Text Classification in R

19/04/2020

1 Loading All Packages

```
library(stringr)
library(tidytext)
library(readr)
library(dplyr)
library(ggplot2)
```

For now, we shall be using these packages. Later, we shall be needing some more packages related to various classification algorithm.

2 The Dataset

For this text classification, we shall be using US Presidential Speeches Dataset (https://github.com/kfogel/presidential-speeches) by Kary Fogel (https://github.com/kfogel) in his github respository. I have gone ahead and downloaded the repository locally, and then unzipped it.

All the speeches till today 2020-04-24 18:21:35 is available in his repository, within the data folder, and we are going to analyze this rich dataset.

```
speeches <- list.files('../datasets/presidential-speeches/')
head(speeches)
```

```
[1] "1789-04-30-first-inaugural-address.txt"
[2] "1789-10-03-thanksgiving-proclamation.txt"
[3] "1790-01-08-first-annual-message-congress.txt"
[4] "1790-12-08-second-annual-message-congress.txt"
[5] "1790-12-29-talk-chiefs-and-counselors-seneca-nation.txt"
[6] "1791-10-25-third-annual-message-congress.txt"
```

As you can see, each filename starts with a date identifying the date when the speech is given.

Let us read one file to see how it looks like.

```
tmp <- read_lines('../datasets/presidential-speeches/1789-04-30-first-inaugural-add
  ress.txt')
tmp[1:5]
```

```
[1] "President: George Washington"
[2] ""
[3] "Fellow Citizens of the Senate and the House of Representatives:"
[4] ""
[5] "Among the vicissitudes incident to life, no event could have filled me with gr
eater anxieties than that of which the notification was transmitted by your order,
and received on the fourteenth day of the present month. On the one hand, I was sum
moned by my Country, whose voice I can never hear but with veneration and love, fro
m a retreat which I had chosen with the fondest predilection, and, in my flattering
hopes, with an immutable decision, as the asylum of my declining years: a retreat w
hich was rendered every day more necessary as well as more dear to me, by the addit
ion of habit to inclination, and of frequent interruptions in my health to the grad
ual waste committed on it by time. On the other hand, the magnitude and difficulty
of the trust to which the voice of my Country called me, being sufficient to awaken
in the wisest and most experienced of her citizens, a distrustful scrutiny into his
qualification, could not but overwhelm with dispondence, one, who, inheriting infer
ior endowments from nature and unpractised in the duties of civil administration, o
ught to be peculiarly conscious of his own deficencies. In this conflict of emotion
s, all I dare aver, is, that it has been my faithful study to collect my duty from
a just appreciation of every circumstance, by which it might be affected. All I dar
e hope, is, that, if in executing this task I have been too much swayed by a gratef
ul remembrance of former instances, or by an affectionate sensibility to this trans
cendent proof, of the confidence of my fellow-citizens; and have thence too little
consulted my incapacity as well as disinclination for the weighty and untried cares
before me; my error will be palliated by the motives which misled me, and its conse
quences be judged by my Country, with some share of the partiality in which they or
iginated. "
```

As you can also see, each speech starts with “President:” followed by the name of the president as first line. Therefore, we can take use of that to make a dataframe containing the speeches, the corresponding president name and the date of the speech.

For the first job of extracting name of the dates, we shall use `substr` function.

```
speech_dates <- as.Date(substr(speeches, 1, 10))
```

Now, we create a list as long as the number of speeches, and we use a for loop to fill up that list by reading the text of the speech and with variables like president name and party names for references. Although a for loop is not usually recommended in `R`, but using it here makes it clear of our aim.

```
speech_list <- vector(mode = "list", length = length(speeches))  # create a blank l
  ist

for (i in 1:length(speeches)) {
    tmp <- read_lines(paste0('../datasets/presidential-speeches/', speeches[i]))
    speech_list[[i]] <- tibble(line = 2:length(tmp), text = tmp[2:length(tmp)])  #
   we do not take the first line
    speech_list[[i]]$president <- substring(tmp[1], 12)   # Take it from 12 charact
  er onwards
    speech_list[[i]]$date <- speech_dates[i]
}

speech_df <- bind_rows(speech_list)  # finally bind all rows of
```

Now we can take a look at the data.

```
speech_df[1:1000, ]   # prints out only first 1000 rows
```

line <int>
2
3
4
5
6
7
8
9
10
11

1-10 of 1,000 rows | 1-1 of 4 columns Previous **1** 2 3 4 5 6 … 100 Next

Since we see there are lots of blank rows we need to remove them. Also, I shall also need the affiliated party names for each of the presidents. For this, I have used the list provided by Wikipedia (https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States) and used a bit of web scrapping to build a dataset containing this.

```
pres_df <- readRDS('../datasets/US-president.Rds')
pres_df
```

date <date>	party <chr>	president <chr>
1789-04-30	Unaffiliated	George Washington
1797-03-04	Federalist	John Adams

date <date>	party <chr>	president <chr>
1801-03-04	Democratic-Republican	Thomas Jefferson
1809-03-04	Democratic-Republican	James Madison
1817-03-04	Democratic-Republican	James Monroe
1825-03-04	Democratic-Republican	John Quincy Adams
1829-03-04	Democratic	Andrew Jackson
1837-03-04	Democratic	Martin Van Buren
1841-03-04	Whig	William Henry Harrison
1841-04-04	Whig	John Tyler

1-10 of 45 rows Previous 1 2 3 4 5 Next

The next job is to merge this two dataframes.

```
speech_df <- speech_df %>% filter(text != "") %>%
    left_join(pres_df[, c(2, 3)], by = c("president" = "president"))

speech_df[1:1000, ]
```

line <int>
3
5
7
9
11
13
15
17
3
5

1-10 of 1,000 rows | 1-1 of 5 columns Previous 1 2 3 4 5 6 … 100 Next

3 Preprocessing

3.1 Cleaning the texts

Here, the first thing to do is to lowercase all the letters, so that we have a consistency in the data.

```
speech_df <- speech_df %>% mutate(text = str_to_lower(text) )
```

The very next thing would be to remove all special symbols, punctuations, numbers etc. But before that, since this is a speech data, it would contain the usage of apostrophe.

For example, “n’t” needs to be regarded as “not”, “’ve” to be regarded as “have” etc. Also, we need to remove the salutations like “mr.”, “mrs.” etc. Once we replace these specific patterns, we can remove all characters except the alphabetical characters.

```
# the next thing is processing apostrophe
patterns <- c(
    "n't" = " not",
    "'ve" = " have",
    "'ll" = " will",
    "'m" = " am",
    "'re" = " are",
    "this's" = "this is",
    "that's" = "that is",
    "it's" = "it is",
    "'s" = "",
    "mr." = "",
    "mrs." = "",
    "ms." = "",
    "sr." = ""
)

speech_df <- speech_df %>%
    mutate(text = str_replace_all(string = text, pattern = patterns ) ) %>%
    mutate(text = str_replace_all(text, pattern = "[^a-zA-Z\\s]", replacement = " "
  ) )
```

3.2 Creating Word Tokens and Calculating tf-idf

In this part, we shall split the words in the texts to create a dataframe, where each row corresponds to an appearence of a word in a speech. `unnest_tokens()` function in `tidytext` package helps us to do that.

```
words_df <- speech_df %>% unnest_tokens(word, text)
```

Also, there are “stop words” which are most common words like article, prepositions which are extensively used all over English literature. Hence, removing these words would create a meaningful dataset which only contains the words representing the particular nature of the documents.

```
data("stop_words")
words_df <- words_df %>% anti_join(stop_words)
```

Finally, we the term document matrix, as a dataframe. Note that, here each of the speech works like a document, and this is identified by the date of the speech. Then, we calculate tf-idf based on this dataframe.

```
term_df <- words_df %>% count(word, date)  # create the term document dataframe
term_df <- term_df %>% bind_tf_idf(word, date, n)
```

Finally, corresponding to each data, we shall attach the president name and the affiliated party. This would prepare the dataframe for classification analysis.

