Big Data and Analytics (CS6444-10)
Class Project #2 - Group #1
24 July 2017
Jing Si, Hassan D. M. Sambo, James Frick, Ian Ferri

1. Throughout our analysis and experimentation, we utilized the following packages: **tm**, **NLP**, **stringr**, **ggplot2**, **reshape**, **RColorBrewer**, and **wordcloud**.

2. To identify the top 100 most prolific email writers, we first needed to get the individual email files into a format easy to work with in R. We did some basic scripting in Python to extract text from each sent message and used the same filtering methodology used in Project #1 to filter out duplicates by only keeping messages with unique "from - to - datetime" combinations. We also did our best to remove any forwarded messages, reply-chains, and any perceived "administrative" accounts that sent thousands of unique emails. Once we narrowed down our list of top 100 emailers, we stored their email addresses, and each email message in a pipe-delimited .csv file to be read into R, as depicted below, for our analysis.

```r
# use tm
library(tm)

#============================================
#set directory and read files
#============================================

#set the directory
setwd("../BDA_P1-master")
getwd()

# read in the emails. Col 1 is emailer, Col 2 is the text of the email as a "document"
emails = read.csv("../allTopEmailers.csv",sep='|')

# turn the documents into a corpus, as in lecture 4
corpus = VCorpus(VectorSource(emails$txt))

# inspect VCorpus
inspect(corpus)
```

```
[[997]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 346

[[998]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 153

[[999]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 73

[[1000]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 48
```

3. inspect(corpus) gave us an output for the first 1,000 documents in our corpus. To the right, there is a sample of these outputs. As you can see each document has the same metadata characteristics, but they all vary in character count.

4. corpus[[1]][1] gives us a plaintext output of the content, whitespace characters and all, within the first PlainTextDocument in our corpus.

```
> corpus[[1]][1]
$content
[1] "\n\nAmerex West-\nAll deals checked out w
ell.\n\nBloomberg\nAll deals checked out well.
\n\n\n\nThanks!!"
```

5. Experimenting with functions in Lecture 4:

```
> corpusTDM=TermDocumentMatrix(corpus)
> corpusTDM
<<TermDocumentMatrix (terms: 14804, documents: 1610)>>
Non-/sparse entries: 54473/23779967
Sparsity           : 100%
Maximal term length: 82
Weighting          : term frequency (tf)
```

6. Since the mail files contain the receiver and the sender's email address, we write a function Remove email to clean that.

```
RemoveEmail <- function(x) {
  require(stringr)
  str_replace_all(x,"[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+", "")
}
```

7. Before using content_transformer(), we could inspect the corpus and see the content of these mails. As you can see below, the #1000 mail contains lots of slashes and numbers which does not need in the analysis of text. So we need to remove them. Also, after removing the punctuations and numbers, we found many extra white spaces which is quite noising. So again we remove them.

```
[[998]]
[1] \n\ni'll call you tomorrow and fill you in on the conversation\n\nlindai also made a sma
ll commitment on tobacco issues that i'll need to \ntalk to you about!

[[999]]
[1] \n\n\tfreddi greenberg <>\n\t05/23/2001 10:35 am\n\t\t \n\t\t

[[1000]]                                          content_transformer(FUN)
[1] \n\n\tsusan m landwehr\n\t01/23/2001 09:52 pm\n\t\t \n\t\t

 [ reached getOption("max.print") -- omitted 610 entries ]
```

```
> corpusTrans2<- tm_map(corpusLowCase, RemoveEmail)
> corpusTrans2 <- tm_map(corpusTrans2,removeNumPunct)
> corpusClean2<- tm_map(corpusTrans2,stripWhitespace)
```

8. After the clean of the redundancy, we can see from the screenshot below that the text are much clean now.

```
[[998]]
[1]  ill call you tomorrow and fill you in on the conversation lindai also made a small comm
itment on tobacco issues that ill need to talk to you about

[[999]]
[1]  freddi greenberg am

[[1000]]
[1]  susan m landwehr pm

 [ reached getOption("max.print") -- omitted 610 entries ]
```

Now we need to transform the content so we could generate the term document matrix.

```
> cleanTDM <- TermDocumentMatrix(corpusClean)
> cleanTDM
<<TermDocumentMatrix (terms: 7954, documents: 1610)>>
Non-/sparse entries: 46426/12759514
Sparsity           : 100%
Maximal term length: 48
Weighting          : term frequency (tf)
```

Compared to the previous corpus, the number of terms decreased from 14804 to 7954. Almost half cutted!

