I will take your words for it

Predicting US-presidents by their speeches

Benjamin Arthur Maier

16/11/2020

Introduction

In the movielens project we constructed a recommender-system based on a large, but simplistic data set of movies, users and ratings. Real-world problems are often more complex and involve data sets with a greater amount of features or sets where the data isn't neatly ordered. For my capstone project I wanted to try something different from what we have done in the course, so far. My background is microbiology, and I'm writing my thesis on, among other things, large transcriptomics data sets. It would have been easy to use one of the myriad of published sets and simply apply the code I wrote for my stuff (thanks to this course, btw!), but nah. I wanted something different. Since I can't open a web-browser without getting swamped by US-politics (mostly because, outside of a a crippling pandemic, nothing ever happens where I am from), I thought I want a project related to the presidency. I selected a data set created by the GrammarLab containing transcripts of speeches made by US-presidents from George Washington to Barack Obama[1]. Using the text analysis package `Quanteda`[2] we will try to predict each president by the contents of his speeches. The accuracy of this model will set the baseline. Our aim will be to build our own model that has a higher accuracy than `Quanteda`'s. To this end, we will explore the data to get a feeling of its structure and contents and discuss data-cleaning steps that might help us to improve our model and reduce computing time.

Models

Support Vector Machine. The idea behind support vector machines (SVM) is that for truly different classes, it is possible to draw a hyperplane between the features that neatly separates them into their respective classes. As an example, we take two Presidents removed from each other by quite some time. George Washington would talk about the constitution, the British, militias or native Americans (though not using that term) while George W. Bush would talk about Irak, Afghanistan, the war on terror and so on. Both of these words have frequencies which can be made into data points e.g. on a word - frequency graph and a line can be drawn between them, neatly seperating Bush and Washington. The SVM will thereby try to maximize the distance between the hyperplane and the two features, belonging to different classes, closest to the hyperplane. These two point are what is referred to as "support vectors" and their distance to the hyperplane will determine the confidence in the classification. `Quanteda` contains two ML implementations for supervised learning. Naive Bayes (NB) and SVM. We will use the SVM model as it outperforms the NB model on our data. In our own analysis we will use the package `LiblineaR`[3] with caret. The SVM method contained in this library combines regularization with SVM for improved performance.

Logistic Regression. The principles behind logistic regression (logReg) are similar to the ones behind SVM[4]. It also assumes that a linear line or hyperplane can be used to separate truly different classes. In contrast to SVM that tries to only maximise the distance between the two points closest to the hyperplane (the support vectors), logReg ascribes a probability to each outcome and tries to maximise that. Therefore, while both "draw" hyperplanes, SVM relies on the geometrical properties of the data in an n-dimensional space while logReg is based on a statistical approach. To build our model, we will used a regularized multinomal logReg (multinomal = more than two categories to predict) contained in the `LiblineaR` library.

Data Loading and Baseline-Setting

Data loading and basic filtering

In the first section, we load the library and perform some basic filtering to get the data ready for `Quanteda`. In the set provided by the GrammarLab, all speeches are in separate sub-folders. For simplicity reasons, I copied them all into one folder. We begin by loading the required libraries:

```r
library(quanteda)
library(quanteda.textmodels)
library(tm)
library(ggplot2)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(caret)
library(ggfortify)
library(dplyr)
library(stringr)
library(readr)
library(readxl)
library(matrixStats)
library(tinytex)
library(gplots)
library(LiblineaR)
library(pamr)
library(ggpubr)
```

Next we load the library and extract the names of the respective president from the file names.

```r
# Create a list containing all speeches
collection <- list.files("capstone_project")
speeches <- list()
for(file in collection){
  speeches[[file]] <- read_file(paste("capstone_project/",file,sep=""))
}

#Extract the speeches
speechtable <- c()
speechtable <- sapply(1:length(speeches), function(x)
{
  append(speechtable,speeches[[x]])
})

# Extract the presidents' names from the file names
presidents <- sapply(1:length(collection), function(x){
  gsub("_.*", "", collection[x])
})
```

Now we combine the speaker with the speech and remove debates, as they contain transcripts not belonging to a president:

```r
# Create a data frame containing presidents and speeches
data <- data.frame(presidents = presidents, text = speechtable)
data <- data %>% filter(!str_detect(data$text,
                                    pattern = "title=\\\"Debate with"))

data <- data %>% mutate(text = gsub(".*\">", "", text))

# Dimensions of the data set
dim(data)
```

```
## [1] 950   2
```

```r
# Number of presidents contained in the data set
length(unique(data$presidents))
```

```
## [1] 43
```

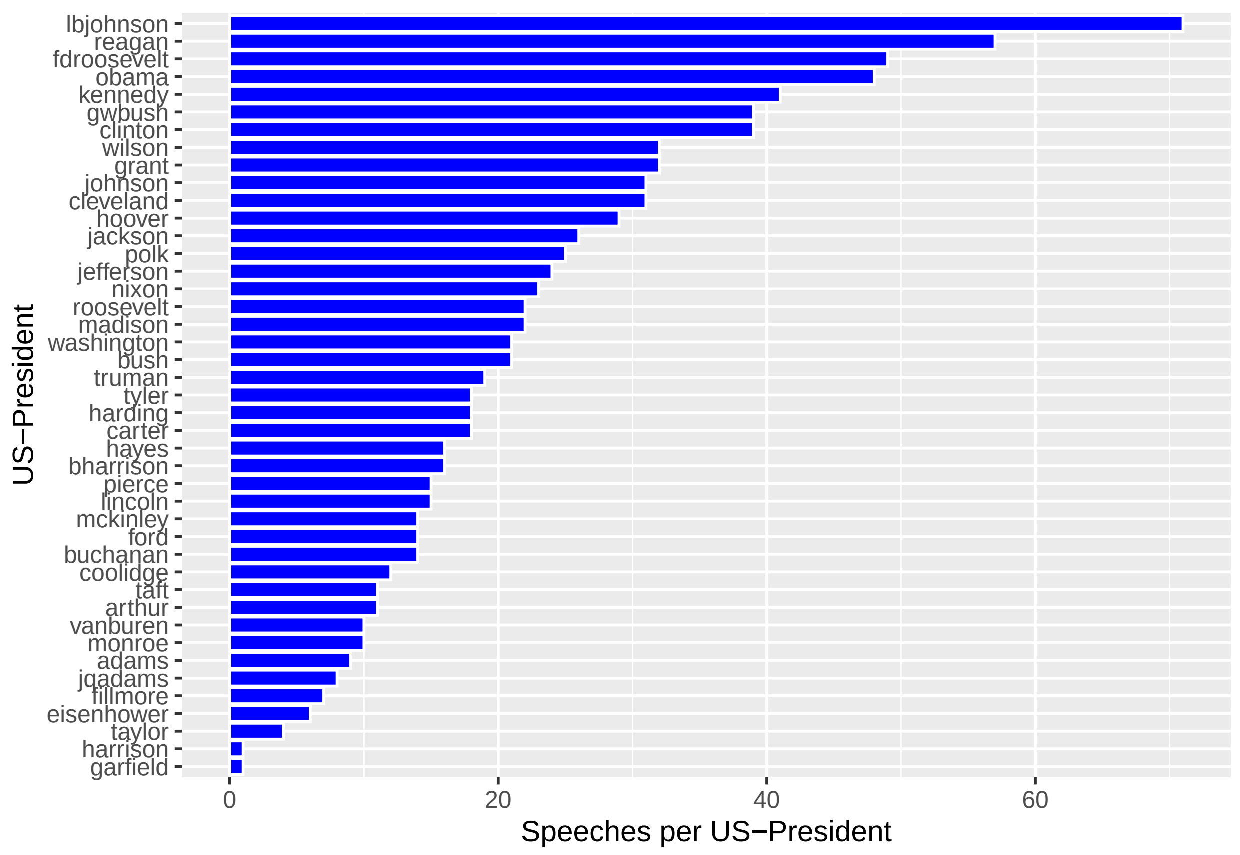
We see our data now contains 950 speeches from 43 presidents. It is not entirely complete but it will suffice. Of course, all the speeches are just very long strings for now and not very useful. But it is sufficient for `Quanteda` to work with. First, let's make a copy of the data for exploration purposes:

```r
exp_data <- data
```

Now let's have a look at how many speeches we have for each president.

```r
# Add a word-count metric to the data for later data exploration
exp_data <- exp_data %>% mutate(word_count = sapply(strsplit(exp_data$text, " "),
                                                    length))

# Create a president-summary containing the amount of speeches,
# the amount of words said and the amount of words per speech
prsum <- exp_data %>%
                      group_by(presidents) %>%
                      summarize(speeches = n(),
                                words = sum(word_count),
                                s_length = sum(word_count)/n())
```

```r
prsum %>% ggplot(aes(x=reorder(presidents, speeches),y=speeches)) +
                      geom_bar(stat = "identity", color = "white", fill = "blue") +
                      coord_flip() +
                      labs(y = "Speeches per US-President",
                           x = "US-President")
```



```r
max(prsum$speeches)
```

```
## [1] 71
```

```r
min(prsum$speeches)
```

```
## [1] 1
```

We have the most speeches for Lyndon B. Johnson and the fewest for James Garfield and William Harrison. As we will be splitting the data set into train and test data later, we will remove presidents with fewer than 10 speeches in our data. I'm sorry Mr. Garfield, Mr. Harrison, Mr. Taylor, Mr. Eisenhower, Mr. Fillmore and both Mr. Adams. But you seven are a sacrifice that I'm willing to make. That leaves us with 36 US-presidents.

```r
#Create filter and remove presidents with fewer than 10 speeches
filter <- prsum %>% mutate(filt = speeches >= 10)
filter <- filter %>% select(presidents, filt)
exp_data <- left_join(exp_data,filter, by = "presidents")
exp_data <- exp_data %>% filter(filt == TRUE)
exp_data <- exp_data %>% select(presidents, text)
exp_data$presidents <- factor(exp_data$presidents)
```

With this first data-cleaning step, we are ready to set our baseline. As discussed, we will make use of the textmining package `quanteda`. Quanteda is short for *qu*antitative *an*alysis of *te*xtual *da*ta and is a very powerful natural language processing tool. Of interest for us is the accompanying library `quanteda.textmodels` that features two in-built classification methods: Naive Bayes and Support Vector Machines. For our base-line model we will draw heavily from this (tutorial)[5] made by Cosima Meyer from the University of Mannheim, Germany.

Baseline setting using the `quanteda` library

We begin by creating a corpus as input for `quanteda`. We give every speech an id by which we can then partition the data.

```r
#Create corpus
speechcorpus <- corpus(exp_data, text_field = "text")
docvars(speechcorpus,"id") <- 1:ndoc(speechcorpus)
```

Now we split the corpus into a training and a test corpus in a $70 : 30$ ratio.

```r
set.seed(1337, sample.kind = "Rounding")
index <- createDataPartition(exp_data$presidents, times = 1, p = 0.7)

train_corpus <-
  corpus_subset(speechcorpus, !id %in% index$Resample1) %>%
  dfm(stem = TRUE)

test_corpus <-
  corpus_subset(speechcorpus, id %in% index$Resample1) %>%
  dfm(stem = TRUE)

# Match words from test and train data so they contain the same features.
test_corpus <-   dfm_match(test_corpus, features = featnames(train_corpus))
```

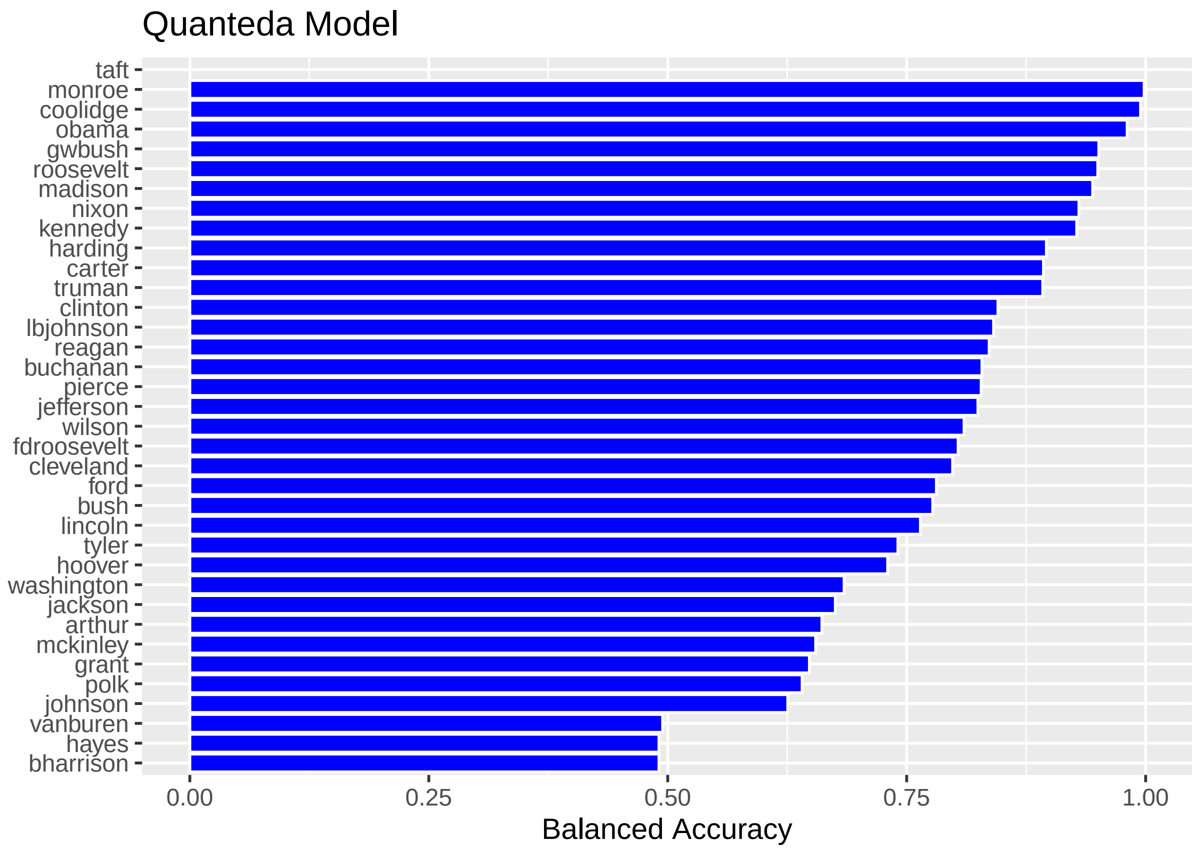
With those simple steps we can already fit the model and see how it fits the training data.

```r
# SVM model
speechmodel <- textmodel_svm(train_corpus, docvars(train_corpus, "presidents"))

# Let's see how well the model fits the training data
selffit1 <- predict(speechmodel, train_corpus)
confusionMatrix(table(docvars(train_corpus, "presidents"), selffit1))$overall["Accuracy"]
```

```
## Accuracy
##        1
```