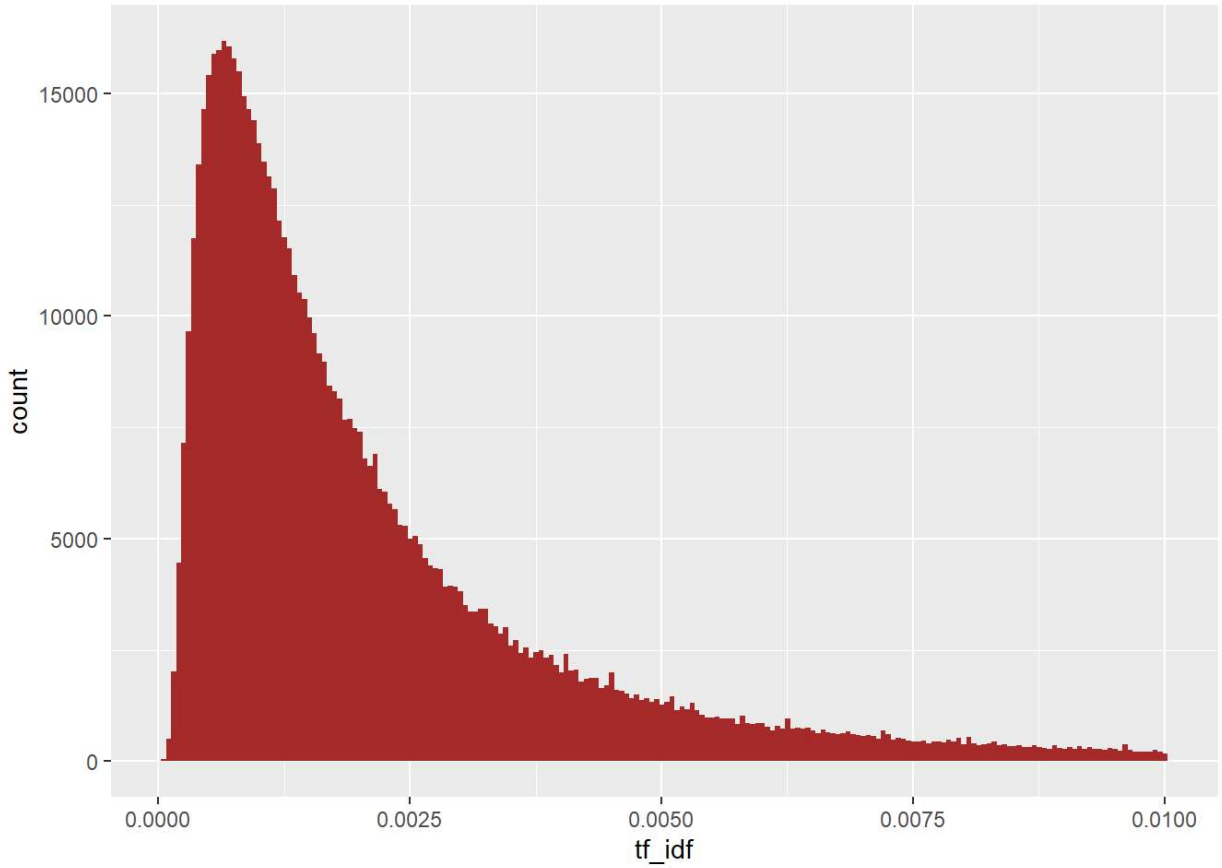
```
tmp <- speech_df %>% group_by(date, president, party) %>% summarise()   # create a
   dataframe with speech date, president, party
term_df <- term_df %>% left_join(tmp, by = c("date" = "date"))

term_df[1:1000, ]   # the first 1000 rows of the dataframe
```

word <chr>	date <date>	n <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>	president <chr>
aaa	1937-03-09	1	6.877579e-04	6.880384	0.0047320386	Franklin D. Roosevelt
aana	1894-12-03	4	3.164557e-04	6.880384	0.0021773367	Grover Cleveland
aaron	1807-01-22	2	2.288330e-03	5.270946	0.0120616617	Thomas Jefferson
aaron	1807-10-27	1	1.177856e-03	5.270946	0.0062084172	Thomas Jefferson
aaron	1870-06-13	1	1.005025e-03	5.270946	0.0052974333	Ulysses S. Grant
aaron	1870-07-14	1	4.559964e-04	5.270946	0.0024035322	Ulysses S. Grant
aaron	2000-01-27	2	5.873715e-04	5.270946	0.0030960036	Bill Clinton
aarp	2003-12-08	2	1.879699e-03	6.880384	0.0129330528	George W. Bush
ab	1971-05-20	2	1.818182e-02	6.880384	0.1250978924	Richard Nixon
aback	1977-05-22	1	8.130081e-04	6.880384	0.0055938082	Jimmy Carter

1-10 of 1,000 rows | 1-7 of 8 columns Previous **1** 2 3 4 5 6 … 100 Next

While none of the `tf_idf` is 0, the distribution of the `tf_idf` turns out to be as follows,

```
term_df %>% dplyr::filter(tf_idf < 0.01) %>%
    ggplot(aes(x = tf_idf)) + geom_histogram(fill = "brown", bins = 2e2)
```



4 Classification Analysis

4.1 Creating Training and Testing Sets

Before proceeding with classification analysis, let us first understand what we want to classify. Here, the input is a speech of some US president, and we wish to know whether the president is from Republican party or Democratic party, and we wish to see whether we can find out some particular words which are more important in identifying this classification.

Based on this, we filter out the party to be either “Republican” or “Democratic”.

```
term_df <- term_df %>% dplyr::filter(party %in% c("Republican", "Democratic"))
```

The speeches corresponding to these two party affiliated presidents comes from the following dates.

```
speech_dates <- unique(term_df$date)
length(speech_dates)
```

```
[1] 809
```

Now, we see there are 809 speeches. We create the training data corresponding to 609 speeches, which corresponds to about 0.75% of the total number of speeches, and the rest 200 speeches (about 0.25%) are left as testing dataset. We also see that the proportion of speeches coming from both the parties are on balance in both the training and testing set, so that the classification algorithm would not be very biased towards a particular party.

```
set.seed(2020)  # set a seed so that this is reproducible
trainIndex <- sample(speech_dates, size = 609)

# create training set
traindata <- term_df %>% filter(date %in% trainIndex)
traindata %>% group_by(party) %>% summarise(n = length(unique(date)))
```

party <chr>	n <int>
Democratic	324
Republican	286

2 rows

```
# create testing set
testdata <- term_df %>% filter(! (date %in% trainIndex))
testdata %>% group_by(party) %>% summarise(n = length(unique(date)))
```

party <chr>	n <int>
Democratic	119
Republican	81

2 rows

4.2 Creating Term Document Matrix and Labels

Firstly, we shall create a vector of affiliated party names, indexed by the speech dates. This can be later used to construct labels for the speeches in training and testing set.

```
party_names <- tmp$party
names(party_names) <- tmp$date
```

We need to create the term document matrix. For that, we shall convert the term document dataframe into a sparse matrix using `Matrix` package.

```
library(Matrix)
train_doc <- traindata %>% cast_sparse(date, word, n)
dim(train_doc)
```

```
[1]   609 30701
```

We see that, it contains 30701 words, and hence it is clear that we need to identify each document with a 30701 length vector, which constitutes the feature of the speech. It is extremely computationally intensive to work with these many features. We need to perform some dimensionality reduction technique first.

4.3 Using tf-idf based thresholding

To perform a simple dimension reduction technique, we use the following heuristic algorithm.