9. Then remove the stop words from the corpus.

```
> #Remove stopwords from the corpus
> myStopwords <- c(stopwords('english'))
> removeStop <- tm_map(corpusClean, removeWords, myStopwords)
> inspect(removeStop[1:10])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 10

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 71

[[2]]
```

10. We can see in the corpus names clean TDM, the sparsity are almost 100%. That is because the data set are too large that many terms only appear no more than once. So we are trying to remove sparse tems to an appropriate level. After removing the sparse tems, the terms left are only 178.

```
> removeSparse<- removeSparseTerms(corpusTDM2, 0.98)
> removeSparse
<<TermDocumentMatrix (terms: 178, documents: 1610)>>
Non-/sparse entries: 12344/274236
Sparsity           : 96%
Maximal term length: 11
Weighting          : term frequency (tf)
```

11. Find terms with a frequency of 5 or more.

```
> freq.term=findFreqTerms(removeSparse,lowfreq = 5)
> freq.term
  [1] "access"      "additional"  "address"     "agreement"   "also"
  [6] "amerex"      "america"     "amount"      "approval"    "april"
 [11] "attached"    "available"   "back"        "best"        "bill"
 [16] "bloomberg"   "bob"         "broker"      "business"    "buy"
 [21] "buys"        "c"           "call"        "can"         "center"
 [26] "change"      "changes"     "check"       "chris"       "comments"
 [31] "company"     "contact"     "contract"    "corp"        "currently"
 [36] "d"           "date"        "day"         "deal"        "deals"
 [41] "done"        "dont"        "due"         "eb"          "ect"
 [46] "ees"         "either"      "email"       "end"         "energy"
 [51] "enron"       "fax"         "feel"        "find"        "following"
 [56] "forecasting" "forward"     "frazier"     "free"        "friday"
```

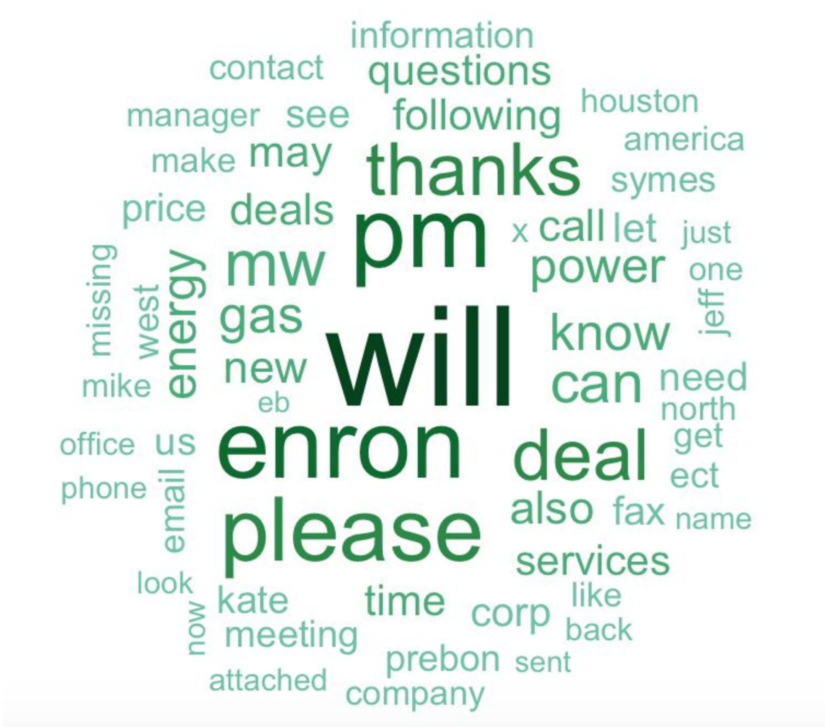
12. Find words associated with “america”

```
> findAssocs(corpusTDM2, "america", 0.25)
$america
        north             street          flynn          smith
         0.80               0.58           0.48           0.46
          fax                 eb           pulp           corp
         0.43               0.42           0.42           0.40
        phone multicurrencycross          debra     perlingiere
         0.40               0.39           0.37           0.37
       except            foreign       patricia         rivera
         0.36               0.36           0.36           0.36
        texas               isda          legal      argentina
         0.36               0.35           0.35           0.34
   intramonth            houston      maranzana         brazil
         0.34               0.33           0.33           0.32
```

