure 1* shows a collection of randomly selected 2000 words with its frequency of appearance on the y-axis. We can see that the frequency of these words highly ranges from 1-1250. The words that appear approximately 0-50 times are relatively insignificant as compared to those that appear more frequently. Therefore, we decided to cut off words that appear less than 50 times. By performing this step of downsizing our word feature matrix, we compromised some accuracy for efficiency.

In the final data pre-processing step, we converted our Document-Term matrix into a dataframe for easier model training.

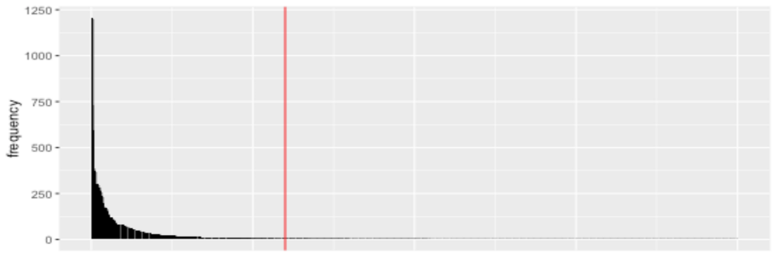


Figure 1: Frequency of words cut off

1.3 Stopword Selection

Just like there are words that appear too few times, there are also words that appear in every email and are therefore relatively insignificant in categorizing senders. For example, all emails begin with phrases, "unclassified u.s. department of state case no. *** doc no. *** date: *** state dept, " so we categorized these words as stop words since they have zero effect on our prediction. We also removed single-letter words as they are not meaningful. Other stop words we included are common daily words such as "will", "can", "also", "just", "say", as well as common email words such as "subject", "sent", "original". The full list of stopwords is included in the Appendix.

1.4 Power Feature

1) Length of email: After the data cleaning process, we recognized that some particular senders write significantly longer emails than others. So we included the length of each email to build our models. Since adding the raw word count into the feature matrix can offset the other features, we decided to normalize the word count with the scale() function.

2) Manually read through the emails by each sender to find unique characteristics that each sender exhibits. In other words, senders used different semantics and punctuations that allowed us to categorize each. For example, sender 4 mainly sent emails regarding scheduling, so most emails included phrases such as, "mini schedule" or "10:15am", "11:00am" etc. There were also other unique ways of spelling words such as "pls" or "thx" by individual senders. Another example from sender 5 would be that many of his emails ended with advertisements, such as "$400 deal!". In total, we found 28 power features.

3) To distinguish the time "am" from first-person singular present indicative of be, a two-digit number in front of "am" was enforced in our regex.

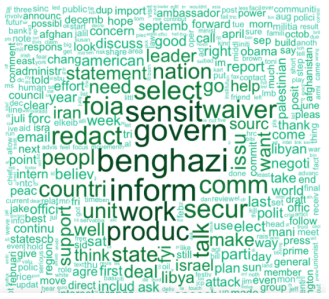
1.5 Data Description Report

Step	Total # of features
Raw	74,020 (including numbers, punctuations etc)
Remove Stop words	71416 (including numbers, punctuations etc)
Stemming	61416 (including numbers, punctuations etc)
Remove white spaces, punctuation	41181 (including numbers)
Adding power features	+ 29 features. 41210

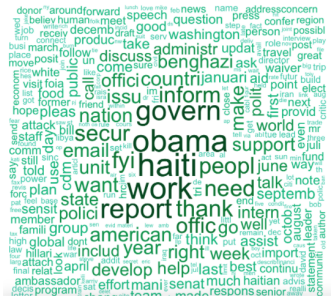
Table 2: Summary Statistics of Each Cleaning Step

1.6 Optional: Wordcloud

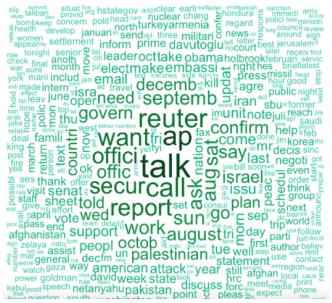
Below, we created word clouds for senders that are based on the frequency of the words they use.