That's a perfect fit. But of course, human language is a complicated thing and a perfect fit on the training data could just be indicative of overfitting. So next we fit the test test and evaluate total accuracy and the balanced accuracy for each president.

```r
# Use the model to predict the presidents of the training data
fit <- predict(speechmodel, test_corpus)

# Plot balanced accuracy for each President
result_svm <- confusionMatrix(table(docvars(test_corpus, "presidents"), fit))
bal_acc_svm <- data.frame(president = names(result_svm$byClass[,"Balanced Accuracy"]),
                           "balanced accuracy" = result_svm$byClass[,"Balanced Accuracy"])
bal_acc_svm <- bal_acc_svm %>% mutate(president = sapply(1:nrow(bal_acc_svm), function(x) {
  (gsub(".*: ", "", bal_acc_svm$president[x]))
}))
bal_acc_svm %>% ggplot(aes(y = reorder(president, balanced.accuracy),
                            x = balanced.accuracy)) + geom_col()+
  geom_col(color = "white", fill = "blue") +
  labs(x = "Balanced Accuracy",
       y = "",
       title = "Quanteda Model")
```



```
# Overall model accuracy
result_svm$overall["Accuracy"]
```

```
##  Accuracy
## 0.6021341
```

```
# Mean balanced accuracy over all presidents
mean(bal_acc_svm$balanced.accuracy, na.rm = T)
```

```
## [1] 0.7895639
```

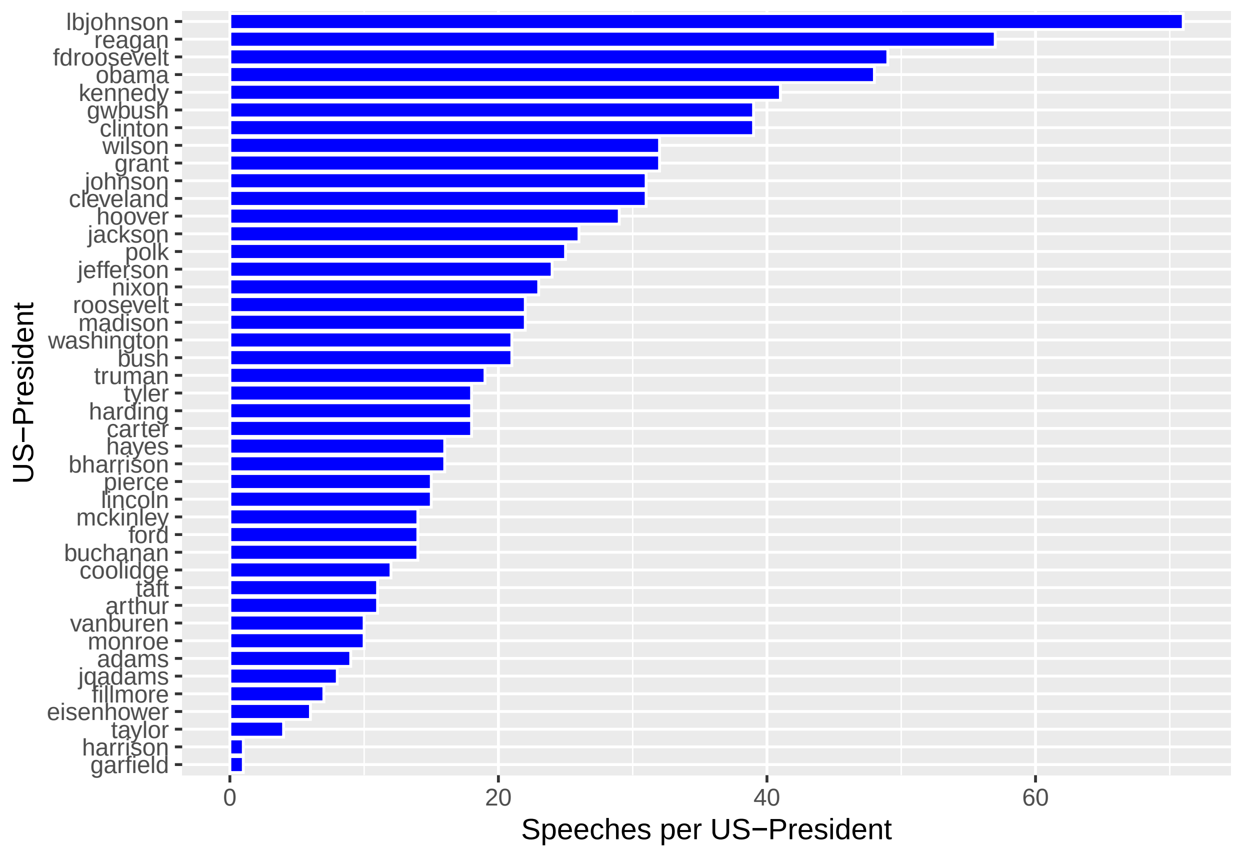
```r
# Save the accuracy in a data_frame
model_performance <- data.frame(quanteda = result_svm$overall["Accuracy"])
```

That is the score we aim to beat. We will try to beat an accuracy of 0.60 (and a mean balanced accuracy of 0.79). So let's get started.

Data Exploration

Now that we have set our baseline, we need to think about or own model. Let's return first to the presidential summary and investigate again the number of speeches, the number of words spoken and the average number of words per speech.

```r
# How many speeches did each president give?
prsum %>% ggplot(aes(x=reorder(presidents, speeches),y=speeches)) +
  geom_bar(stat = "identity", color = "white", fill = "blue") +
  coord_flip() +
  labs(y = "Speeches per US-President",
       x = "US-President")
```

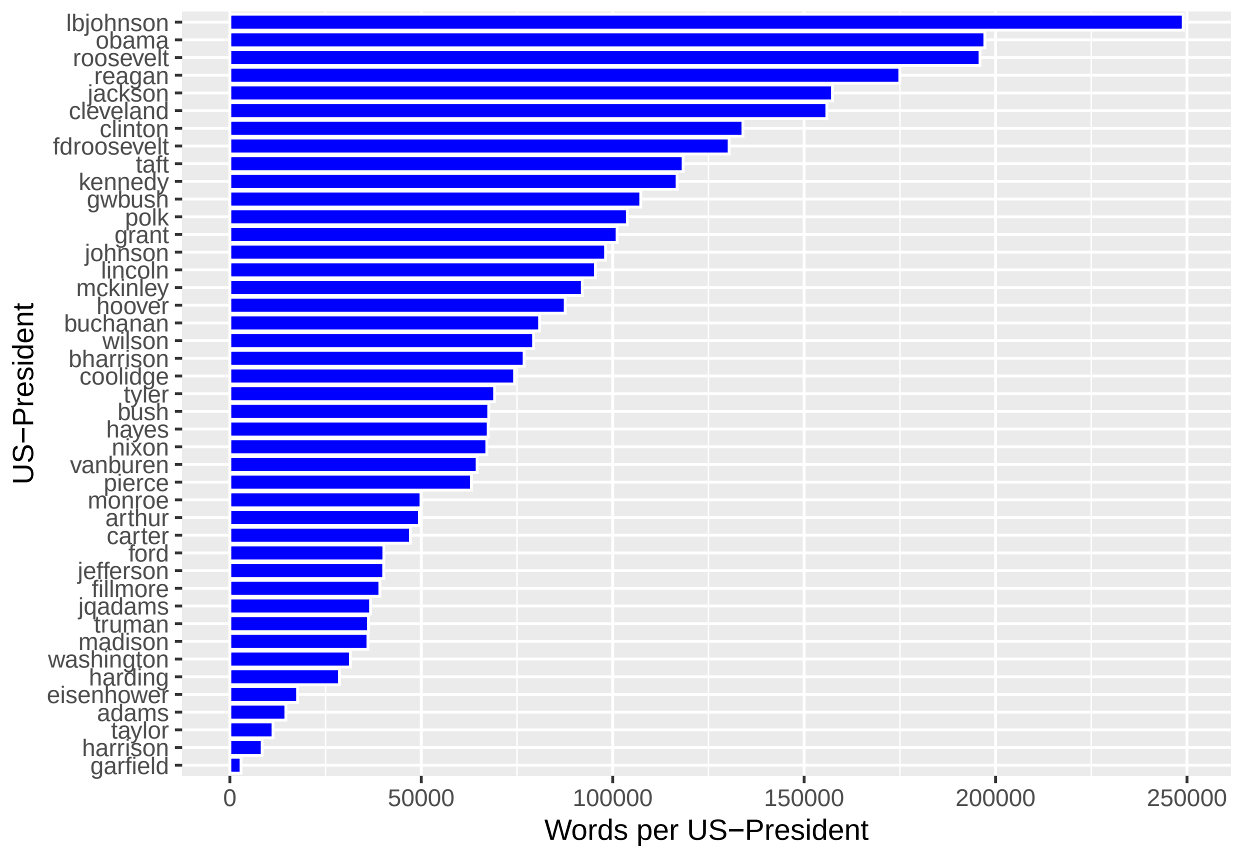


Our data set is highly imbalanced. Even though we threw out the bottom five, that still leaves us with six times the number of speeches for LBJ than we have for Martin Vanburen. For the predictions, a smaller amount of speeches could be both advantageous and disadvantageous. Advantageous because with fewer speeches, these might be centered around a more uniform topic and have much in common, while within 60 speeches, these might range from financial to social politics, addresses to congress about specific laws, etc.

But of course, the fewer speeches there are, the fewer data is there for the model to be trained on. We can't do much about it, but we have to keep it in mind.

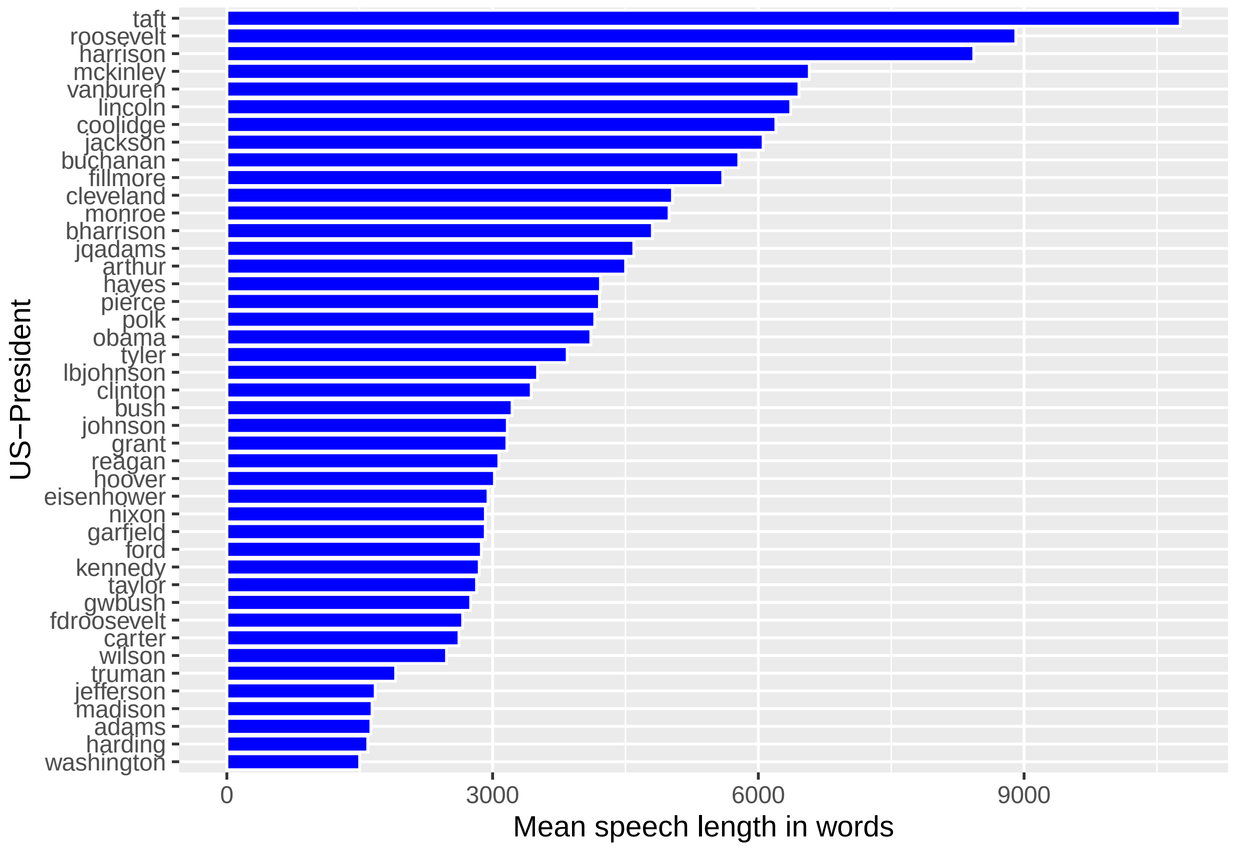
As words will be our features, let's see how the data set looks in that regard.

```r
prsum %>% ggplot(aes(x=reorder(presidents, words),y=words)) +
                      geom_bar(stat = "identity", color = "white", fill = "blue") +
                      coord_flip() +
                      labs(y = "Words per US-President",
                           x = "US-President")
```



Here, too, LBJ is in the lead, followed by Barack Obama and Theodore Roosevelt. Harrison and Garfield come last again. As for the presidents who were most in love with their own voice. . .

```r
prsum %>% ggplot(aes(x=reorder(presidents, s_length),y= s_length)) +
                      geom_bar(stat = "identity", color = "white", fill = "blue") +
                      coord_flip() +
                      labs(y = "Mean speech length in words",
                           x = "US-President")
```



```
max(prsum$s_length)
```

```
## [1] 10762.64
```

```
min(prsum$s_length)
```

```
## [1] 1498.952
```

This list is led by William Howard Taft and Theodore Roosevelt, with the former being the latter ones prodigy. Sadly, while Taft inherited Roosevelt's tendency to talk too much, he did not inherit his charisma and oratory skills, which is why his quotes feature much less on motivational posters, Cadillac ads [6] and your dads facebook posts. Interestingly, William Harrison features on a prominent 3rd place. Too bad we only have this one speech of him.