1. Group the dataframe into two parts, for two parties.
2. Compute average tf-idf for each word, averaged over all speeches, in each group.
3. For each group, find the top 100 words arranged according to tf-idf value in decreasing order.

```
train_red <- traindata %>%
    group_by(party, word) %>%
    summarise(mean_tf_idf = mean(tf_idf)) %>%
    group_by(party) %>%
    arrange(desc(mean_tf_idf)) %>% top_n(100)    # compute the top 100 words having
  highest mean_tf_idf

# filter both the traindata and testdata to contain only required words
red_traindata <- traindata %>% filter(word %in% train_red$word)
red_testdata <- testdata %>% filter(word %in% train_red$word)
```

We note that, there are only 197 unique words.

```
length(unique(train_red$word))
```

```
[1] 197
```

Now, using this reduced training and testing dataframe, we compute the new term document matrix from training set, as well as from testing data also. To ensure that all 197 words appear, we add a dummy document containing all of these 197 words.

```
red_words <- sort(unique(train_red$word))
red_traindata <- rbind(red_traindata, tibble(word = red_words, date = NA, n = 0, tf
  = 0, idf = 0, tf_idf = 0, president = NA, party = NA) )

train_doc <- red_traindata %>% cast_sparse(date, word, n)
train_doc <- train_doc[!is.na(rownames(train_doc)), red_words]
dim(train_doc)
```

```
[1] 277 197
```

We see that there are now 277 speeches, and 197 words in the training set. Similarly, in testing set, we have 76 speeches.

```
red_testdata <- rbind(red_testdata, tibble(word = red_words, date = NA, n = 0, tf =
  0, idf = 0, tf_idf = 0, president = NA, party = NA) )

test_doc <- red_testdata %>% cast_sparse(date, word, n)
test_doc <- test_doc[!is.na(rownames(test_doc)), red_words]

dim(test_doc)
```

```
[1]  76 197
```

Finally, we create the training and testing labels.

```
training_labels <- party_names[rownames(train_doc)]
testing_labels <- party_names[rownames(test_doc)]
```

To compare the performance of the different classification algorithms, we create an utility function as follows, which prints out a classfication report based on several classfication characteristics.

```
classification_report <- function(True_labels, Predicted_labels){
  # Making a function for the classification characteristics for a problem with lev
  els 1 and 0.
  # Here x and y are two vectors. x is the actual value while y is the vector of pr
  edicted values.

  class_table = table(Predicted_labels , True_labels)
  cat("Classification Matrix is (Row = Predicted, Column = Actual):\n\n")
  print(class_table)

  TP <- class_table[1, 1]; FP <- class_table[1, 2]; FN <- class_table[2, 1]; TN <-
  class_table[2, 2];

  Accuracy = (TP+TN)/sum(class_table)
  cat("\n Accuracy : ", Accuracy, "\n")
  cat("Classification error : ", 1 - Accuracy, "\n\n")

  preci = c(TP / (TP + FP), TN / (TN+FN) )
  recall = c( TP / (TP + FN), TN / (TN + FP) )
  F1 = 2 *(preci*recall)/(preci+recall)

  class_df <- data.frame(Class = rownames(class_table), Precision = preci, Recall =
  recall, F1.measure = F1)
  cat("Classification Report :\n")
  print(class_df)

  Pe = (TP+FP)*(TP+FN)/(sum(class_table)^2) + (TN+FP)*(TN+FN)/(sum(class_table)^2)
  cat("\n Cohen's Kappa: ", (Accuracy - Pe)/(1 - Pe) )

  }
```

4.3.1 Performance of Linear Discriminant Analysis

```
fit.LDA <- MASS::lda(train_doc, factor(training_labels))

# in training set
preds <- predict(fit.LDA, newdata = train_doc)
classification_report(training_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        128         33
      Republican         13        103

 Accuracy :  0.833935
Classification error :  0.166065

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.7950311 0.9078014  0.8476821
2 Republican 0.8879310 0.7573529  0.8174603

 Cohen's Kappa:  0.6668932
```

```
# in testing set
preds <- predict(fit.LDA, newdata = test_doc)
classification_report(testing_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         25         26
      Republican          8         17

 Accuracy :  0.5526316
Classification error :  0.4473684

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4901961 0.7575758  0.5952381
2 Republican 0.6800000 0.3953488  0.5000000

 Cohen's Kappa:  0.1438038
```

4.3.2 Performance of Naive Bayes Classifier

```
library(e1071)
fit.NB <- naiveBayes(x = as.matrix(train_doc), y = factor(training_labels))

# in training set
preds <- predict(fit.NB, newdata = as.matrix(train_doc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         69        107
      Republican         72         29

 Accuracy :  0.3537906
Classification error :  0.6462094

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.3920455 0.4893617  0.4353312
2 Republican 0.2871287 0.2132353  0.2447257

 Cohen's Kappa:  -0.2987663
```

```
# in testing set
preds <- predict(fit.NB, newdata = as.matrix(test_doc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         25         37
      Republican          8          6

 Accuracy :  0.4078947
Classification error :  0.5921053

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4032258 0.7575758  0.5263158
2 Republican 0.4285714 0.1395349  0.2105263

 Cohen's Kappa:  -0.09335038
```

4.3.3 Performance of SVM

We first use polynomial Kernels.

```
fit.SVM.poly <- svm(x = as.matrix(train_doc), y = factor(training_labels), kernel =
  "polynomial", cost = 3)

# in training set
preds <- predict(fit.SVM.poly, newdata = as.matrix(train_doc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         76          0
      Republican         65        136

 Accuracy :  0.765343
Classification error :  0.234657

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 1.0000000 0.5390071  0.7004608
2 Republican 0.6766169 1.0000000  0.8071217

 Cohen's Kappa:  0.5344779
```

```
# in testing set
preds <- predict(fit.SVM.poly, newdata = as.matrix(test_doc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic          7          7
      Republican         26         36

 Accuracy :  0.5657895
Classification error :  0.4342105

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5000000 0.2121212  0.2978723
2 Republican 0.5806452 0.8372093  0.6857143

 Cohen's Kappa:  0.05287009
```

Next, we use SVM with Gaussian (or radial basis) kernel.

```
fit.SVM.radial <- svm(x = as.matrix(train_doc), y = factor(training_labels), kernel
  = "radial", cost = 2)

# in training set
preds <- predict(fit.SVM.radial, newdata = as.matrix(train_doc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         95         14
      Republican         46        122

 Accuracy :  0.7833935
Classification error :  0.2166065

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.8715596 0.6737589  0.7600000
2 Republican 0.7261905 0.8970588  0.8026316

 Cohen's Kappa:  0.5684462
```

```
# in testing set
preds <- predict(fit.SVM.radial, newdata = as.matrix(test_doc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         17         14
      Republican         16         29

 Accuracy :  0.6052632
Classification error :  0.3947368

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5483871 0.5151515  0.5312500
2 Republican 0.6444444 0.6744186  0.6590909

 Cohen's Kappa:  0.1909155
```

Finally, using Sigmoid kernel.

```
fit.SVM.sigmoid <- svm(x = as.matrix(train_doc), y = factor(training_labels), kerne
  l = "sigmoid")

# in training set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(train_doc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        129         60
      Republican         12         76

 Accuracy :  0.7400722
Classification error :  0.2599278

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6825397 0.9148936  0.7818182
2 Republican 0.8636364 0.5588235  0.6785714

 Cohen's Kappa:  0.4767003
```