(a) Sender 1

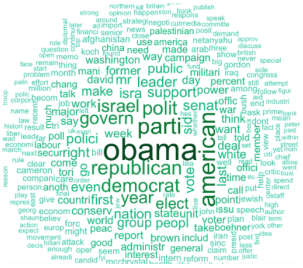


(b) Sender 2



(c) Sender 3

(d) Sender 4



(e) Sender 5

Figure 2: Wordcloud

2. Supervised Model

2.1 Random Forest Model

In the supervised model process, we created 3 random forest model: 1) with only word feature matrix, 2) with only power features, 3) with both the word feature ma-

trix and power features. This serves as exploratory analysis to see the improvements for each step.

Step	Total # of features	Total Accuracy	Total Accuracy (OOB)
1)	1854	77.36%	73.43%
2)	29	64.61%	59.66%
3)	1883	79.82%	76.28%

Table 3: Random Forest Model Comparison

2.1.1 Optimal Tuning Parameters

1) For making random forest models, we need to specify two parameters, mtry (1) and ntree (2). In this section, we will justify how we chose our optimal tuning parameters. Mtry: the number of variables available for splitting at each tree node. For classification models, the default is the square root of the number of predictor variables (rounded down). Random forests using mtry = $\sqrt{p}$ leads to a reduction in both test error and OOB error. I.e. int(sqrt(1889))=mtry=43

2) Ntree: number of trees to grow. Larger number of trees produce more stable models and covariate importance estimates, but require more memory and a longer run time. For accuracy, we decided on 500 and using more will marginally decrease error rate as depicted in the graph below.

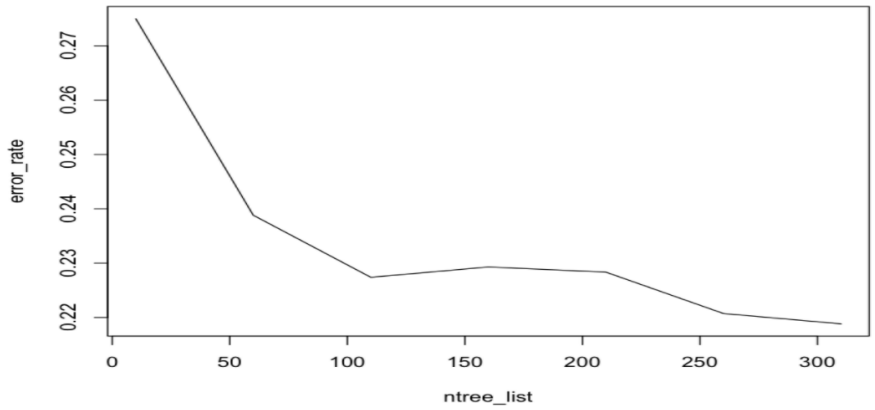


Figure 3: Graph showing number of trees grown vs error rate of random forest

Step	Accuracy per Sender Class 1, 2, 3, 4, 5				
1)	51.53%	78.55%	80.88%	66.50%	83.94%
2)	16.36%	66.90%	76.50%	41.00%	86.53%
3)	52.51%	81.81%	82.95%	65.50%	89.64%