We now have an overview over the bulk data. We have seen, that it is not a very balanced data set. Some presidents have more speeches with more words from which to train an algorithm from. We already had to get rid of seven, as they had fewer than 10 speeches in our library. To do a more in-depth analysis and look at the contents of the speeches, we will use the `tm` library, a very basic text mining tool for R [7]. We will also use `tm` to build a term matrix, with every word being being a row with an assigned numerical value based on its presence in each speech. This term matrix will serve as the basis for our model building. To do this, we first need to built a corpus that extracts words and makes them into features, or tokens. This task will also perform some fundamental data cleaning such as removal of punctuation, numbers, white spaces or stop words. Stop words in that context refers to frequent words without much meaning such as "the", "as", "for", "in" etc. We will also remove the words "can", "will" and the "applause" that is transcribed in modern speech-transcripts and "else", "for" and "if" because R tends to be confused by them.

```r
# Create a corpus of words from the strings of speeches
corp_data <- Corpus(VectorSource(exp_data$text))

# Convert the text to lower case
corp_data <- tm_map(corp_data, content_transformer(tolower))

# Remove stopwords (will take a while)
corp_data <- tm_map(corp_data, removeWords, stopwords("english"))

# Remove punctuations
corp_data <- tm_map(corp_data, removePunctuation)

# Remove numbers
corp_data <- tm_map(corp_data, removeNumbers)

# Remove white spaces
corp_data <- tm_map(corp_data, stripWhitespace)

# Remove other common words not in stopwords and "applause" from transcripts
corp_data <- tm_map(corp_data, removeWords, c("the", "will","applause",
                                              "laughter" ,"can", "if",
                                              "else", "for"))
```

From this corpus, `tm` allows us to easily construct the term matrix:

```r
# Build a term matrix of all words said by all presidents with their frequencies
term_matrix <- TermDocumentMatrix(corp_data)
term_matrix <- as.matrix(term_matrix)
colnames(term_matrix) <- exp_data$presidents
dim(term_matrix)
```

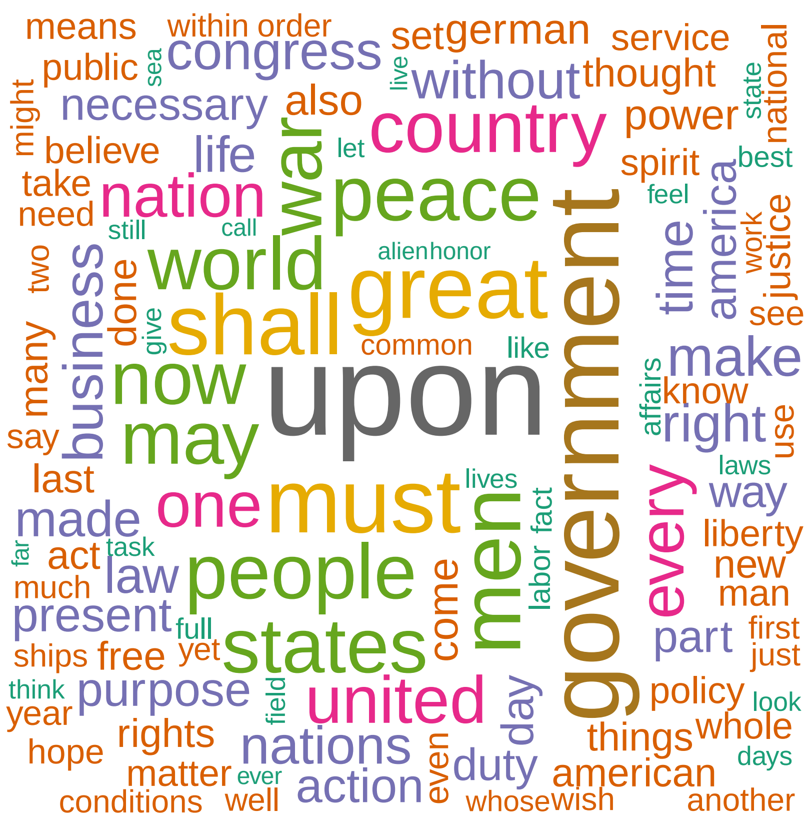
```
## [1] 34735   914
```

Our `term_matrix`now contains 34 735 words from 914 speeches from 36 presidents. That makes a total of 31,747,790 observations. That should give us plenty to work with. It is about 4 times the size of the data set I'm writing my thesis about. As our algorithm will decide the speaker by the words they used, let's have look at the most favorite terms of some presidents. We do this by using the `wordcloud` library. Wordclouds are visual representations of word-frequencies displayed as... well, word clouds. We won't do this for all remaining 36 presidents, but instead pick out a few representatives whose terms in office were defined by well-know occurrences. If I misrepresent one of them, cut me some slack, I'm not American. I would have taken our own leaders but they can't settle on a common tongue and the most prominent language is in a dialect considered to be something of an insult to our same-language neighbors. So US-presidents it is.

```r
# Woodrow Wilson - Scholar president during WW1
wilson <- term_matrix[,colnames(term_matrix) == "wilson"]

# Civil war president. Ended slavery.
lincoln <- term_matrix[,colnames(term_matrix) == "lincoln"]

# Good ol Teddy. Slightly crazy. Got shot and shrugged it of. Also Panama canal
roosevelt <- term_matrix[,colnames(term_matrix) == "roosevelt"]
```

Let's start with No. 28, Woodrow Wilson. He was a political science and history scholar and the only US-president to hold a PhD. Wilson led the US during the First World War. He is most famous for his 14 points and for fathering and trying to get the Senate to accept his brain-child, the *League of Nations*.

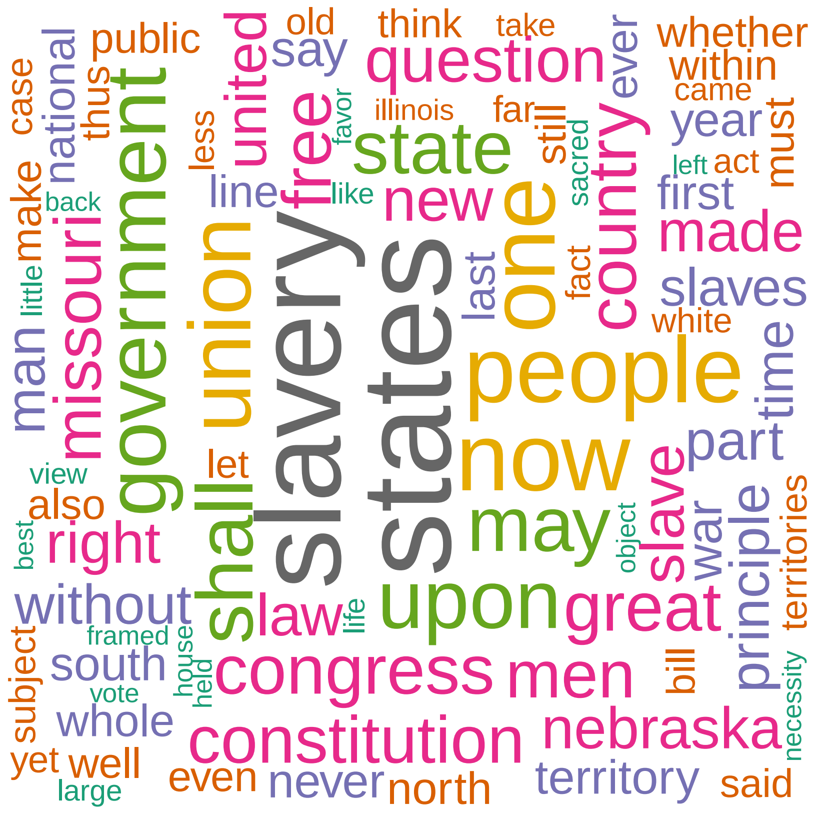
```
# Cloud for Woodrow Wilson
ww <- sort(rowSums(wilson),decreasing=TRUE)
wilsonc <- data.frame(word = names(ww),freq=ww)

wordcloud(words = wilsonc$word, freq = wilsonc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



There is not much to see. *Upon*, *government*, *must* and *great* are almost stopwords. The only interesting terms are *peace*, *world* and *war* as well as a rather small *german*.

Next is Abraham Lincoln. No further introduction needed. Probably the single most famous president there is after George Washington.

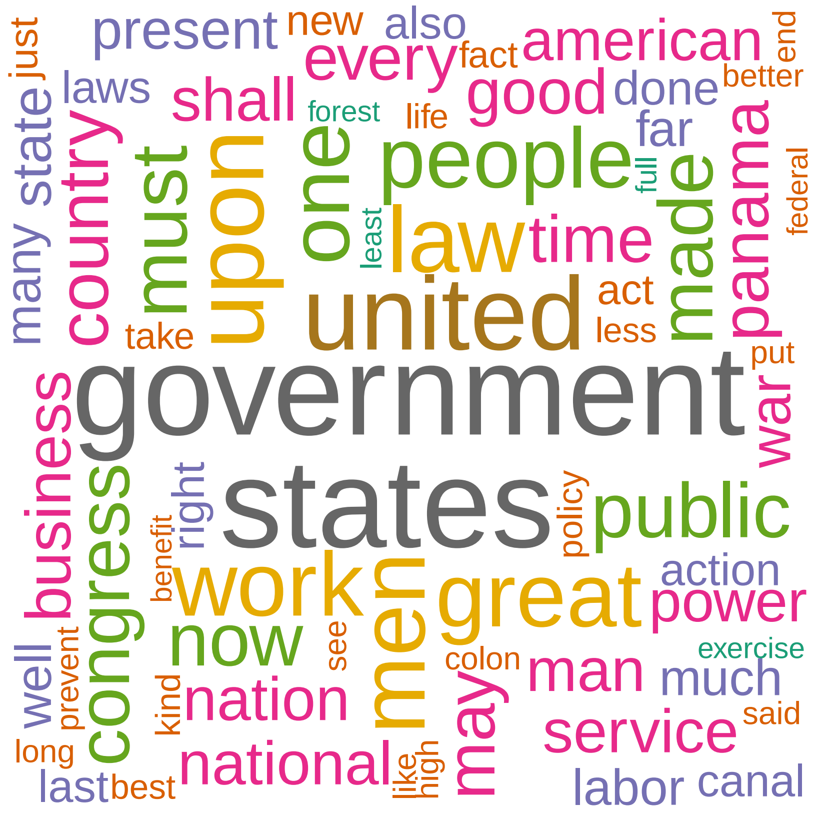
```
al <- sort(rowSums(lincoln),decreasing=TRUE)
lincolnc <- data.frame(word = names(al),freq=al)

wordcloud(words = lincolnc$word, freq = lincolnc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



Here, we find more terms that could be used to identify Abe. Especially *slavery*, *free*, *slaves* and *union* feature prominently. But again, we find *government* and *great* which we have see before.

Now let's move on to the Bull Moose. The elder Roosevelt was known for his populist style, trust busting, national parks, the *Great White Fleet* and the *Panama Canal*.

```
# Cloud for Teddy Roosevelt
tr <- sort(rowSums(roosevelt),decreasing=TRUE)
teddyc <- data.frame(word = names(tr),freq=tr)

wordcloud(words = teddyc$word, freq = teddyc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2") )
```



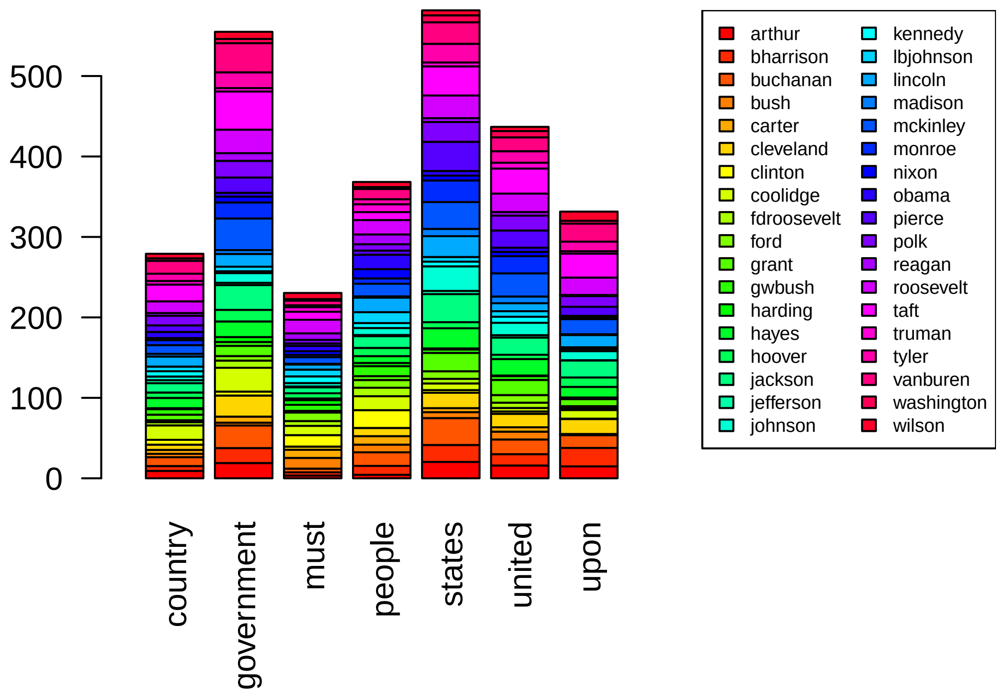
We see *Panama* there, but most prominently featured are *states*, *government* and *united*, and our usual *must* and *great*.