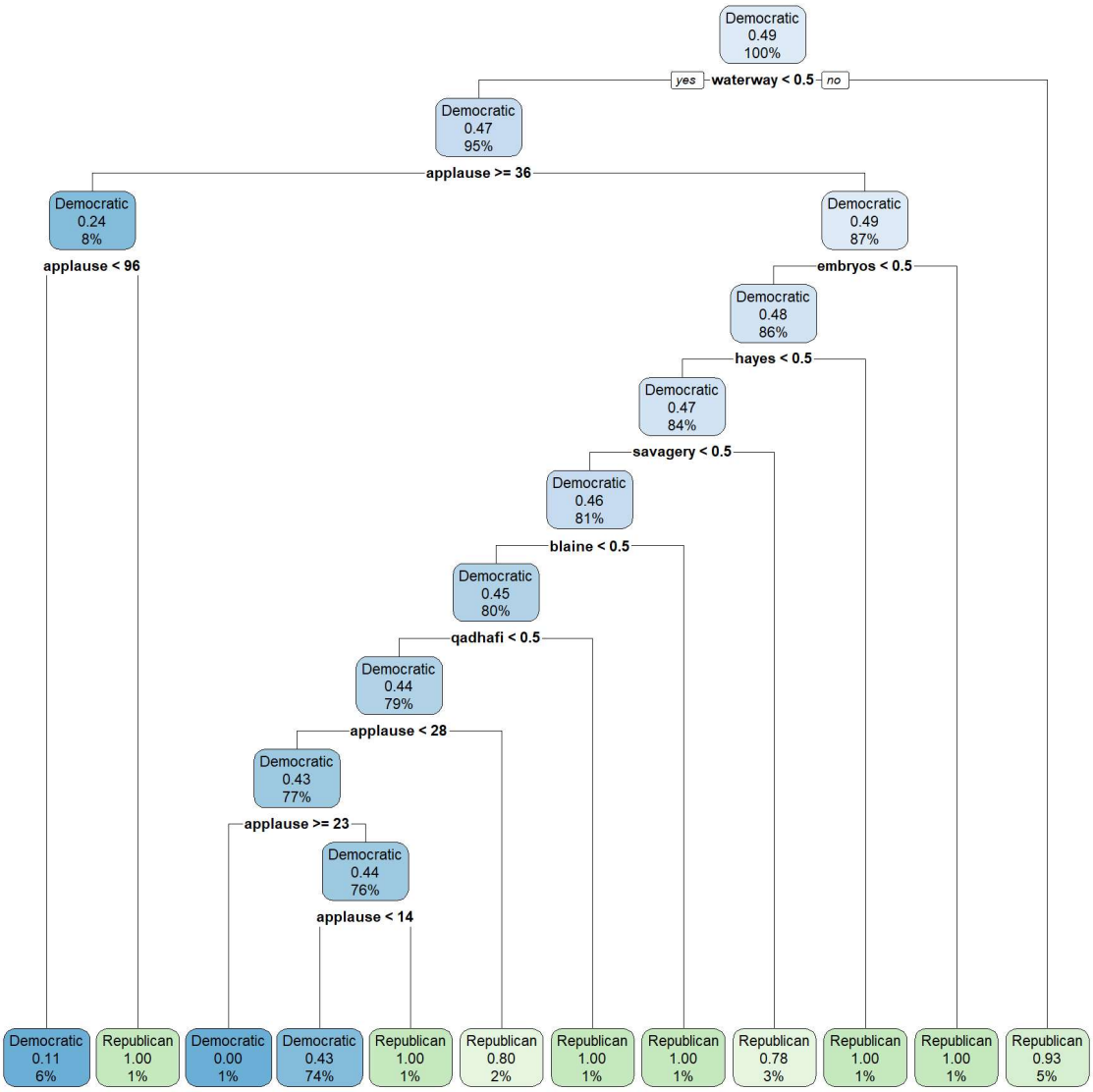
```
# in testing set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(test_doc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         23         28
      Republican         10         15

 Accuracy :  0.5
Classification error :  0.5

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4509804 0.6969697  0.5476190
2 Republican 0.6000000 0.3488372  0.4411765

 Cohen's Kappa:  0.04307488
```

4.3.4 Performance of CART

Now, we use Classification And Regression Tree (CART). The fitted decision tree looks as follows.

```
library(rpart)
library(rpart.plot)

fit.CART <- rpart(LABEL ~ . , data = cbind(as.data.frame(as.matrix(train_doc)), LAB
  EL = factor(training_labels)),
                  method = "class", minsplit = 10)
rpart.plot(fit.CART)
```




```
# in training set
preds <- predict(fit.CART, type = "class")
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

               True_labels
Predicted_labels Democratic Republican
      Democratic        137         90
      Republican          4         46

 Accuracy :  0.6606498
Classification error :  0.3393502

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6035242 0.9716312  0.7445652
2 Republican 0.9200000 0.3382353  0.4946237

 Cohen's Kappa:  0.3133801
```

```
# in testing set
preds <- predict(fit.CART, newdata = as.data.frame(as.matrix(test_doc)), type = "cl
  ass")
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         30         30
      Republican          3         13

 Accuracy :  0.5657895
Classification error :  0.4342105

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic    0.5000 0.9090909  0.6451613
2 Republican    0.8125 0.3023256  0.4406780

 Cohen's Kappa:  0.1930502
```

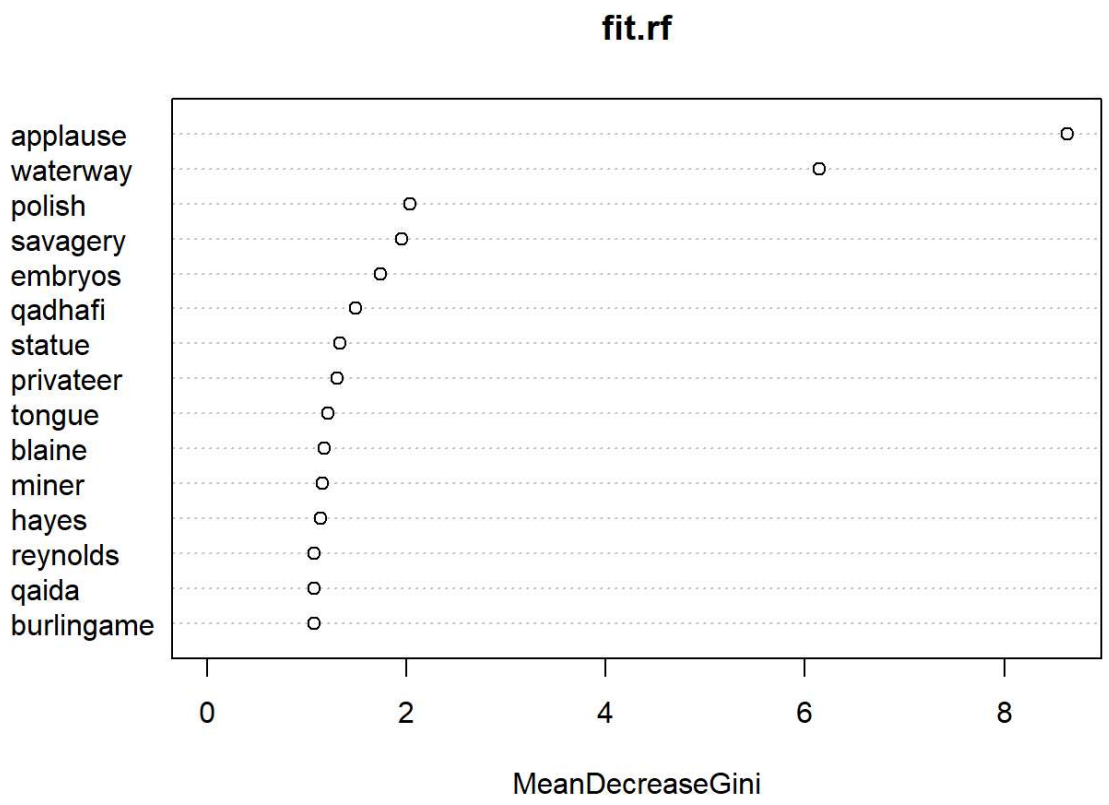
4.3.5 Performance of Random Forest

Finally, we try training Random Forest classifier. Although, since CART did not perform well, it seems that Random Forest will fail too.

```
library(randomForest)
fit.rf <- randomForest(x = as.matrix(train_doc), y = factor(training_labels))
```