Table 4: Random Forest Model Accuracy per Sender Class

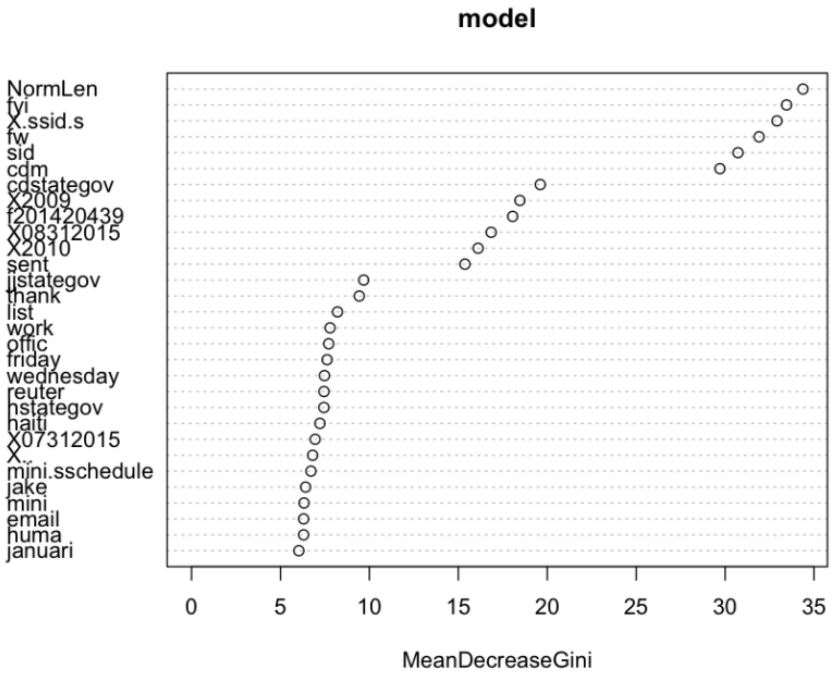


Figure 4: Random Forest Features by Gini Index

2.1.2 Random Forest: Top 10 Important Features

"NormLen"	"fyi"	"X.ssid.s"	"fw"	"sid"
"cdm"	"cdstategov"	"X2009"	"f201420439"	"X08312015"

2.2 SVM

2.2.1 Kernel Selection

Tuning function from package caret is used for shrinking our search scope for the possible optimal values of Cost, Gamma and Degree. 5-fold cross validation is implemented to evaluate the accuracy of models with different parameters. Below is a table on SVM Kernel Performance and radial kernel gives the highest accuracy among the three models.

Kernel	**Accuracy (%)**	**Best Cost**	**Best Gamma**	**Degree**
Linear	61.18	10	0.0005313	NA
Polynomial	54.00	1000	0.0005314	3
Radial	72.70	100	0.0005	NA

Table 5: Comparison of SVM Model with Different Kernel

2.2.2 Parameter Tuning

Since radial kernel consistently gives higher accuracy than the other two, we decide to use this model to further tune the parameters. We assign six candidates to cost and gamma separately, thus there are 36 pairs. Below is a plot of overall accuracy rate against various cost. When cost equals to 100 and gamma is 0.0005, it presents the highest accuracy, which is 72.7%.

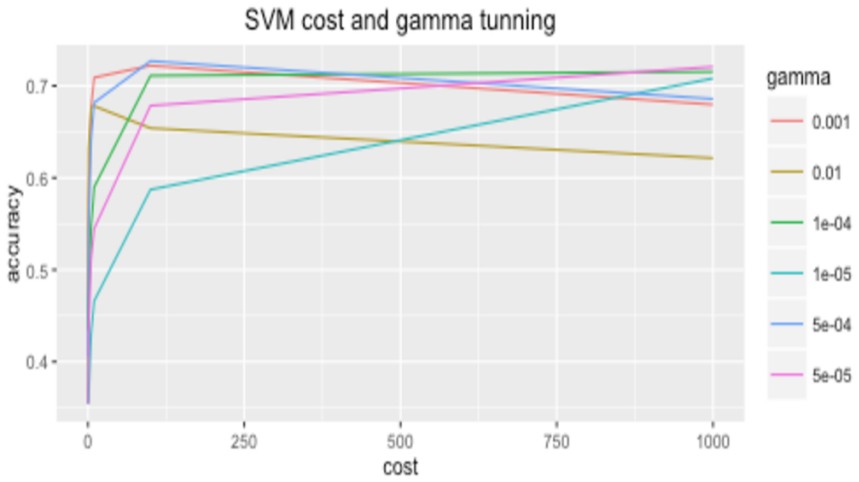


Figure 5: SVM cost and gamma tuning

2.2.3 Final Model for Support Vector Machine

SVM model with	Word feature	Power feature	Word + power feature
Error rate	33.02%	38.82%	27.3%

Table 6: Summary of SVM models with different features

Feature to use	Word + Power feature
SVM-Type	c-classification
SVM-Kernel	radial
Cost	100
Gamma	5e-04
Number of support vectors	1727

Table 7: Optimal SVM Model Summary Statistics

	y_validation				
y_hat	1	2	3	4	5
1	186	5	21	1	1
2	7	267	8	1	1
3	19	25	345	12	3
4	1	1	0	78	0
5	0	0	0	0	69

Table 8: Confusion Matrix of Test Set (Cross Validation)

2.2.4 Multi-class ROC with Optimal SVM Model

Since there are five classes, we plot the ROC for each class (one versus all) based on results from the optimal SVM model.