It is clear from these three clouds, that our data still contains a lot useless words. It was to be expected. Any US president is likely to use the terms *United States*, *government*, *people* and strong sentences including words such as *great* or *must*. It is legitimate to remove them as they are common to most if not all presidents. We can show this by summarizing the words from all speeches under the respective president

```
# favorite words per president
presnames <- unique(exp_data$presidents)
pres_pref <- sapply(1:length(presnames), function(x) {
                    namef <- colnames(term_matrix) == presnames[x]
                    rowSums(term_matrix[,namef])/sum(namef)
})
colnames(pres_pref) <- presnames
```

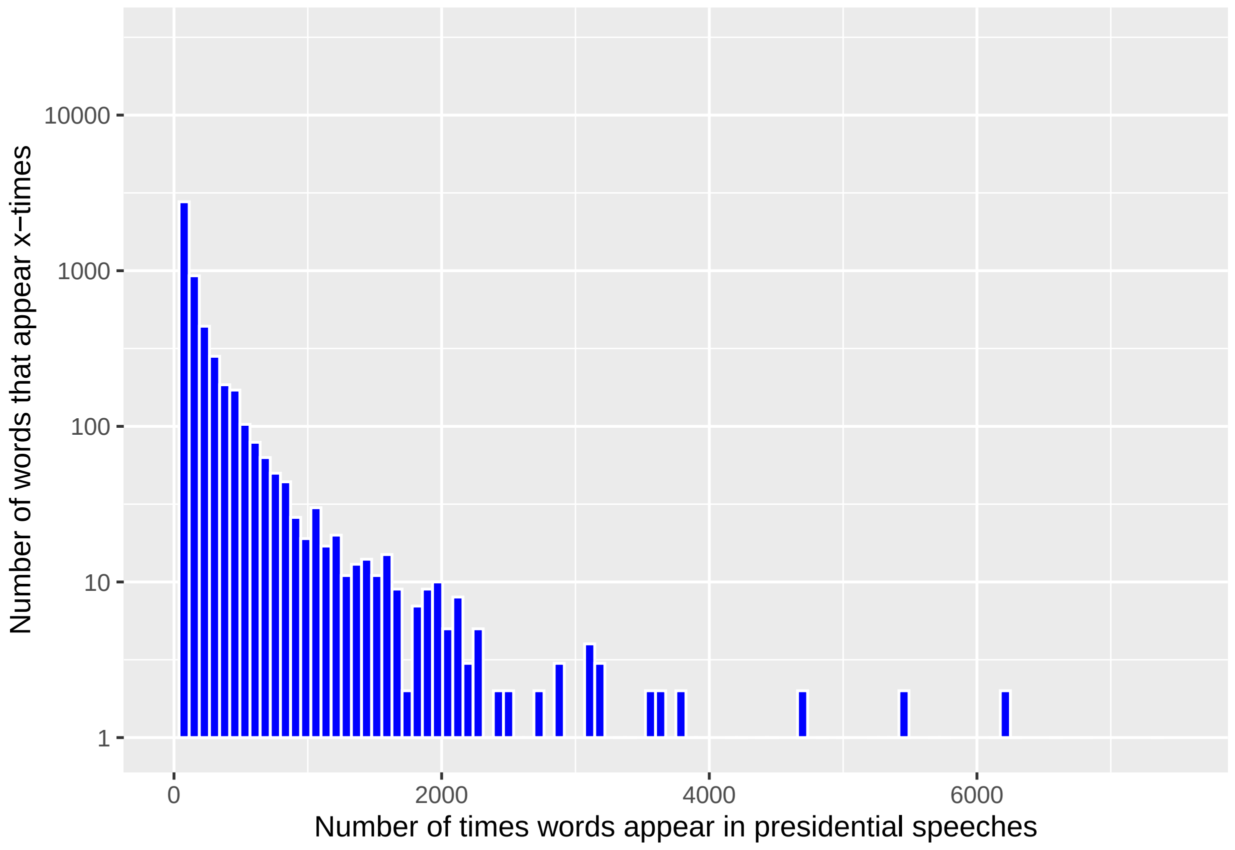
Now we define some “political stopwords” and plot them.

```
# Show the "political stop words"
psw <- c("united","states","upon","government","must","will","people", "country")
swfilter <- rownames(pres_pref) %in% psw
par(mar=c(6,6,4,3)+.1)
t(pres_pref[swfilter,]) %>% barplot(col=rainbow(ncol(pres_pref)),
                                    xlim=c(0, ncol(pres_pref)*0.4),
                                    las = 2)
legend("topright", legend = colnames(pres_pref),
       ncol = 2,
       cex = 0.6,
       fill = rainbow(ncol(pres_pref)),
       col=rainbow(ncol(pres_pref)))
```



You can see that they have all colours of the rainbow, meaning most if not all presidents used them liberally. Now this selection was of course a bit arbitrary. So let's count the frequency of all words and look at their frequency distribution and the ratio of frequent and infrequent words.

```
word_frequency <- data.frame(words = rownames(term_matrix),
                             count = rowSums(term_matrix))

word_frequency %>% ggplot(aes(x=count)) +
                    geom_histogram(bins = 100, fill = "blue", color = "white") +
                    xlim(0,7500) +
                    scale_y_continuous(trans="log10")+
                    labs(x = "Number of times words appear in presidential speeches",
                         y = "Number of words that appear x-times")
```



```r
# What percentage of words is used fewer than 10 times
sum(word_frequency$count < 10) / nrow(term_matrix) * 100
```

```
## [1] 68.75774
```

```r
# What percentage of words is used more than 1000 times
sum(word_frequency$count > 1000) / nrow(term_matrix) * 100
```

```
## [1] 0.725493
```

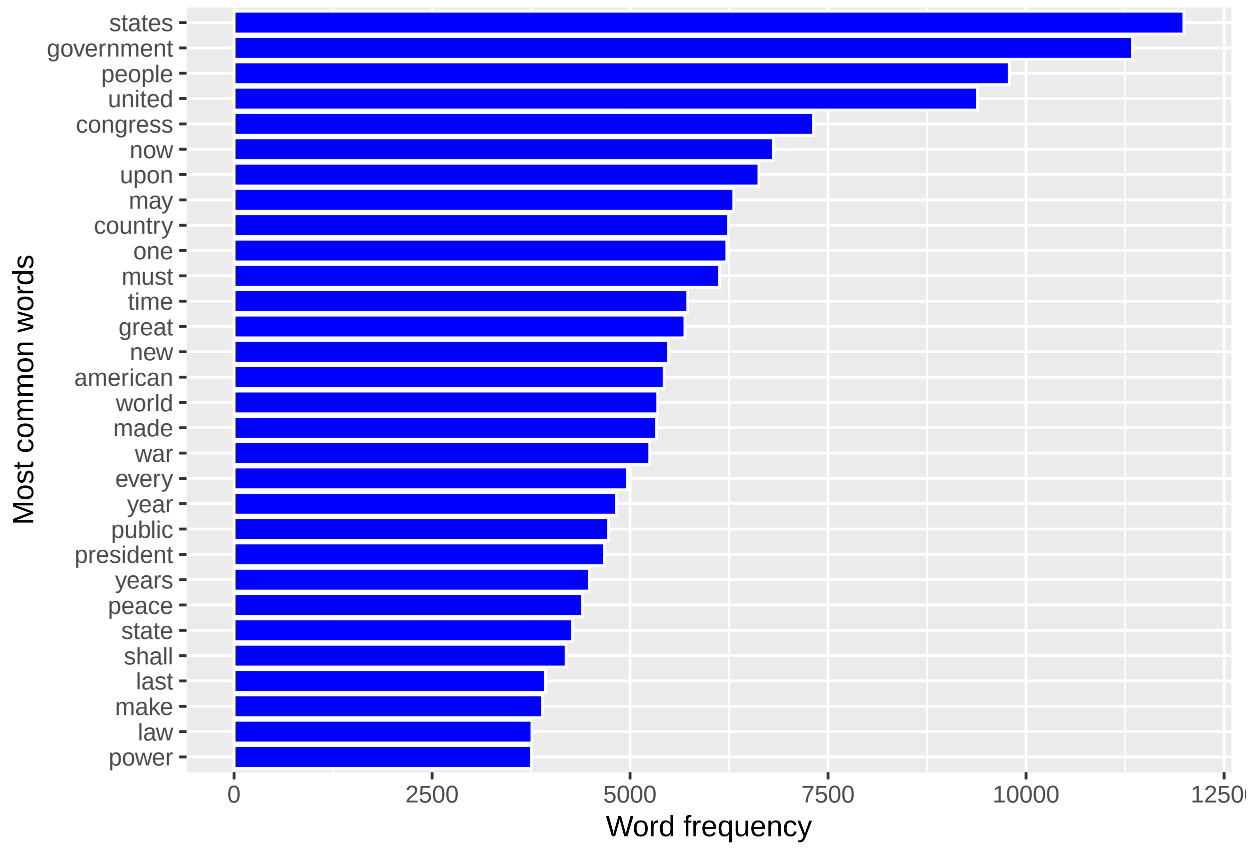
```r
# What percentage of total spoken words are the most frequent ones?
frequent <- word_frequency %>% filter(count > 1000) %>% summarize(count = sum(count))
all <- word_frequency %>% summarize(count = sum(count))
frequent/all
```

```
##       count
## 1 0.3382071
```

Two things are evident from this histogram. 1. There are words that are incredibly frequent and are used more than 1000 times. These words only make up less than 1 % of individual words but more than 30 % of all spoken words 2. The vast majority of individual words, almost 70 %, are used less than 10 times.

We can inspect the top 30 of those high-frequency words.

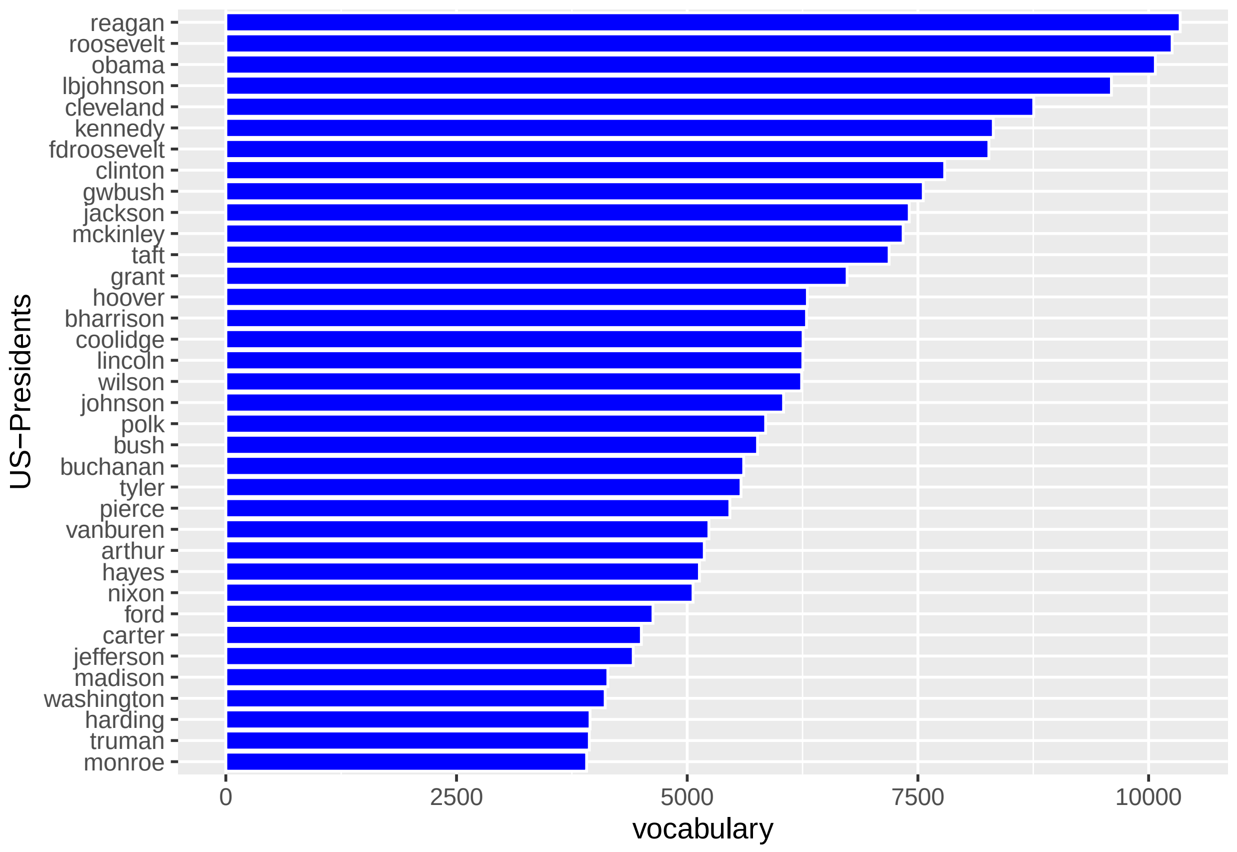
```
word_frequency[order(word_frequency$count, decreasing = TRUE),] %>%
                                top_n(n=30, wt = count) %>%
                                ggplot(aes(y=reorder(words, count), x=count)) +
                                geom_col(fill="blue", color="white") +
                                labs(x = "Word frequency", y = "Most common words")
```



None of them come as a surprise. Except maybe *world* and *war*. To paraphrase principle Skinner: "You Americans sure are a contentious people..."

Above, we have made a table summarizing word-usage by all presidents over all their speeches. By counting all words that appeared at least once, we can get some measure of their vocabulary. So let's see how verbose they were.

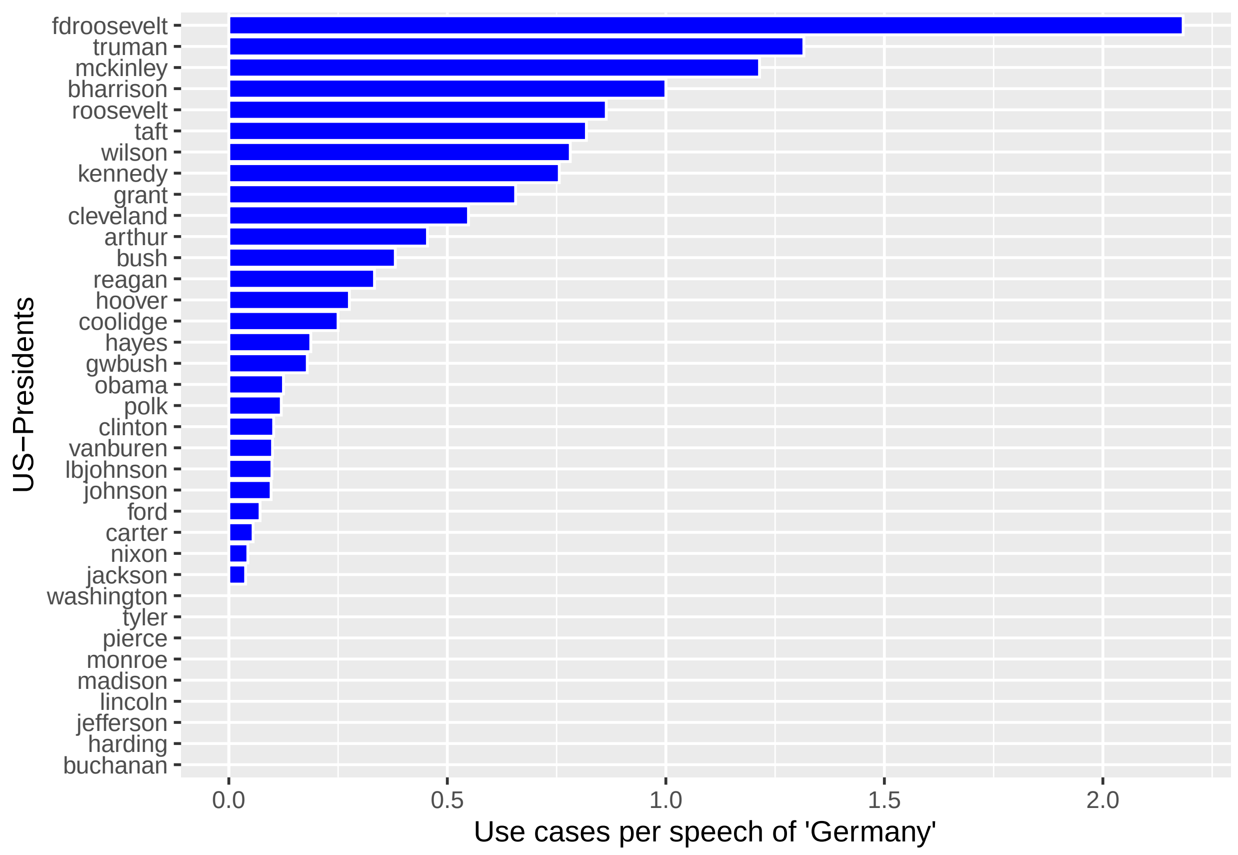
```
verbosity <- data.frame(presidents = colnames(pres_pref),
                        words= colSums(pres_pref > 0))
verbosity %>% ggplot(aes(x=words,
                             y = reorder(presidents, words))) +
                             geom_col(color = "white", fill="blue") +
                             labs(x = "vocabulary",
                                  y = "US-Presidents")
```



The list is lead by Ronald Reagan, Theodore Roosevelt and Barack Obama, all presidents known for their oratory skills. Well, in the case of Ronal Reagan it was more of an up- and down thing, but still... Of course, our analysis here is very basic, as it does count different versions of the same word as a single word. To do this, we would have to *stem* the data. A process that reduces each word to its most basic form. However, for our purposes, this is not necessary.