Next, we wish to see which of the words it found to be important in discriminating.

```
varImpPlot(fit.rf, n.var = 15)
```




```
# accuracy in training set
preds <- predict(fit.rf)
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        108         96
      Republican         33         40

 Accuracy :  0.534296
Classification error :  0.465704

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5294118 0.7659574  0.6260870
2 Republican 0.5479452 0.2941176  0.3827751

 Cohen's Kappa:  0.0605726
```

```
# in testing set
preds <- predict(fit.rf, newdata = as.matrix(test_doc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         26         28
      Republican          7         15

 Accuracy :  0.5394737
Classification error :  0.4605263

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4814815 0.7878788  0.5977011
2 Republican 0.6818182 0.3488372  0.4615385

 Cohen's Kappa:  0.1272966
```

4.4 Using Latent Semantic Analysis

We first convert the train dataset from dataframe to a sparse term document matrix.

```
train_doc <- traindata %>% cast_sparse(date, word, n)
```

Now, we perform LSA on this training document term matrix. This would give us a representation; $X \approx U\Sigma V^T$, which is the singular value decomposition. Restricting Σ to be a $k \times k$ matrix, which is very low compared to number of documents or number of words. Then, we can create a new representations for each document using; $\hat{d}_i = \Sigma^{-1} V^T d_i$. This new representation becomes the features of the document.

To understand these features with the words, we link it using $\tilde{t}_i = \Sigma^{-1} U^T t_i$. These $\tilde{t}_i$'s are called Pseudo-terms, which are basically some linear combination of the words, possibly referring to different topics. For this, we shall choose 10 latent topics.

```
n_topic <- 20
lsa.fit <- svd(train_doc, nu = n_topic, nv = n_topic)  # obtain the SVD decompositi
  on
```

Before proceeding with the classification, let us see how these 20 topics are related with the terms. For this, we consider top 10 words with highest magnitude of coefficients in V matrix, corresponding to each topic.

```
rownames(lsa.fit$v) <- colnames(train_doc)

# just a simple for loop, using with apply
topic_mat <- apply(abs(lsa.fit$v), 2, function(x) {names(sort(x, decreasing = T)[1:
  10])} )
topic_mat <- t(topic_mat)  # transpose it so that row-wise we have topics, and colu
  mns are the words
rownames(topic_mat) <- paste("Topic", 1:n_topic)
colnames(topic_mat) <- paste("Word", 1:10)

as.data.frame(topic_mat)
```

	Word 1 <fctr>	Word 2 <fctr>	Word 3 <fctr>	Word 4 <fctr>	Word 5 <fctr>	Word 6 <fctr>
Topic 1	government	united	people	congress	country	time
Topic 2	people	president	america	world	government	applause
Topic 3	examination	service	classified	commission	person	examinati
Topic 4	slavery	missouri	slave	nebraska	compromise	principle
Topic 5	president	applause	united	america	soviet	business
Topic 6	president	gold	world	war	peace	nations
Topic 7	gold	united	public	world	notes	silver
Topic 8	applause	united	ve	mexico	treaty	national
Topic 9	power	public	treasury	constitution	bank	gold
Topic 10	congress	applause	america	law	business	president

1-10 of 10 rows | 1-7 of 11 columns

Now, we apply the required transformation, to obtain the transformed training and testing document matrix.

```
new_traindoc <- train_doc %*% lsa.fit$v %*% (diag(1 / lsa.fit$d[1:n_topic]))

# we add a dummy document containing all the words from training set
testdata <- testdata %>% filter(word %in% colnames(train_doc)) %>%
  rbind( tibble(word = colnames(train_doc), date = NA, n = 0, tf = 0, idf = 0, tf_i
  df = 0, president = NA, party = NA) )

test_doc <- testdata %>% cast_sparse(date, word, n)
test_doc <- test_doc[-nrow(test_doc), ]   # the last document is dummy one

new_testdoc <- test_doc %*% lsa.fit$v %*% (diag(1 / lsa.fit$d[1:n_topic]))

# finally add some column names
colnames(new_traindoc) <- colnames(new_testdoc) <- paste("Topic", 1:n_topic)
```

Now, we are ready to apply our classification algorithms once again, when we prepare the training and testing labels for these new set of documents.

```
training_labels <- party_names[rownames(train_doc)]
testing_labels <- party_names[rownames(test_doc)]
```

4.4.1 Performance of Linear Discriminant Analysis

```
fit.LDA <- MASS::lda(new_traindoc, factor(training_labels))

# in training set
preds <- predict(fit.LDA, newdata = new_traindoc)
classification_report(training_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        244        219
      Republican         79         67

 Accuracy :  0.5106732
Classification error :  0.4893268

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5269978 0.7554180  0.6208651
2 Republican 0.4589041 0.2342657  0.3101852

 Cohen's Kappa:  -0.01061389
```

```
# in testing set
preds <- predict(fit.LDA, newdata = new_testdoc)
classification_report(testing_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        108         73
      Republican         11          8

 Accuracy :  0.58
Classification error :  0.42

Classification Report :
       Class Precision     Recall F1.measure
1 Democratic 0.5966851 0.90756303       0.72
2 Republican 0.4210526 0.09876543       0.16

 Cohen's Kappa:  0.00720955
```

4.4.2 Performance of Naive Bayes Classifier

```
fit.NB <- naiveBayes(x = as.matrix(new_traindoc), y = factor(training_labels))

# in training set
preds <- predict(fit.NB, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         84         38
      Republican        239        248

 Accuracy :  0.545156
Classification error :  0.454844

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6885246 0.2600619  0.3775281
2 Republican 0.5092402 0.8671329  0.6416559

 Cohen's Kappa:  0.1222729
```

```
# in testing set
preds <- predict(fit.NB, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic          0          0
      Republican        119         81

 Accuracy :  0.405
Classification error :  0.595

Classification Report :
       Class Precision Recall F1.measure
1 Democratic       NaN      0        NaN
2 Republican     0.405      1  0.5765125

 Cohen's Kappa:  0
```

4.4.3 Performance of Support Vector Machine

We first use polynomial Kernels.

```
fit.SVM.poly <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), kerne
  l = "polynomial", degree = 20)

# in training set
preds <- predict(fit.SVM.poly, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        322        258
      Republican          1         28

 Accuracy :  0.5747126
Classification error :  0.4252874

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5551724 0.9969040  0.7131783
2 Republican 0.9655172 0.0979021  0.1777778

 Cohen's Kappa:  0.09995036
```

```
# in testing set
preds <- predict(fit.SVM.poly, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        119         81
      Republican          0          0

 Accuracy :  0.595
Classification error :  0.405

Classification Report :
       Class Precision Recall F1.measure
1 Democratic     0.595      1  0.7460815
2 Republican       NaN      0        NaN

 Cohen's Kappa:  0
```

Next, we use SVM with Gaussian (or radial basis) kernel.

```
fit.SVM.radial <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), ker
  nel = "radial", cost = 2)

# in training set
preds <- predict(fit.SVM.radial, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        296        167
      Republican         27        119

 Accuracy :  0.681445
Classification error :  0.318555

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6393089 0.9164087  0.7531807
2 Republican 0.8150685 0.4160839  0.5509259

 Cohen's Kappa:  0.3420836
```

```
# in testing set
preds <- predict(fit.SVM.radial, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        119         81
      Republican          0          0

 Accuracy :  0.595
Classification error :  0.405

Classification Report :
       Class Precision Recall F1.measure
1 Democratic     0.595      1  0.7460815
2 Republican       NaN      0        NaN

 Cohen's Kappa:  0
```