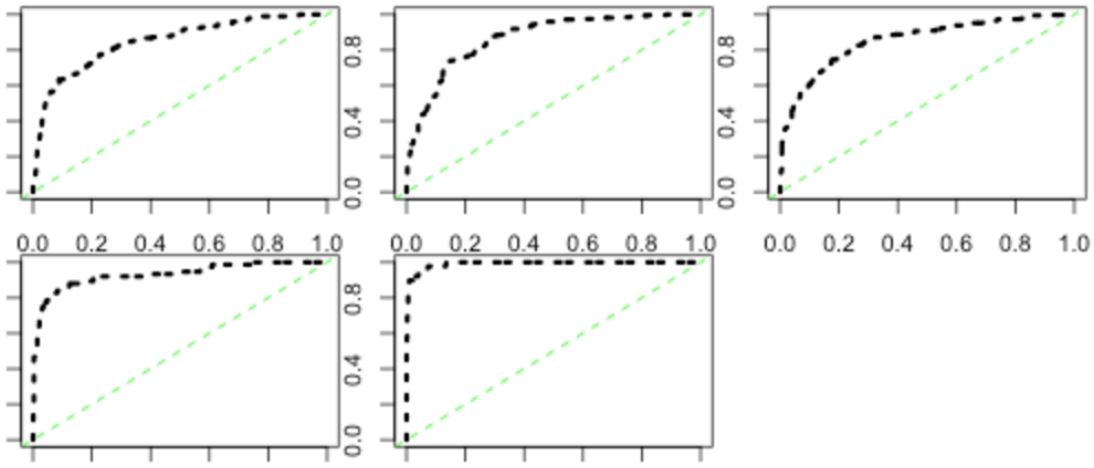


Figure 6: ROC Curve for Sender 1-5

3. Unsupervised Learning with K-means Clustering

3.1 K-means Analysis

K- means clustering is a good unsupervised learning approach for partitioning a data set into K distinct, non-overlapping clusters. In this project, top 100 important word features from random forest model are selected to perform clustering. By comparing results with the given true labels, the importance of features are be evaluated and justified.

Sender 1	Sender 2	Sender 3	Sender 4	Sender 5
19.54%	29.19%	35.41%	7.85%	8.02%

Table 9: Percentage of Different Sender Based on True Labels

Group 1	Group 2	Group 3	Group 4	Group 5
52.64%	41.14%	0.17%	4.25%	1.79%

Table 10: Percentage of Different Group Based on K-means Prediction

3.2 K-means Evaluation

Sender 1	Sender 2	Sender 3	Sender 4	Sender 5	Accuracy
Group 4	Group 2	Group 1	Group 3	Group 5	35.82%

Table 11: K-means result summary

From tables above, a group would be classified to specific sender based on similar ratio. However, in this case, the predicted ratio is very different from the true data and conclusion only can be made based on the same rank. Group 1 may come from sender 3, group 2 may come from sender 2, group 4 might be sender 1, group 3 might be sender 4 and group 5 could be sender 5. Overall, K-means with using top 100 words from Random Forest model does not present a good estimation, since when comparing true percentages and the predictions, there exists significant differences and sender 3 and sender 4 are not very distinguishable because of a relevantly close percentages. Thus, this time K-means does good in clustering but with only use of top 100 features, this method could have a low accuracy.

4. Validation for Test Set

We concluded that the Random Forest model that uses both word and power feature matrix is the best supervised classifier model.