Of course, political stop words won't help our algorithm deciphering the correct speakers. What we need are "defining words". Words that were, if not unique to, significantly more used by some presidents, compared to others. Identify enough of those and it should be possible to determine the correct speaker. Let's try to find some "defining words" by looking at a few examples. In the last century, the US had two big enemies. The Germans in both world wars, and the Soviet Union. In the centuries before, the US was an isolationist state and its problems were directed inwards. We should be able to group presidents by some key words associated with these issues. Let's start with with the country holding the award for "Most World Wars started". How many times did any president mention the Germans?

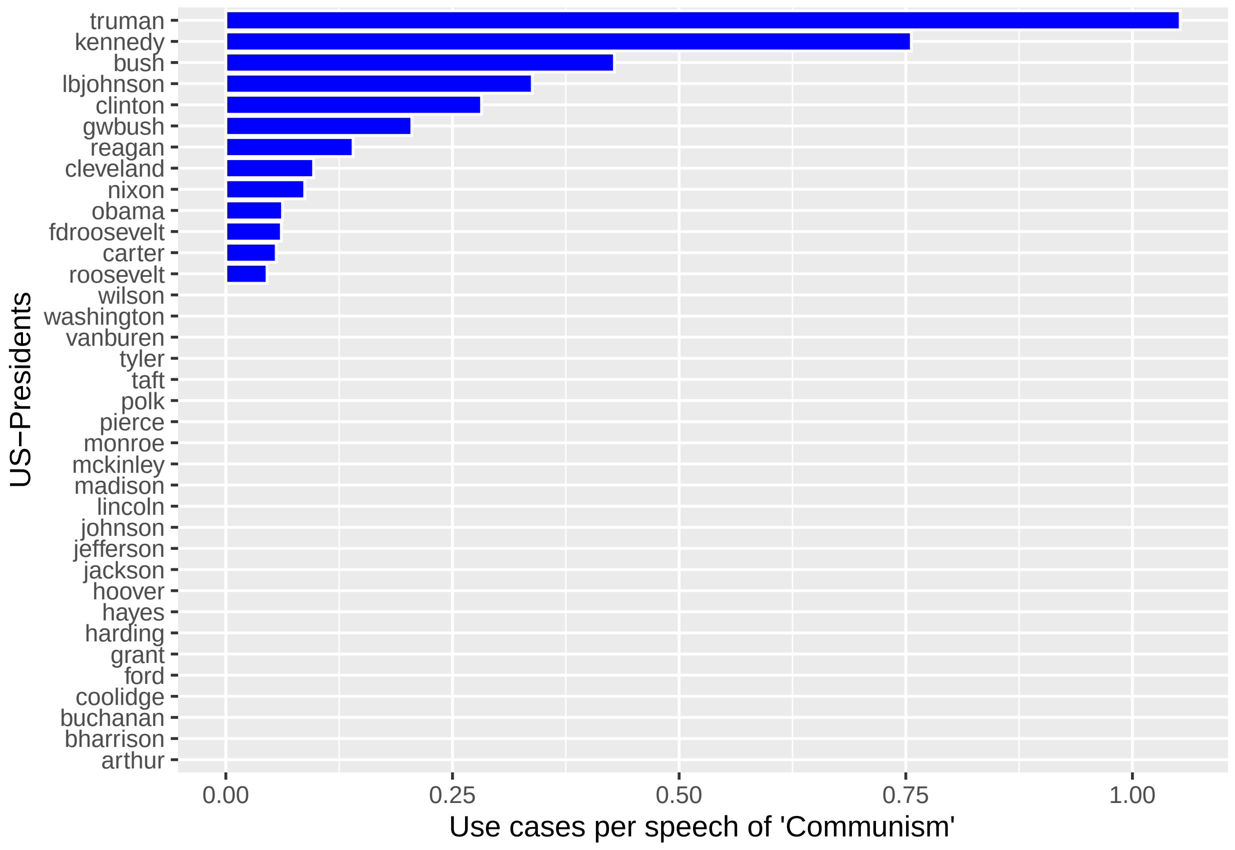
```
germany <- data.frame(presidents = colnames(pres_pref),
                      count = pres_pref["germany",])
germany %>% ggplot(aes(x=count, y= reorder(presidents,count))) +
                      geom_bar(stat="identity", fill = "blue", color = "white") +
                      labs(x= "Use cases per speech of 'Germany'", y = "US-Presidents")
```



Leading are Franklin D. Roosevelt and Harry S. Truman, the two WW2 presidents. Interestingly, William McKinley and Benjamin Harrison were also worried about the Teutons. The reason for McKinley was that he was concerned about German influence in Asia in the wake of the Boxer uprising and German territorial demands. Also, starting from Benjamin Harrison, there was an ongoing struggle over the control of Samoa with the German empire that McKinley finally solved. This struggle also involved the presidency of Grover Cleveland, who is also in the upper third of the list. Leading up to the first World War, the continued blusterous and aggressive nature of the Germans under Kaiser Wilhelm II annoyed many a president, which is while both Theodore Roosevelt and Howard Taft are in leading positions, as well. At least the Germans seemed to have learned their lesson now and stick to making cars, beer and special categories of the gentlemen's-entertainment video market.

Now how about the other big enemy. The Soviet Union. How big was the Red Scare during any period?

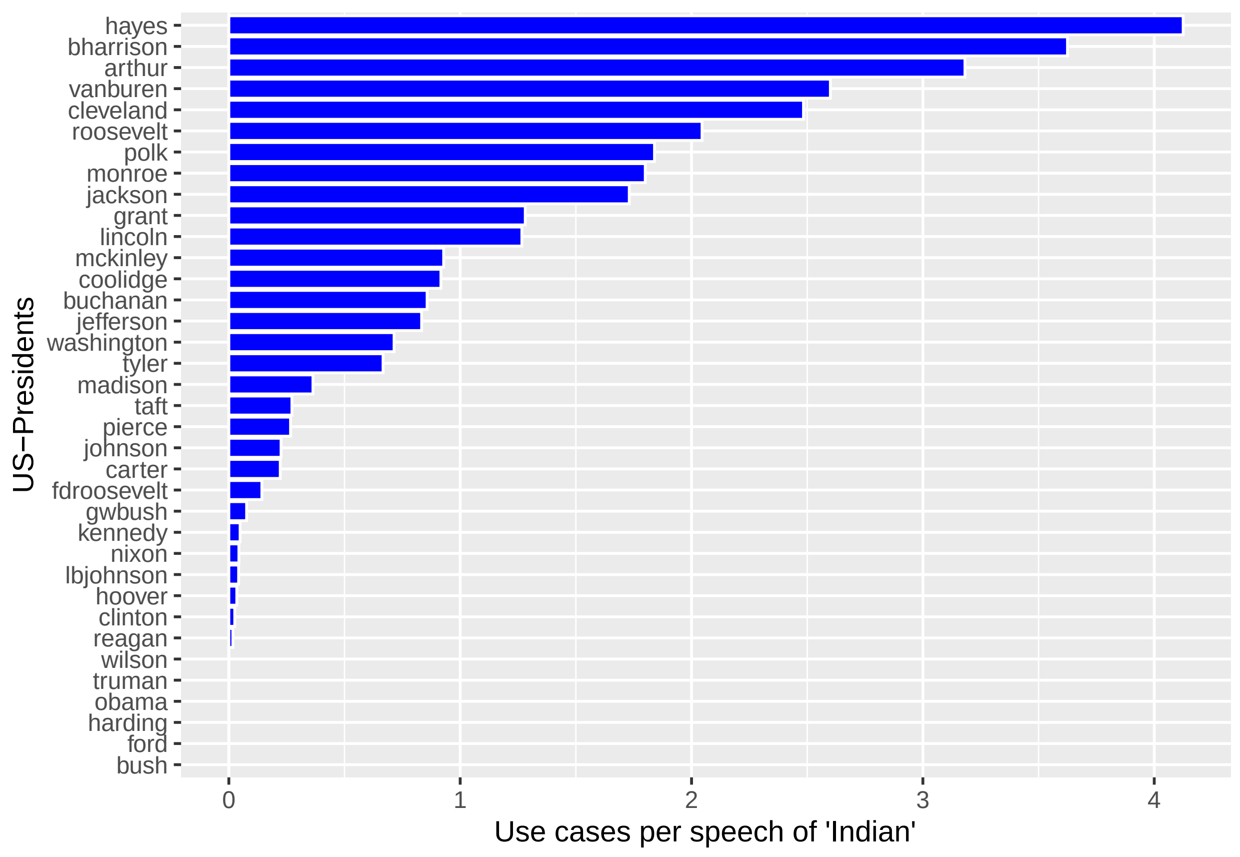
```r
communism <- data.frame(presidents = colnames(pres_pref),
                        count = pres_pref["communism",])
communism %>% ggplot(aes(x=count, y= reorder(presidents,count))) +
                        geom_bar(stat="identity", fill = "blue", color = "white") +
                        labs(x= "Use cases per speech of 'Communism'", y = "US-Presidents")
```



Unsurprisingly, the two presidents leading the score are Truman and Kennedy. Truman's term in office overlapped with the active time of Joseph McCarthy, of McCarthyism fame. And Kennedy's troubles with the Reds are very well known. Most surprising to me is the low use of "communism" by Ronald Reagan, who, together with Kennedy is seen as one of the two key Cold War presidents. Well, at least here in the Old World.

Now for the old-timers. After the United States had taught the British the same lesson the Vietnamese taught them almost 200 years later, their external foes were few. Instead US-president had to deal with the question of how best to deal with the few remaining indigenous Americans who were just not happy about having their stuff taken, their sacred grounds trampled and their people thrown of their ancestral homelands. For which presidents was this an issue?

```
indian <- data.frame(presidents = colnames(pres_pref),
                     count = pres_pref["indian",])
indian %>% ggplot(aes(x=count, y= reorder(presidents,count))) +
                     geom_bar(stat="identity", fill = "blue", color = "white") +
                     labs(x= "Use cases per speech of 'Indian'", y = "US-Presidents")
```



We see that with the exception of Theodore Roosevelt, who was famously hostile to indigenous Americans, the top spots are all taken by the 18th and 19th century presidents. Notably, this drop in use occurred long before the term “Indian” started to be viewed as racist and anachronistic in the wake of the civil-rights movement.

So we see, that our “defining words” exist. One might not be enough, but coupled together, they should give us an idea. However, they are buried under a few, highly frequent words. So, let’s remove them! We will only consider words used less than 1000 times. Also we saw that more than 70% of words only appeared less than 10 times. These are also worthless as predictors and clutter our data to an even larger extend, as each one gets an entire row to itself. Thus we shall say: “Begone clutter!”.

```r
wordfilter <- word_frequency$count < 1000 & word_frequency$count > 10
lftm <- term_matrix[wordfilter,]
# Dimensions of the original term matrix
dim(term_matrix)
```

```
## [1] 34735   914
```

```r
# Dimensions of the cleaned term matrix
dim(lftm)
```

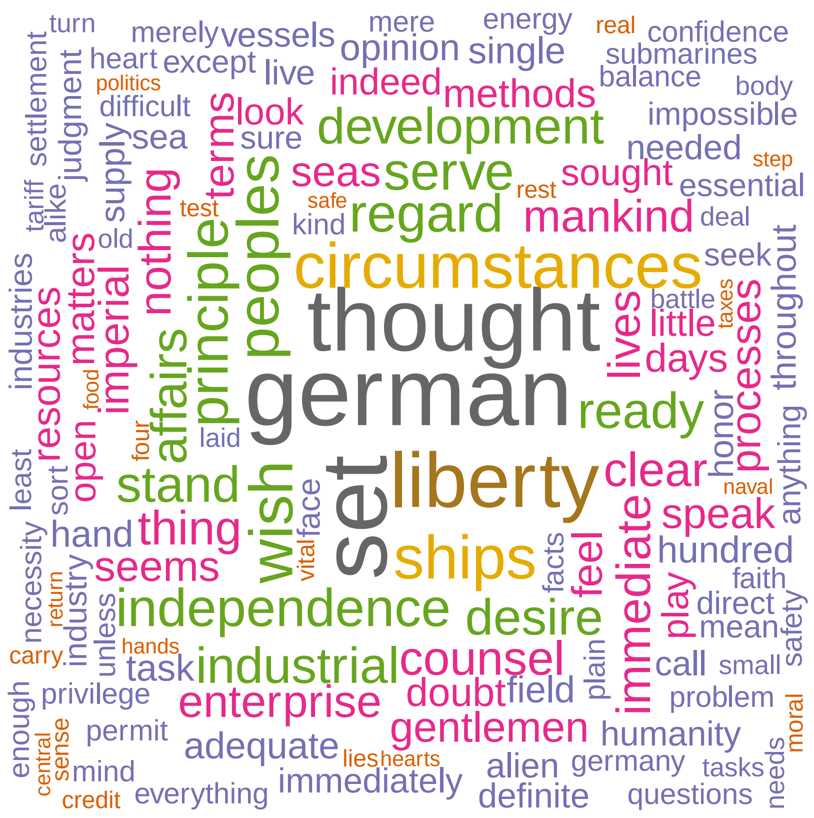
```
## [1] 10105   914
```

Now with this, our word clouds should look much more interpretable. Let’s pick out some presidents, again. We will go with Woodrow Wilson, Abraham Lincoln and Theodore Roosevelt again, but this time also include John F. Kennedy and Barack Obama.

```r
wilson <- lftm[,colnames(lftm) == "wilson"]
lincoln <- lftm[,colnames(lftm) == "lincoln"]
roosevelt <- lftm[,colnames(lftm) == "roosevelt"]
obama <- lftm[,colnames(lftm) == "obama"]
kennedy <- lftm[,colnames(lftm) == "kennedy"]
```