Finally, using Sigmoid kernel.

```
fit.SVM.sigmoid <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), ke
  rnel = "sigmoid")

# in training set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        157        170
      Republican        166        116

 Accuracy :  0.4482759
Classification error :  0.5517241

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4801223 0.4860681  0.4830769
2 Republican 0.4113475 0.4055944  0.4084507

 Cohen's Kappa:  -0.1084243
```

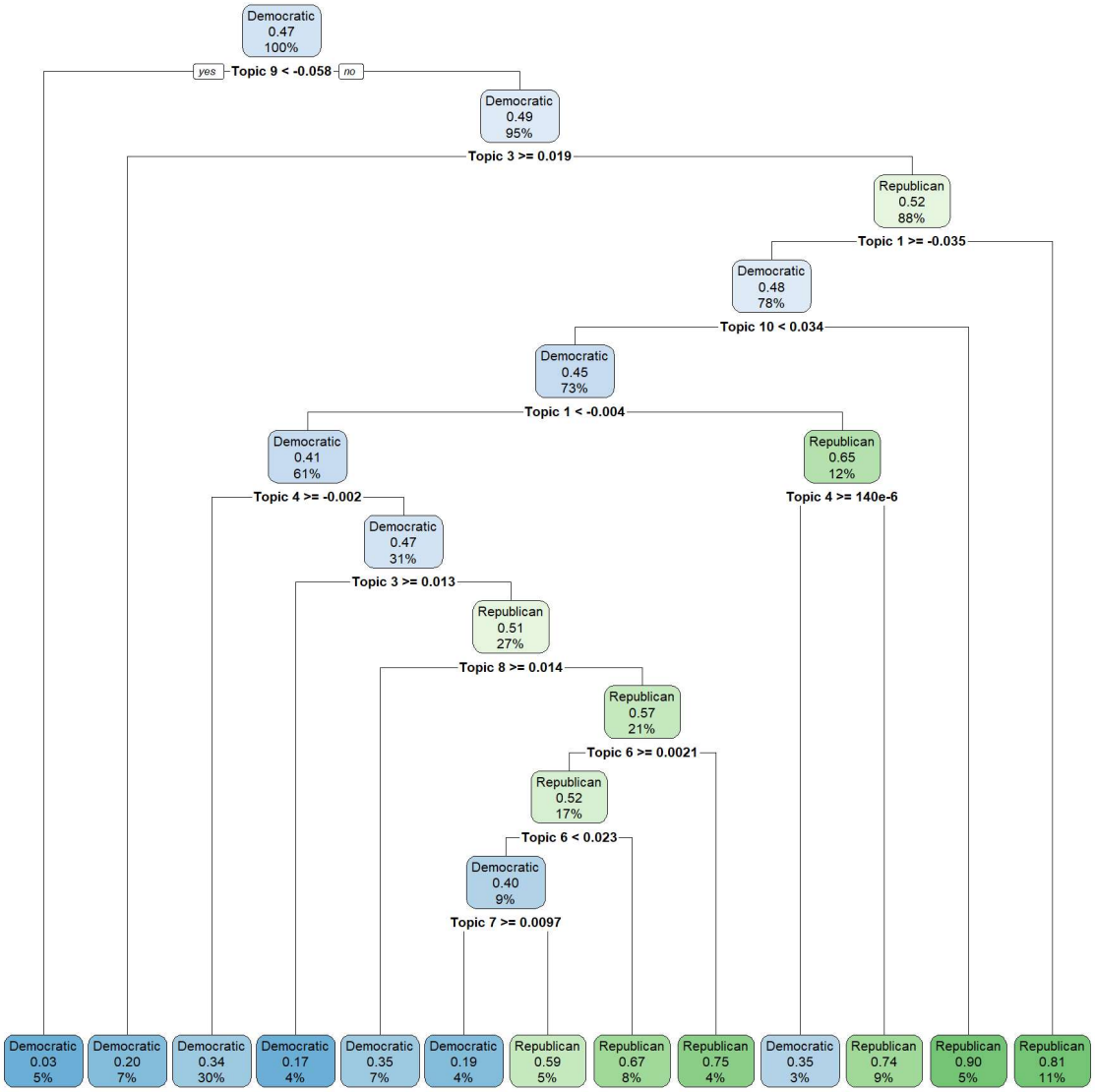
```
# in testing set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         21         19
      Republican         98         62

 Accuracy :  0.415
Classification error :  0.585

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic    0.5250 0.1764706  0.2641509
2 Republican    0.3875 0.7654321  0.5145228

 Cohen's Kappa:  -0.0502693
```

4.4.4 Performance of CART

Now, we use Classification And Regression Tree (CART). The fitted decision tree looks as follows.

```
fit.CART <- rpart(LABEL ~ . , data = cbind(as.data.frame(as.matrix(new_traindoc)),
  LABEL = factor(training_labels)),
                  method = "class", minsplit = 50)
rpart.plot(fit.CART)
```




```
# in training set
preds <- predict(fit.CART, type = "class")
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        260         99
      Republican         63        187

 Accuracy :  0.7339901
Classification error :  0.2660099

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic  0.724234 0.8049536  0.7624633
2 Republican  0.748000 0.6538462  0.6977612

 Cohen's Kappa:  0.4621315
```

```
# in testing set
preds <- predict(fit.CART, newdata = as.data.frame(as.matrix(new_testdoc)), type =
  "class")
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         44         33
      Republican         75         48

 Accuracy :  0.46
Classification error :  0.54

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.5714286 0.3697479  0.4489796
2 Republican 0.3902439 0.5925926  0.4705882

 Cohen's Kappa:  -0.03478011
```

4.4.5 Performance of Random Forest

Finally, we try training Random Forest classifier. Although, since CART did not perform well, it seems that Random Forest will fail too.

```
fit.rf <- randomForest(x = as.matrix(new_traindoc), y = factor(training_labels))

# accuracy in training set
preds <- predict(fit.rf)
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        237        111
      Republican         86        175

 Accuracy :  0.6765189
Classification error :  0.3234811

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6810345 0.7337461  0.7064083
2 Republican 0.6704981 0.6118881  0.6398537

 Cohen's Kappa:  0.3473734
```

```
# in testing set
preds <- predict(fit.rf, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic          3          1
      Republican        116         80

 Accuracy :  0.415
Classification error :  0.585

Classification Report :
       Class Precision     Recall F1.measure
1 Democratic 0.7500000 0.02521008 0.04878049
2 Republican 0.4081633 0.98765432 0.57761733

 Cohen's Kappa:  0.01048714
```