Test set was converted the data frame into a vector corpus at first. As what we did before using tm_map function, we changed all the words to lowercase, removed whitespace and punctuations in all emails. Then instead of filtering some words with low frequency, we convert the corpus to word matrix directly and matching the colnames with training set's word matrix, if there are some words not in the test set and does show in the training set, we need manually add it in to make sure they have the same word and power features. Then fit the test word matrix into optimal model and use predict() to get final labels.

Step	Total # of features used	Total Accuracy(xx%)	Accuracy per sender class
RF	1883		

Table 12: Prediction Summary Statistics

5. Discussion

5.1 Technical Difficulty in Data Processing

The nltk package in Python and tm package in R each has its own stemming method and stopword list. For best results, in the beginning, we used both to process the raw data into sparse word feature matrices. Both matrices were used to fit the supervised models. We finally decided to go with R tm package it gives higher accuracy rate.

However, while using R, there was an issue with word completion after word stemming. Since word stemming did not drastically reduce the size of the feature matrix, we decided to omit word completion in this process.

5.2 Uncertainty of which stopwords to take out

One of the recurring issue in this project is figuring out what data to feed into our supervised models in order to get the best predictions. If we falsely classify part of the data as stop words but in retrospect, they can be used as reliable features, that could lower our accuracy rate. The team resolved this issue via trials and errors by running numerous models.

5.3 Removing Numbers

We deliberately skipped this step in our data cleaning process because some of the numbers actually actually behave as important features. (Refer to Gini Index Graph in 2)) Some of these numbers represent date, year or case number that are unique to certain senders For example, 2009, 201420439, 08312015 and 07312015 all appeared on the variable importance plot. That implies that these number have high Gini Indexes.

6. Conclusion

In this project, we explored various supervised and unsupervised models for classifying senders based on the contents of their emails. By implementing these machine learning models, we were able to systematically classify emails to a satisfactory level of accuracy. Through cleaning raw data and performing feature engineering, we optimized our models, with Random Forest as the most accurate of all. Our SVM model gave good accuracies as well, with K-means clustering the least accurate. From this project, we learnt to analyze large and complex raw data sets and build models to

make better prediction output. Furthermore, we learnt the importance of correctly choosing word and power features as they significantly impact our final prediction accuracy.

Reference

[1] Feinerer, Ingo. "Introduction to the Tm Package Text Mining in R." N.p., 3 July 2015. Web. 30 Nov. 2016.
[2] Galili, Tal. "Intro to Text Analysis with R." *R-bloggers*. N.p., 26 Jan. 2016. Web. 20 Nov. 2016.
[3] Jurka, Timothy P., Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, and Wouter Van
[4] Atteveldt. "RTextTools: A Supervised Learning Package for Text Classification." *The R Journal* 5.1 (2013): 6-12. Web. 20 Nov. 2016.
[5] M. Miah, Improved k-NN Algorithm for Text Classification, DMIN (2009) 434-440.
[6] l.Wang, X. Li, An improved KNN algorithm for text classification, 2010
[7] Wesley. "Text Mining." *R-bloggers*. N.p., 15 Oct. 2012. Web. 20 Nov. 2016.
[8] "Text Mining in R and Python: 8 Tips To Get Started." R-bloggers. N.p., 06 Oct. 2016. Web.15 Nov.2016.
[9] Williams, Graham. "Hands-On Data Science with R Text Mining." N.p., 10 Jan. 2016. Web. 20 Nov.2016.

Appendix

Stopwords given by tm package:

[1] "i" "me" "my" "myself" "we" "our" "ours" "ourselves"
[9] "you" "your" "yours" "yourself" "yourselves" "he" "him" "his"
[17] "himself" "she" "her" "hers" "herself" "it" "its" "itself"
[25] "they" "them" "their" "theirs" "themselves" "what" "which" "who"
[33] "whom" "this" "that" "these" "those" "am" "is" "are"
[41] "was" "were" "be" "been" "being" "have" "has" "had"
[49] "having" "do" "does" "did" "doing" "would" "should" "could"
[57] "ought" "i'm" "you're" "he's" "she's" "it's" "we're" "they're"
[65] "i've" "you've" "we've" "they've" "i'd" "you'd" "he'd" "she'd"
[73] "we'd" "they'd" "i'll" "you'll" "he'll" "she'll" "we'll" "they'll"
[81] "isn't" "aren't" "wasn't" "weren't" "hasn't" "haven't" "hadn't" "doesn't"
[89] "don't" "didn't" "won't" "wouldn't" "shan't" "shouldn't" "can't" "cannot"
[97] "couldn't" "mustn't" "let's" "that's" "who's" "what's" "here's" "there's"
[105] "when's" "where's" "why's" "how's" "a" "an" "the" "and"
[113] "but" "if" "or" "because" "as" "until" "while" "of"
[121] "at" "by" "for" "with" "about" "against" "between" "into"

[129] "through" "during" "before" "after" "above" "below" "to" "from"
[137] "up" "down" "in" "out" "on" "off" "over" "under"
[145] "again" "further" "then" "once" "here" "there" "when" "where"
[153] "why" "how" "all" "any" "both" "each" "few" "more"
[161] "most" "other" "some" "such" "no" "nor" "not" "only"
[169] "own" "same" "so" "than" "too" "very"

Stopwords we selected

[1] "unclassified" "us" "department" "state" "case" "doc" "date"
[8] "dept" "subject" "sent" "am" "pm" "re" "fw"
[15] "a" "b" "c" "d" "e" "f" "g"
[22] "h" "i" "j" "k" "l" "m" "n"
[29] "o" "p" "q" "r" "s" "t" "u"
[36] "v" "w" "x" "y" "z" "monday" "tuesday"
[43] "wednesday" "thursday" "friday" "saturday" "sunday" "today" "tomorrow"
[50] "yesterday" "bb" "bbd" "das" "will" "release" "can"
[57] "also" "now" "just" "get" "like" "let" "according"
[64] "full" "said" "message" "part" "know" "agreement" "say"
[71] "full" "original" "one" "two" "back" "call" "clinton" "foreign"
[79] "house" "may" "meeting" "minister" "new" "november" "president" "secretary"
[87] "see" "time"

Random Forest Top 100 Features

[1] "fyi" "X.ssid.s" "fw" "sid" "cdm" "cdstategov"
[7] "X2009" "f201420439" "X08312015" "sent" "X2010" "jjstategov"
[13] "hstategov" "thank" "reuter" "list" "friday" "offic"
[19] "work" "wednesday" "X.." "X07312015" "huma" "jake"
[25] "email" "mini" "haiti" "tuesday" "januari" "thursday"
[31] "confirm" "monday" "statement" "add" "need" "talk"
[37] "pleas" "attach" "pis" "april" "mini.sschedule" "want"
[43] "secretari" "call" "number" "octob" "rout" "X06302015"
[49] "depart" "cell" "inform" "report" "saturday" "sunday"
[55] "best" "good" "june" "schedul" "X..en.sroute" "pls"
[61] "nov" "ask" "X...1" "speech" "thu" "b5b6"
[67] "septemb" "oct" "march" "hillari" "offici" "sun"
[73] "discuss" "aug" "make" "oper" "august" "sat"
[79] "mon" "decemb" "parti" "posit" "fri" "well"
[85] "polici" "thx" "secur" "juli" "polit" "nora"
[91] "ive" "room" "say" "draft" "address" "look"
[97] "issu" "question" "tri" "wed"

Power Features

[1] "am" "pm"
[3] "monday" "tuesday"
[5] "wednesday" "thursday"
[7] "friday" "saturday"
[9] "sunday" "today"
[11] "tomorrow" "yesterday"
[13] "sent" "thx"
[15] "pls" "fw"
[17] "X.." "X...1"
[19] "X...2" "deal"
[21] "libya" "X.ssid.s"
[23] "special.sassistant.sto.ssecretary" "mini.sschedule"
[25] "X..en.sroute" "sent.svia.scingular.sxpress.smail.swith.sblackberry"
[27] "sent.sfrom.smy.sverizon.swireless.sblackberry" "copyright.sthe.sfinancial"