We begin with *Woodrow Wilson*…

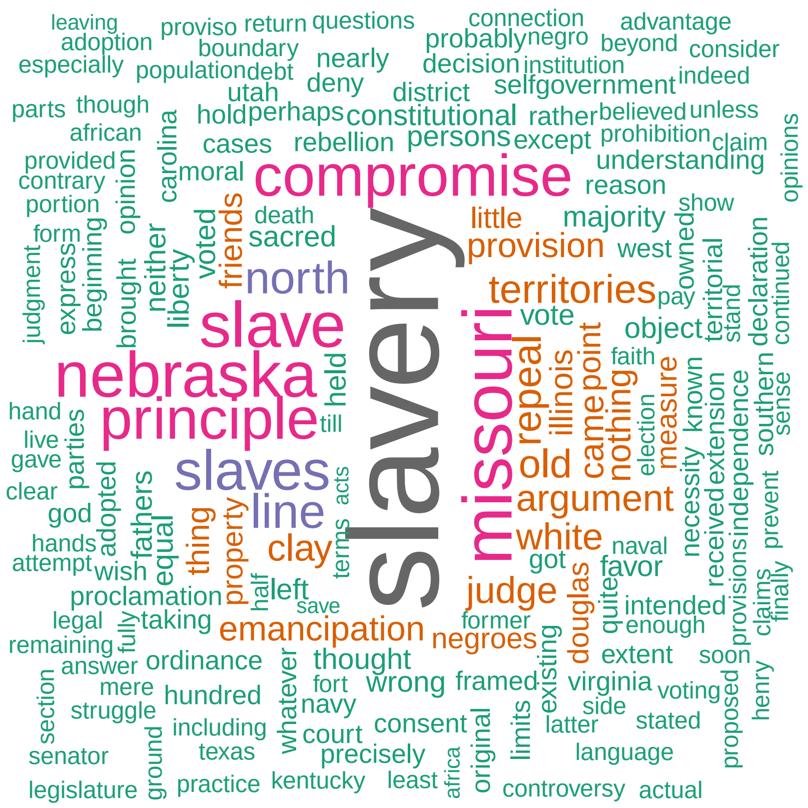
```r
ww <- sort(rowSums(wilson),decreasing=TRUE)
wilsonc <- data.frame(word = names(ww),freq=ww)

wordcloud(words = wilsonc$word, freq = wilsonc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35, scale=c(3,0.05),
          colors=brewer.pal(8, "Dark2"))
```



… who leads with a very big *german*, followed by two almost stopwords. We see *ships*, *liberty*, *principle* and *independence* appear with some frequency, too. These fit into Wilson's proclamations of the independence of small nations and the freedom for all people trapped in Empires. *Ships* might refer to the ships sunk by the German U-Boat campaign during WW1, one of the key factors that swayed public opinion in favor of the war.

Of course it will not spoiler anyone that the topic foremost on the mind of everyone's favorite president *Abraham Lincoln* was…

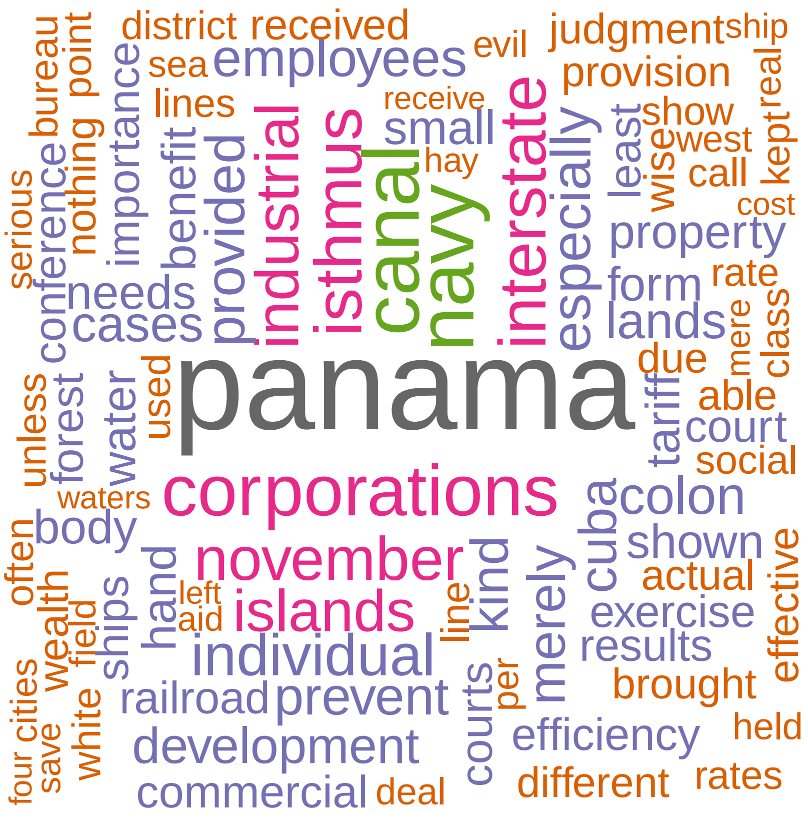
```r
al <- sort(rowSums(lincoln),decreasing=TRUE)
lincolnc <- data.frame(word = names(al),freq=al)

wordcloud(words = lincolnc$word, freq = lincolnc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



… *slavery* and the *emancipation* of *slaves*. The state names *Missouri* and *Nebraska* might refer to Lincolns opposition to the *Nebraska*-Kansas Act and the *Missouri*-Compromise which admitted more slave-holding states to the Union.

As for *Theodore Roosevelt*…

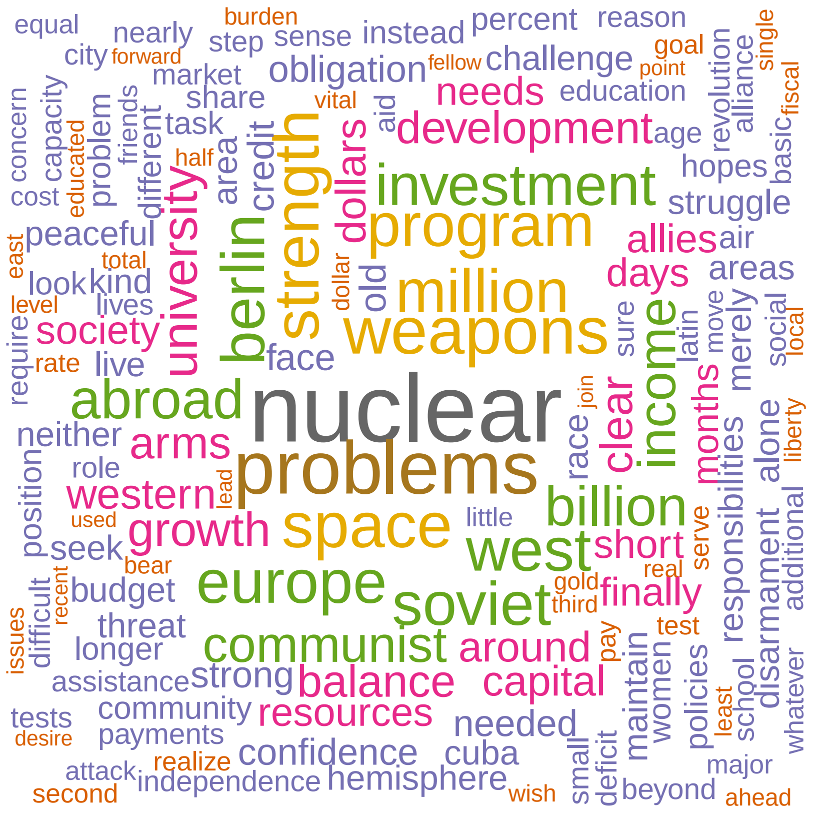
```r
tr <- sort(rowSums(roosevelt),decreasing=TRUE)
teddyc <- data.frame(word = names(tr),freq=tr)

wordcloud(words = teddyc$word, freq = teddyc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2") )
```



... that man rally loved his *canal* that was going through the *isthmus* of *panama*. We can also see clear evidence of his famous trust-busting in *corporations*, *commercial*, *employees* and *industrial*, and his love of national parks in *water* and *forests*. Of course *water* also goes well with *navy* and the Great White Fleet. Roosevelt loved himself his A.T. Mahan.

As for the winner of the "Most handsome US-president" contest, the only ever catholic head of state, *John F. Kennedy*...

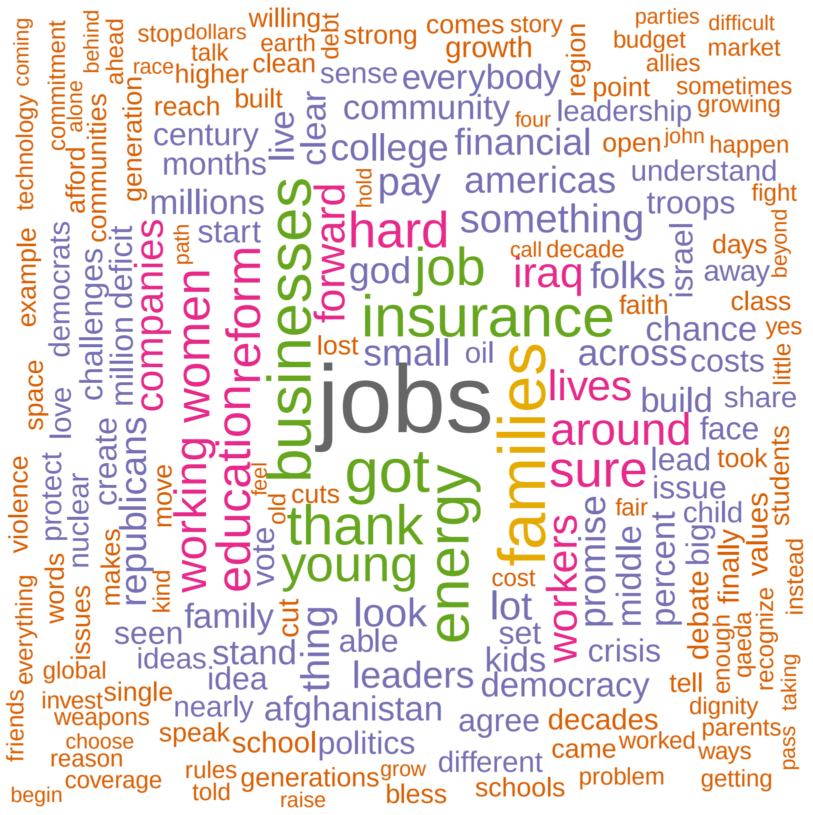
```r
jfk <- sort(rowSums(kennedy),decreasing=TRUE)
jfkc <- data.frame(word = names(jfk),freq=jfk)

wordcloud(words = jfkc$word, freq = jfkc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35, scale=c(3,0.1),
          colors=brewer.pal(8, "Dark2"))
```



… his word cloud is also full of defining terms. From “Ich bin ein *Berlin*er”, to the *space* program and the *problems* with the *communist* threat of *nuclear weapons* in Cuba, JFK should be easily identifiable by his speeches.

Last but not least, the most current US-president on our list, *Barack Obama*.

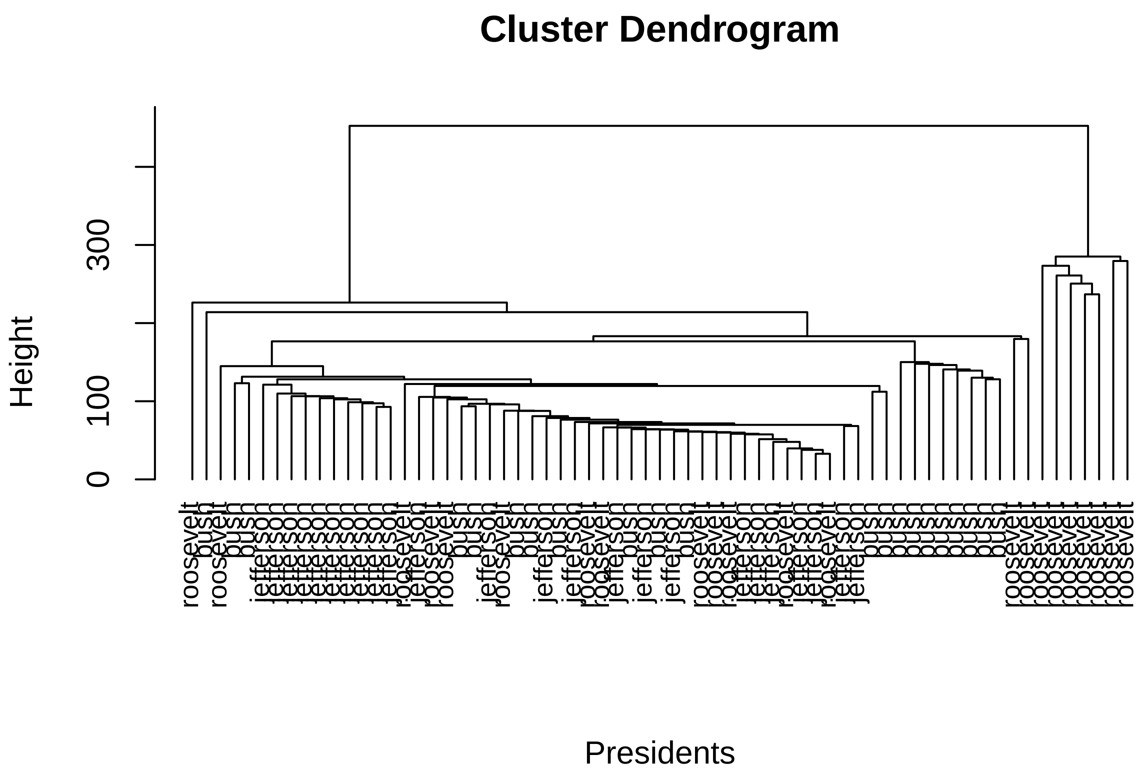
```
bo <- sort(rowSums(obama),decreasing=TRUE)
obamac <- data.frame(word = names(bo),freq=bo)

wordcloud(words = obamac$word, freq = obamac$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35, scale=c(3,0.2),
          colors=brewer.pal(8, "Dark2"))
```



Obama inherited the wars in *Iraq* and *Afghanistan* but more defining where his actions during the 2008 financial crisis that cost a lot of American *jobs*, hurt *businesses*, *workers* and *families* alike. And of course there was the Affordable Care Act, his plan for an extended health *insurance*.

Now the clouds are much clearer. We have reduced our data set by almost two thirds without losing any valuable information. We saw our "defining words" in action and can now, with some confidence, start building our model. But before we start, let's finish Data Exploration by performing a simple cluster analysis on our data set. In a way, this is already a very simple form of machine learning. If our data is good, we should see some decent separation of our presidents. Let's take George Bush the Elder, Thomas Jefferson and Teddy Roosevelt for example:

```r
selected_pres <- c("bush", "jefferson", "roosevelt")
cluster_example <- lftm[, colnames(lftm) %in% selected_pres]

ex_cluster <- dist(scale(t(cluster_example), center = T, scale = T))
tree <- hclust(ex_cluster, method = "ward.D2")
plot(tree, hang = - 1, ylab = "Height", xlab = "Presidents", sub = "", cex = 0.8)
```



We see some very good separation on one side, with distinct "Bush" and "Roosevelt" clusters and a "Jefferson" cluster in the middle, but we also see some intermixing between all three presidents in the center. This might indicate that there are some speeches for which word-choice or topic aren't as full of "defining words" as others. Hopefully, a more sophisticated ML algorithm will be able to differentiate. So let's try some of those!

Model construction

Regularized Support Vector Machine Model

Quantedas supervised learning feature used a SVM model to predict the presidents. Therefore we will start with the same. Our data set is still rather large with over 92 million entries. We will therefore use the `liblineaR` package for both of our models. It comes with two implemented loss-functions for regularization of both the SVM and the logReg model and is optimized for large data sets. As `liblineaR` is fully integrated into caret, we won't even have to find the best parameters ourselves. Caret's `train` function will take care of that.

To start construction of the model, we split the data analogous to what we have done for the bench-marking.

```
set.seed(1337, sample.kind = "Rounding")
index <- createDataPartition(colnames(lftm), p = 0.7, list = FALSE)

testset <- lftm[,-index]
trainset <- lftm[,index]
```

Before we test the SVM model, we want to see how well we'd fare by just guessing presidents

```
dim(testset)
```

```
## [1] 10105   258
```

```
rs <- sample(presnames, 258, replace = T)
confusionMatrix(as.factor(rs), as.factor(colnames(testset)))$overall["Accuracy"]
```

```
##   Accuracy
## 0.03100775
```

Not very well... An accuracy of around 3 % is really nothing to write home about. But it puts our models in perspective. Because we have 36 classes and a highly unbalanced dataset, the 60% accuracy we got from `quanteda` is already an immense gain. But of course we want more than that. So let's fit the SVM model to the train data...

```
svm_model <- train(t(trainset), colnames(trainset), method = "svmLinear3")
```

... and observe how well it fits.

```
selffit2 <- predict(svm_model, t(trainset))
confusionMatrix(as.factor(selffit2), as.factor(colnames(trainset)))$overall["Accuracy"]
```

```
## Accuracy
##        1
```

Of course, we also had a perfect fit for the `quanteda` model. And the overall accuracy still was only about 60%. Does our model perform better than that?

```r
svm_fit <- predict(svm_model, t(testset))
svm <- confusionMatrix(as.factor(svm_fit),as.factor(colnames(testset)))
svm$overall["Accuracy"]
```

```
##  Accuracy
## 0.6860465
```

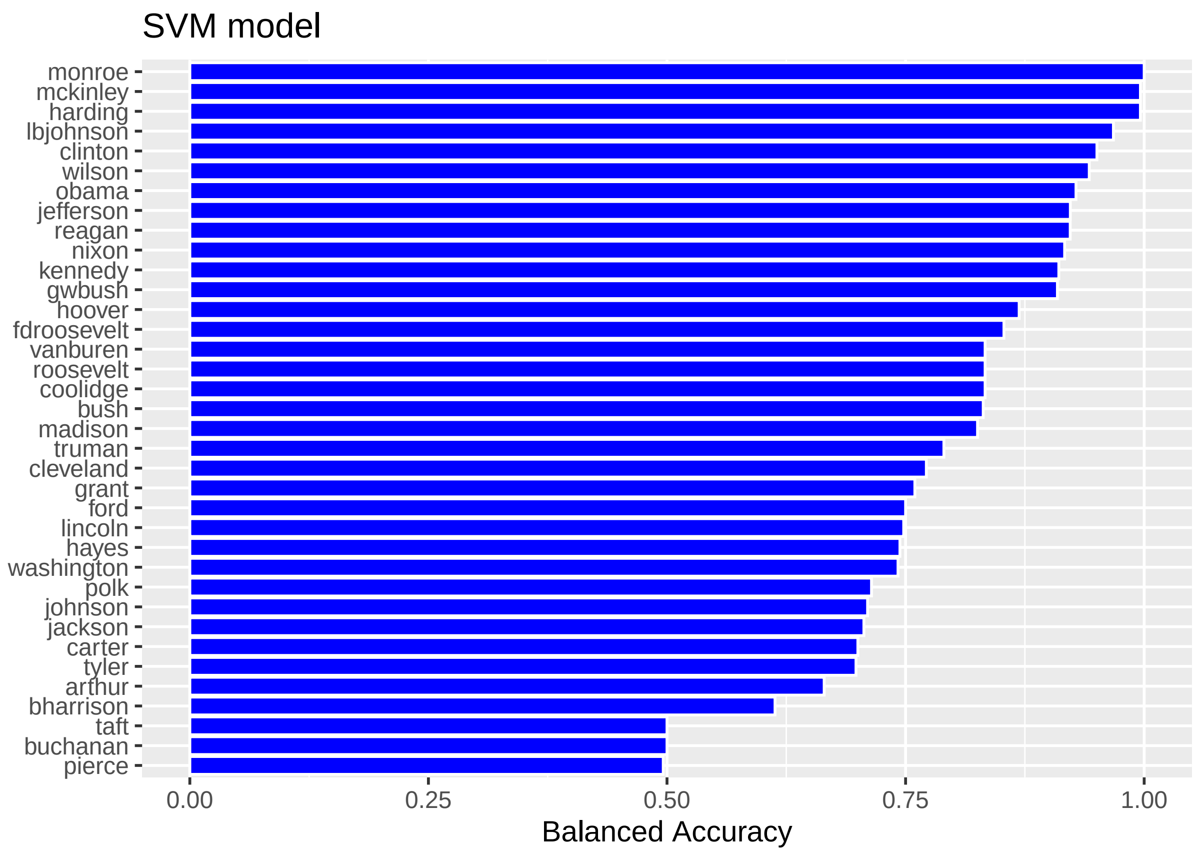
It does. We have improved our accuracy by 0.08 and therefore already beaten the base line. Let's save this.

```r
model_performance <- cbind.data.frame(model_performance, SVM = svm$overall["Accuracy"])
model_performance
```

```
##           quanteda       SVM
## Accuracy 0.6021341 0.6860465
```

And of course we are interested in the balanced accuracy per president!

```r
# Calculate and plot balanced accuracy for each president
bal_acc_svm <- data.frame(president = names(svm$byClass[,"Balanced Accuracy"]),
                          "balanced accuracy" = svm$byClass[,"Balanced Accuracy"])
bal_acc_svm <- bal_acc_svm %>% mutate(president = sapply(1:nrow(bal_acc_svm), function(x) {
  (gsub(".*: ", "", bal_acc_svm$president[x]))
}))
bal_acc_svm %>% ggplot(aes(y = reorder(president, balanced.accuracy),
                           x = balanced.accuracy)) + geom_col()+
  geom_col(color = "white", fill = "blue") +
  labs(x = "Balanced Accuracy",
       y = "",
       title = "SVM model")
```



Interestingly, we see now obvious pattern in predictability. The first four presidents, with 100 % or close to 100 % accuracy have few speeches in our data set, but "data rich" presidents like LBJ, Woodrow Wilson and Barack Obama perform very well too. In contrast, James Buchanan and Franklin Pierce have few speeches and seem very hard to predict. Now we have achieved our goal, but some predictions are still not particularly good. Can we improve on the SVM model?

Regularized Logistic Regression Model

We could try other model contained in the `libLineaR` library: Regularized logReg. Same as for the SVM, we don't need to pick the parameters for the regularization, but can simply rely on caret for it.

```r
reglog_model <- train(t(trainset),
                      colnames(trainset), method = "regLogistic")
```

Again we can check how well the training data is fitted, but by now we know the answer to that...

```r
selffit3 <- predict(reglog_model, t(trainset))
confusionMatrix(as.factor(selffit3),
                      as.factor(colnames(trainset)))$overall["Accuracy"]
```

```
## Accuracy
##        1
```

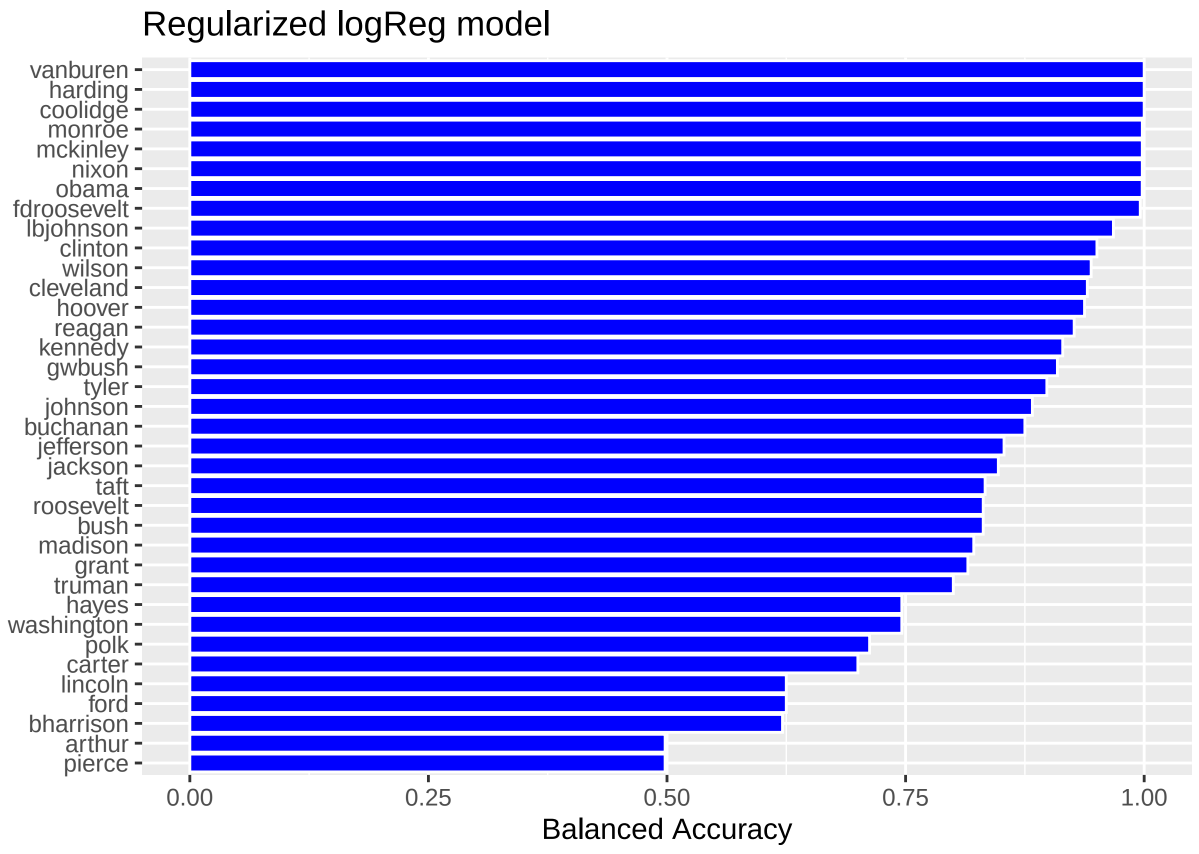
The question remains - can we beat the 0.68 accuracy of our SVM model?

```r
reglog_fit <- predict(reglog_model, t(testset))
reglog <- confusionMatrix(as.factor(reglog_fit),as.factor(colnames(testset)))
reglog$overall["Accuracy"]
```

```
##  Accuracy
## 0.7713178
```

We can. Significantly so. By almost 0.09. While not perfect, this means our model is right, three quarters of the time. Given the imbalanced nature of the data and the complexity of human language, this is not bad. And how does the graph change for the balanced accuracy?

```r
bal_acc_rlr <- data.frame(president = names(reglog$byClass[,"Balanced Accuracy"]),
                          "balanced accuracy" = reglog$byClass[,"Balanced Accuracy"])
bal_acc_rlr <- bal_acc_rlr %>%
                          mutate(president = sapply(1:nrow(bal_acc_rlr),
                             function(x) {
                              (gsub(".*: ", "", bal_acc_rlr$president[x]))
                             }))
bal_acc_rlr %>% ggplot(aes(y = reorder(president, balanced.accuracy),
                             x = balanced.accuracy)) + geom_col()+
                          geom_col(color = "white", fill = "blue") +
                          labs(x = "Balanced Accuracy",
                               y = "",
                               title = "Regularized logReg model")
```



Interestingly, the improvements are mainly made in the middle section of the graph. We also have more almost-perfect predictions. However, at the bottom, two presidents remain terribly inaccurate. The only thing that changed there was that we switched Buchanan with Chester A. Arthur as the worst performing category.

Let's add this to the result table.

```r
model_performance <- cbind.data.frame(model_performance, regLog = reglog$overall["Accuracy"])
```

Results

We can now have a closer look at our results.