4.5 Using Latent Dirichlet Allocation

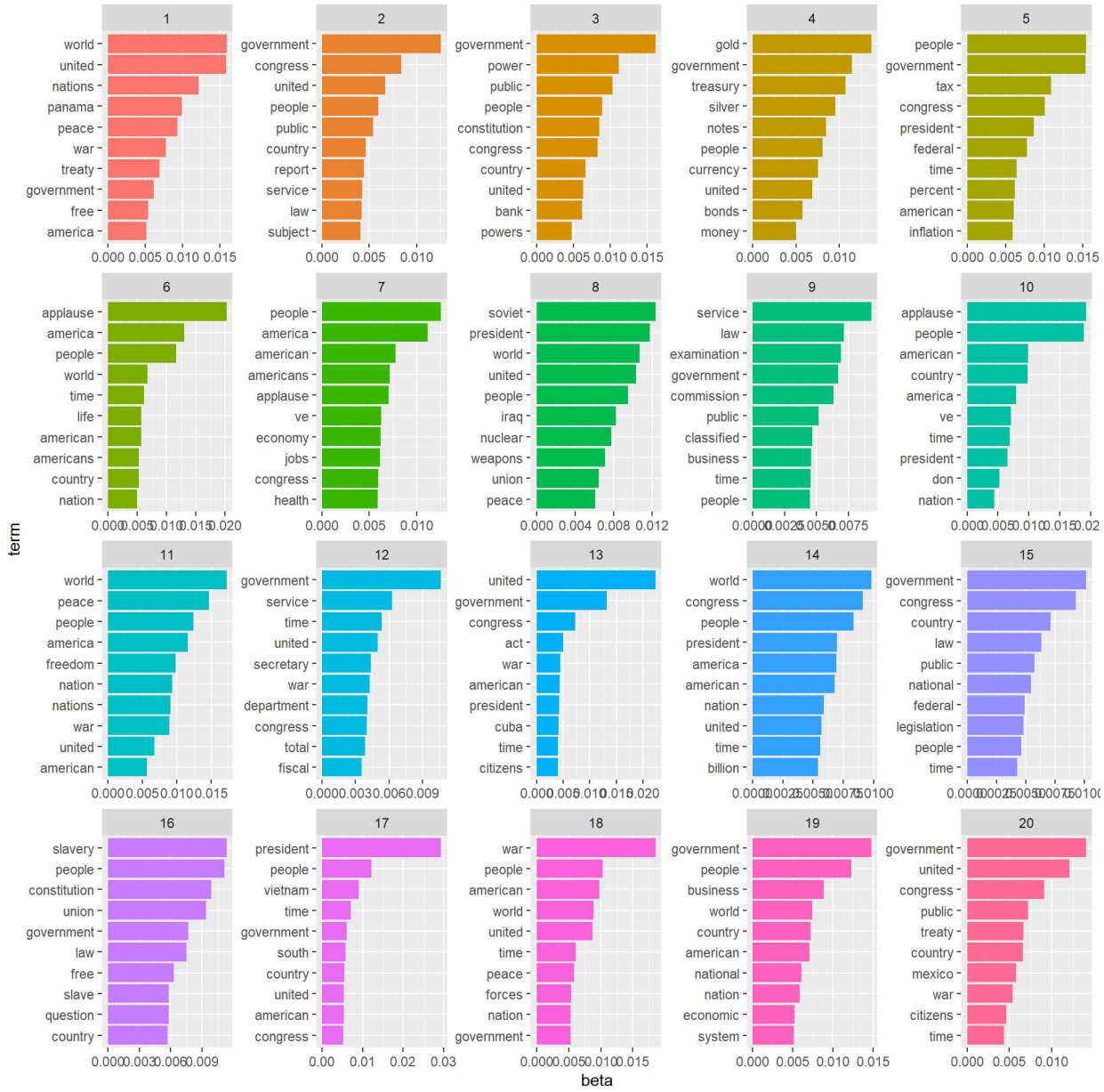
Now, we shall be using Latent Dirichlet Allocation for topic modelling, which would give us a lower level representation of each documents, in different topics. We shall be using `topicmodels` package in R for this. However, in this case, we are going to convert our document term matrix into the `DocumentTermMatrix` class presented in `tidytext` package.

```
library(topicmodels)
train_doc <- traindata %>% cast_dtm(date, word, n)
```

```
# set a seed so that the output of the model is predictable
mod.lda <- LDA(train_doc, k = n_topic, control = list(seed = 1234, verbose = 1))
```

Now, we find out the most important words related to each topics.

```
train_topics <- tidy(mod.lda, matrix = "beta")
train_topics <- train_topics %>% group_by(topic) %>% top_n(10, beta) %>% ungroup()
  %>% arrange(topic, -beta)


train_topics %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip() +
  scale_x_reordered()
```



Now we obtain the posterior probabilities of the topics for each document in the training and testing set, and use these probabilities as our feature for the document.

```
new_traindoc <- posterior(mod.lda)$topics

# we add a dummy document containing all the words from training set
testdata <- testdata %>% filter(word %in% colnames(train_doc)) %>%
  rbind( tibble(word = colnames(train_doc), date = NA, n = 0, tf = 0, idf = 0, tf_i
  df = 0, president = NA, party = NA) )

test_doc <- testdata %>% cast_dtm(date, word, n)
test_doc <- test_doc[-nrow(test_doc), ]   # the last document is dummy one

new_testdoc <- posterior(mod.lda, newdata = test_doc)$topics

# finally add some column names
colnames(new_traindoc) <- colnames(new_testdoc) <- paste("Topic", 1:n_topic)
```

Now, we are ready to apply our classification algorithms for one last time.

4.5.1 Performance of Linear Discriminant Analysis

```
fit.LDA <- MASS::lda(new_traindoc, factor(training_labels))

# in training set
preds <- predict(fit.LDA, newdata = new_traindoc)
classification_report(training_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        250        151
      Republican         73        135

 Accuracy :  0.6321839
Classification error :  0.3678161

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6234414 0.7739938  0.6906077
2 Republican 0.6490385 0.4720280  0.5465587

 Cohen's Kappa:  0.2499258
```

```
# in testing set
preds <- predict(fit.LDA, newdata = new_testdoc)
classification_report(testing_labels, preds$class)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         89         50
      Republican         30         31

 Accuracy :  0.6
Classification error :  0.4

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6402878 0.7478992  0.6899225
2 Republican 0.5081967 0.3827160  0.4366197

 Cohen's Kappa:  0.1359758
```

4.5.2 Performance of Naive Bayes Classifier

```
fit.NB <- naiveBayes(x = as.matrix(new_traindoc), y = factor(training_labels))

# in training set
preds <- predict(fit.NB, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        252        165
      Republican         71        121

 Accuracy :  0.6124795
Classification error :  0.3875205

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6043165 0.7801858  0.6810811
2 Republican 0.6302083 0.4230769  0.5062762

 Cohen's Kappa:  0.2071625
```

```
# in testing set
preds <- predict(fit.NB, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         92         61
      Republican         27         20

 Accuracy :  0.56
Classification error :  0.44

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6013072 0.7731092  0.6764706
2 Republican 0.4255319 0.2469136  0.3125000

 Cohen's Kappa:  0.02146114
```

4.5.3 Performance of Support Vector Machine

We first use polynomial Kernels.

```
fit.SVM.poly <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), kerne
  l = "polynomial", degree = 2)

# in training set
preds <- predict(fit.SVM.poly, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        235        101
      Republican         88        185

 Accuracy :  0.6896552
Classification error :  0.3103448

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6994048 0.7275542  0.7132018
2 Republican 0.6776557 0.6468531  0.6618962

 Cohen's Kappa:  0.3753846
```

```
# in testing set
preds <- predict(fit.SVM.poly, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         85         39
      Republican         34         42

 Accuracy :  0.635
Classification error :  0.365

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6854839 0.7142857  0.6995885
2 Republican 0.5526316 0.5185185  0.5350318

 Cohen's Kappa:  0.2351215
```

Next, we use SVM with Gaussian (or radial basis) kernel.

```
fit.SVM.radial <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), ker
  nel = "radial", cost = 4)

# in training set
preds <- predict(fit.SVM.radial, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        235         78
      Republican         88        208

 Accuracy :  0.727422
Classification error :  0.272578

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.7507987 0.7275542  0.7389937
2 Republican 0.7027027 0.7272727  0.7147766

 Cohen's Kappa:  0.4539179
```

```
# in testing set
preds <- predict(fit.SVM.radial, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         84         24
      Republican         35         57

 Accuracy :  0.705
Classification error :  0.295

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.7777778 0.7058824  0.7400881
2 Republican 0.6195652 0.7037037  0.6589595

 Cohen's Kappa:  0.4008936
```