```
model_performance
```

```
##           quanteda       SVM    regLog
## Accuracy 0.6021341 0.6860465 0.7713178
```

As we set out to do, we improved quite significantly on the `quanteda` model with an accuracy gain of almost 17 %. Sadly, neither SVM nor our regularized LogReg model provide model specific variance importance. We can therefore, with those models, not verify if our hypothesis of "defining words" is accurate. I tried models that provide such variable importance such as random forest, partial least scores or extreme gradient boosting, but they all underperformed compared to the LibLineaR models. However, we can still try one of them. One of the fastest models that gives reliable feature importance values is `pam`. PAM stands for "Partitioning around Medoids" and is a variant of k-means clustering[8]. I have used it frequently for the analysis of RNA sequencing data (a task for which it is highly suited). When we run it, we see that it underperforms the `quanteda` base line model.

```
#Fit PAM model and predict testset (the numeric string comes from the method)
nsc_model <- train(t(trainset), colnames(trainset), method = "pam")
```

```
## 1234567891011121314151617181920212223242526272829301111111111111111111111111
```

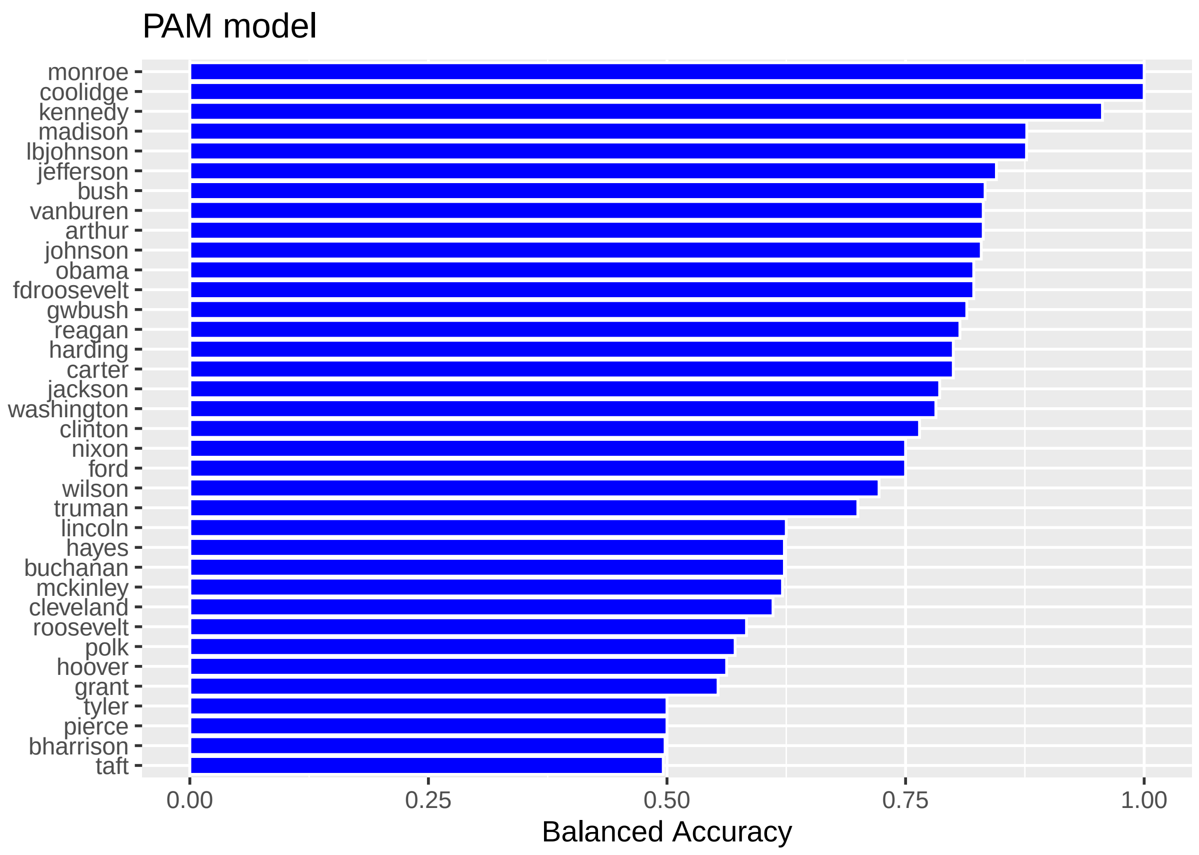
```
nsc_fit <- predict(nsc_model, t(testset))
nsc <- confusionMatrix(as.factor(nsc_fit),as.factor(colnames(testset)))
# Model accuracy
nsc$overall["Accuracy"]
```

```
##  Accuracy
## 0.5232558
```

```
# Mean balanced accuracy
mean(nsc$byClass[,"Balanced Accuracy"])
```

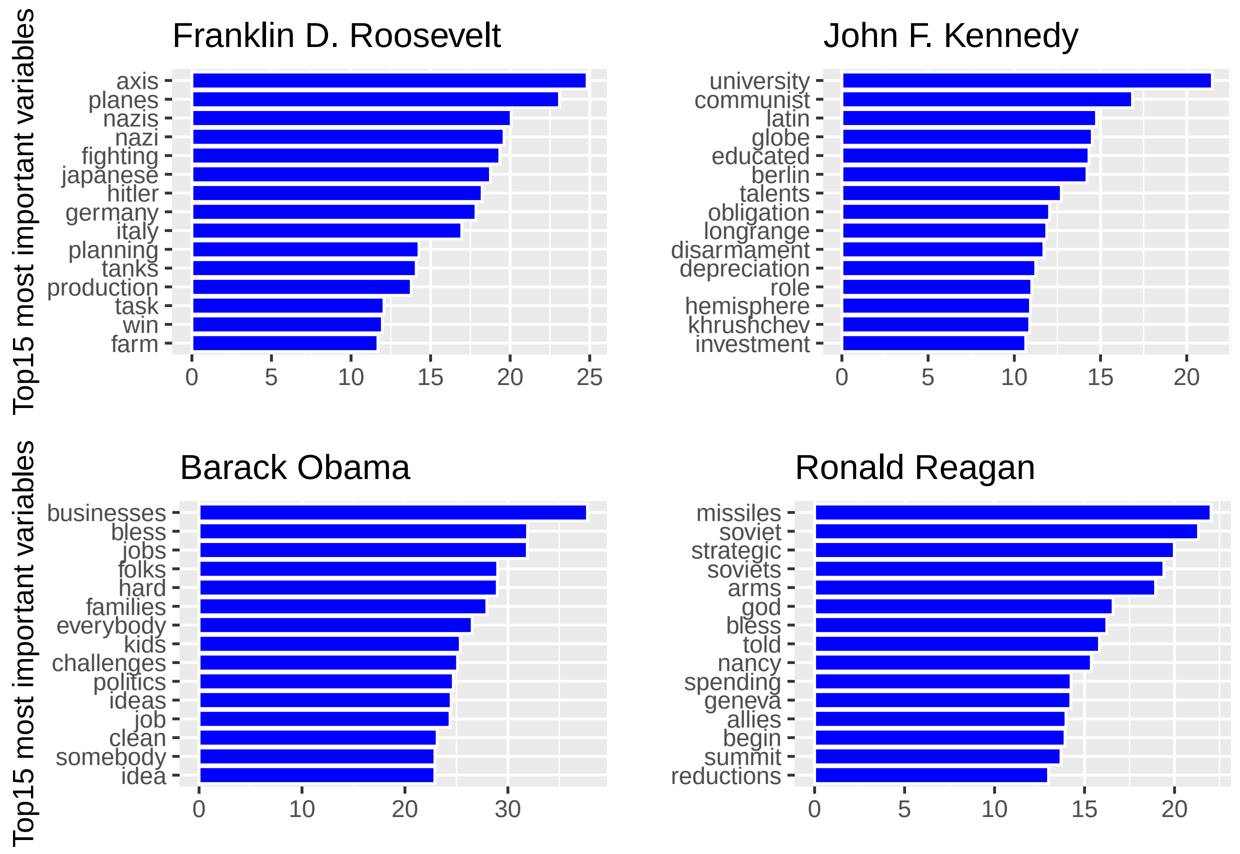
```
## [1] 0.7324165
```

```
bal_acc_nsc <- data.frame(president = names(nsc$byClass[,"Balanced Accuracy"]),
                          "balanced accuracy" = nsc$byClass[,"Balanced Accuracy"])
bal_acc_nsc <- bal_acc_nsc %>% mutate(president = sapply(1:nrow(bal_acc_nsc), function(x) {
  (gsub(".*: ", "", bal_acc_nsc$president[x]))
}))
bal_acc_nsc %>% ggplot(aes(y = reorder(president, balanced.accuracy),
                           x = balanced.accuracy)) + geom_col()+
  geom_col(color = "white", fill = "blue") +
  labs(x = "Balanced Accuracy",
       y = "",
       title = "PAM model")
```



However, it still produces at least some reliable results for several presidents, including JFK, FDR, Obama and Reagan. Those we had included in our `wordcloud` analysis so we know their presumtive "defining words". So let's investigate which words PAM considered the most important features for these four presidents.

```
variableImp <- varImp(nsc_model)$importance
examples <- variableImp %>% select(fdroosevelt, kennedy, reagan, obama)
examples <- cbind(words = rownames(examples), examples)



f <- examples %>% slice_max(n=15, order_by = fdroosevelt) %>%
            ggplot(aes(y=reorder(words, fdroosevelt), x = fdroosevelt)) +
            geom_col(fill="blue", color = "white") +
            labs(x = "",
                 y = "Top15 most important variables",
                 title = "Franklin D. Roosevelt")

k <- examples %>% slice_max(n=15, order_by = kennedy) %>%
             ggplot(aes(y=reorder(words, kennedy), x = kennedy)) +
             geom_col(fill="blue", color = "white") +
             labs(x = "",
                  y = "",
                  title = "John F. Kennedy")

rr <- examples %>% slice_max(n=15, order_by = reagan) %>%
             ggplot(aes(y=reorder(words, reagan), x = reagan)) +
             geom_col(fill="blue", color = "white") +
             labs(x = "",
                  y = "",
                  title = "Ronald Reagan")

o <- examples %>% slice_max(n=15, order_by = obama) %>%
            ggplot(aes(y=reorder(words, obama), x = obama)) +
            geom_col(fill="blue", color = "white") +
            labs(x = "",
                 y = "Top15 most important variables",
                 title = "Barack Obama")

ggarrange(f, k, o, rr,
            ncol = 2, nrow = 2)
```



And we can clearly see that at least for PAM, our hypothesis had some merit. Nearly all important features for FDR are related to the war, for JFK they are often Cold-War related, Obama was predicted by terms relating to jobs and the economy and Reagan again by his (different) cold-war rhetoric. While we can't judge one algorithm by another, we can still see that these words are likely to play a role in our predictions. After all, our selected methods try to maximize the distance (geometric and statistic) between the classes, and the best way to do this is by focusing on differences in word-choice.

Discussion

Summary

In this report we analyzed a collection of presidential speeches covering 36 US-presidents. We showed that era specific challenges and policy goals influenced each president and determined their choice of words and topic emphasis. In our hypothesis, these special terms which we termed “defining words”, would be the determinants upon which the ML algorithms would create their models. We performed some basic data cleaning using textmining packages like `tm` and removed words we considered irrelevant as predictors, in order to shrink the data set to an acceptable size. We then employed different machine learning methods to beat the base line set by text analysis tool `quanteda`. The `LibLineaR` package proved to be the best at generalizing for our purposes, beating methods such as random forest, knn and PLS (data not shown). To demonstrate the importance of “defining words”, we showed the most important, model-specific variables in a PAM model and could show that they matched quite well with what we predicted. However, due to the lack of model-specific variable importance values in both `LibLineaR` methods, we cannot with certainty say that they are employed in these.

Limitations and perspective

Human speech is a complicated thing and recent advances in speech recognition have shaped the lifes of millions of Alexa, Bixby or Google Assistant users, users of smart homes, smart cars or even smart fridges. Our approach is therefore a very basic one as it does not consider meaning or syntax. Textmining tools that are able to do this exist and `quanteda` itself contains this capability. The fact that we could still achieve >75% accuracy with such a simple model, using unbalanced data with more than 10 000 features, is therefore respectable. More sophisticated methods could take into account the above mentioned factors or perform more advanced feature selection. In the course of this exercise I have tried multiple things, including the removal of low-variance features or high-correlation features, the scaling of data or feature importance selection by different methods. However, each approach resulted in a lower overall accuracy which was irritation, as it ran contrary to what I’m used to in sequencing data analysis. It seems therefore that I do not yet fully understand the data set, despite our extensive data exploration. The reduction of the data to only “defining words” should have increased the accuracy but it didn’t, suggesting that my hypothesis is not entirely correct. For future approaches, I’d like a more comprehensive data set, containing more speeches and maybe even written works such as the Federalist Papers, letters to friends and family, etc. In the end however, a perfect method, short of an algorithm that memorized every single speech and therefore has a 100% accuracy, might not exist. Some people, including US-presidents, are not rhetorically eloquent, don’t make for good year book quotes or facebook posts, inherited problems from their predecessors and therefore mirror their speeches, etc. There are things in politics that are repeated *ad nauseam* and can make two politicians sound almost identical. Designing algorithms that detect the minute differences between such politicians is a challenge to which I am not yet suited, I’m afraid.

Acknowledgments

I want to thank you, dear peer and course instructor, who has taken his/her/their time to read this. I started this course because Corona had me locked down and I knew I was going to need more R skills for my thesis. I never thought I would take me this long, would be that much fun or improve my skills to such a degree and I have recommended this course to pretty much everyone. Thanks also goes to Prof. Rafael Irizarry for making this course and everyone of the instructors who answered questions in the forums. And finally I want to thank my GF and my cat who supported my while doing this, even though it cut into our time together. Well, the cat got ample snacks in return so that fat furry bastard at least can’t complain.

References

[1] http://www.thegrammarlab.com/?nor-portfolio=corpus-of-presidential-speeches-cops-and-a-clintontrump-corpus

[2] Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S. and Matsuo, A., 2018. quanteda: An R package for the quantitative analysis of textual data. Journal of Open Source Software, 3(30), p.774.

[3] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R. and Lin, C.J., 2008. LIBLINEAR: A library for large linear classification. Journal of machine learning research, 9(Aug), pp.1871-1874.

[4] https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16

[5] https://www.mzes.uni-mannheim.de/socialsciencedatalab/article/advancing-text-mining/#supervised

[6] https://www.youtube.com/watch?v=ZaFRJmBhWPo&ab_channel=QueAutoComproTV

[7] https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf

[8] https://www.cs.umb.edu/cs738/pam1.pdf