Finally, using Sigmoid kernel.

```
fit.SVM.sigmoid <- svm(x = as.matrix(new_traindoc), y = factor(training_labels), ke
  rnel = "sigmoid", cost = 2)

# in training set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(new_traindoc))
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        161        179
      Republican        162        107

 Accuracy :  0.4400657
Classification error :  0.5599343

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.4735294 0.4984520  0.4856712
2 Republican 0.3977695 0.3741259  0.3855856

 Cohen's Kappa:  -0.1278574
```

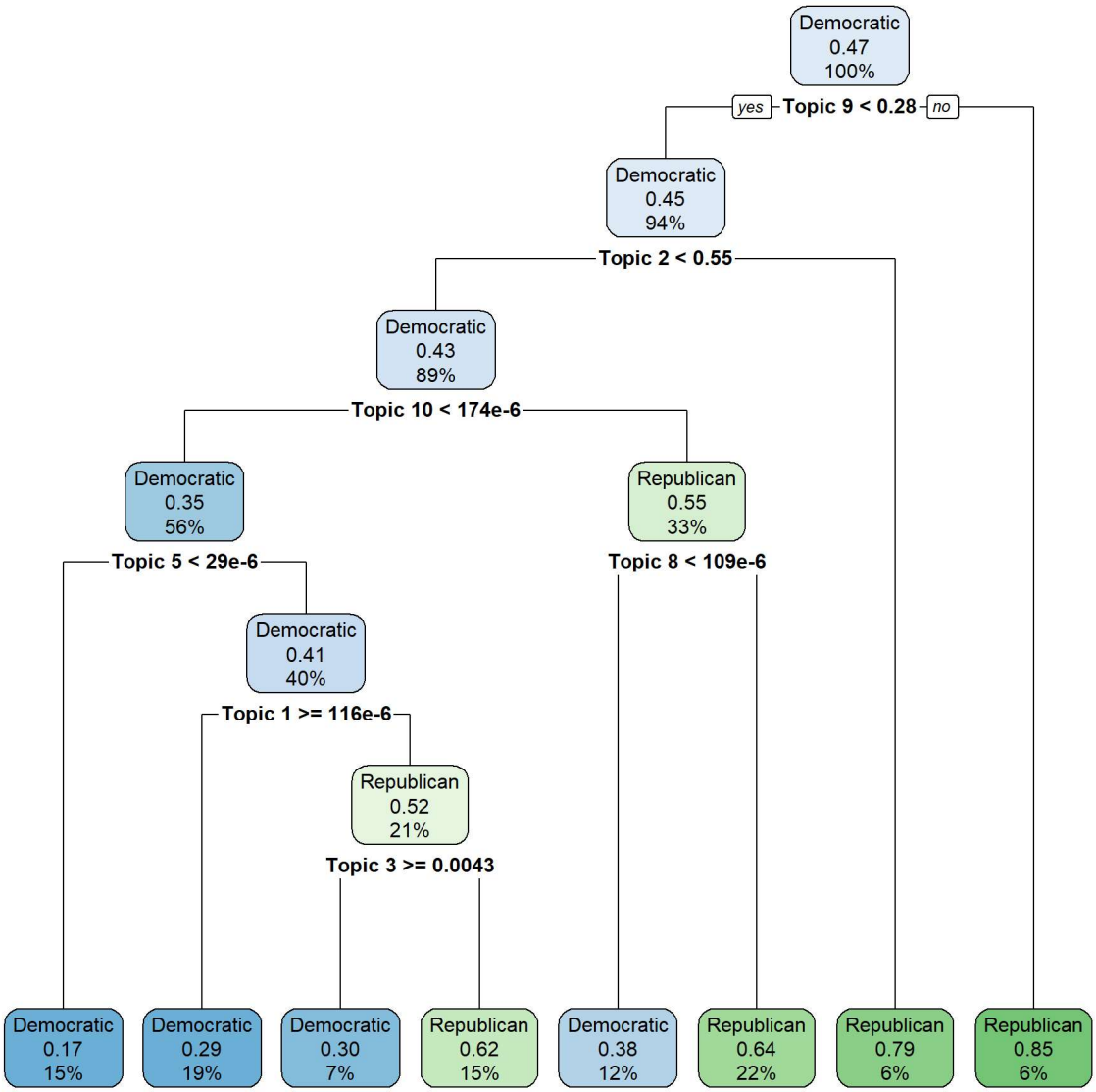
```
# in testing set
preds <- predict(fit.SVM.sigmoid, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         52         34
      Republican         67         47

 Accuracy :  0.495
Classification error :  0.505

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6046512 0.4369748  0.5073171
2 Republican 0.4122807 0.5802469  0.4820513

 Cohen's Kappa:  0.01616988
```

4.5.4 Performance of CART

Now, we use Classification And Regression Tree (CART). The fitted decision tree looks as follows.

```
fit.CART <- rpart(LABEL ~ . , data = cbind(as.data.frame(as.matrix(new_traindoc)),
  LABEL = factor(training_labels)),
                  method = "class", minsplit = 100)
rpart.plot(fit.CART)
```




```
# in training set
preds <- predict(fit.CART, type = "class")
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        230         89
      Republican         93        197

 Accuracy :  0.7011494
Classification error :  0.2988506

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.7210031 0.7120743  0.7165109
2 Republican 0.6793103 0.6888112  0.6840278

 Cohen's Kappa:  0.4005646
```

```
# in testing set
preds <- predict(fit.CART, newdata = as.data.frame(as.matrix(new_testdoc)), type =
  "class")
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         37          9
      Republican         82         72

 Accuracy :  0.545
Classification error :  0.455

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.8043478 0.3109244  0.4484848
2 Republican 0.4675325 0.8888889  0.6127660

 Cohen's Kappa:  0.174678
```

4.5.5 Performance of Random Forest

Finally, we try training Random Forest classifier. Although, since CART did not perform well, it seems that Random Forest will fail too.

```
fit.rf <- randomForest(x = as.matrix(new_traindoc), y = factor(training_labels))

# accuracy in training set
preds <- predict(fit.rf)
classification_report(training_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic        235        105
      Republican         88        181

 Accuracy :  0.683087
Classification error :  0.316913

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.6911765 0.7275542  0.7088989
2 Republican 0.6728625 0.6328671  0.6522523

 Cohen's Kappa:  0.3616526
```

```
# in testing set
preds <- predict(fit.rf, newdata = as.matrix(new_testdoc))
classification_report(testing_labels, preds)
```

```
Classification Matrix is (Row = Predicted, Column = Actual):

                True_labels
Predicted_labels Democratic Republican
      Democratic         60         23
      Republican         59         58

 Accuracy :  0.59
Classification error :  0.41

Classification Report :
       Class Precision    Recall F1.measure
1 Democratic 0.7228916 0.5042017  0.5940594
2 Republican 0.4957265 0.7160494  0.5858586

 Cohen's Kappa:  0.2056573
```

5 Conclusion

From the above empirical analysis, we can derive the following conclusions.

1. The best performing classification rule is obtained using a Support Vector Machine with radial or gaussian basis kernel, in which case, the data has been preprocessed by Latent Dirichlet Allocation.
2. In general, LSA and LDA dimensional reduction techniques make it easier for classification algorithms to run more smoothly and create better results.
3. Based on the latent topics obtained by LSA and LDA, the topics seemed to related towards the following:
    a. Addressing peace and unity, government treaties and foreign relations.
    b. Addressing issues like government, public relations, service, law and various reports.
    c. Adressing bank, money, currency, treasury and economic situations.
    d. Addressing issues like Soviet and Vietnam war.
4. Since the best performing classification rule is only 70% accurate, which is not very satisfactory, it seems that it could really be difficult to classify the US presidents based on their speeches, to assign their philosophies into a republican or democratic ones. The topics in which Republican and Democratics generally disagree, they are obtained using the topic modelling to some great extent.