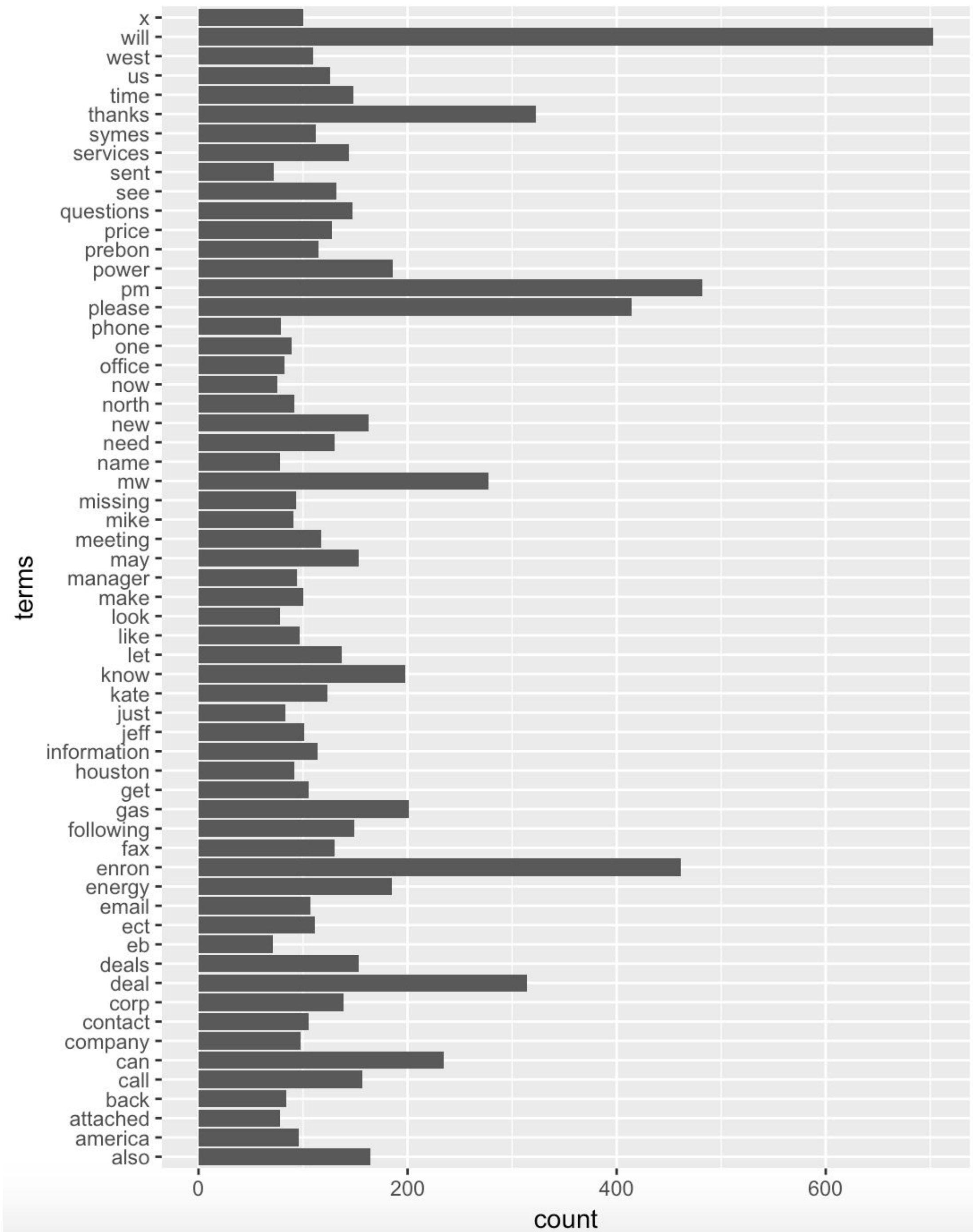
13. Find the frequency of each tem

```
> term.freq<- rowSums(as.matrix(removeSparse))
> term.freq<-subset(term.freq, term.freq>5)
> df<- data.frame(term = names(term.freq), freq = term.freq)
> term.freq
   access additional    address  agreement       also     amerex    america
       55         44         42         78        164         87         96
   amount   approval      april   attached  available       back       best
       38         76         90         78         49         84         36
     bill  bloomberg        bob     broker   business        buy       buys
       49         50         59         55         70         72         74
        c       call        can     center     change    changes      check
       55        157        235         80         51         54         38
    chris   comments    company    contact   contract       corp  currently
       50         67         98        105        116        139         39
```

14. The word cloud after removing the sparse item



15. gg plot after removing the sparse items



c,d) To compute the topics in the corpus, we employed Latent Dirichlet Allocation using the topicmodels package in R. We took the stopword-removed, densified VCorpus and ran a 20-topic LDA model on it.

```
# use Latent Dirichlet Allocation to determine topics:
topicModel = LDA(termDocMat,20)

# get topic-document matrix
topDocMat = as.data.frame(topicModel@gamma)

# get top topics per doc and count them
toptopics = as.data.frame(cbind(document = row.names(topDocMat),
                  topic = apply(topDocMat,1,function(x) names(topDocMat)[which(x==max(x))[1]])))

# print the number of documents with that topic among its top topics
table(toptopics$topic)
```

This returned a fit LDA model, whose gamma matrix represents the topic-by-document matrix. We can use this matrix to compute the top topics for each document. We can then group the dataframe by topic number and count the rows, which means gives us the following document-topic counts:

Topic	Doc Count
1	449
2	328
3	414
4	600
5	307
6	408
7	309
8	378
9	434
10	542
11	285
12	619
13	286
14	520
15	402
16	358
17	520
18	209
19	395
20	176

e. For the 5 largest documents of the 25, draw the dendrogram and the word cloud for each. Include in your writeup.

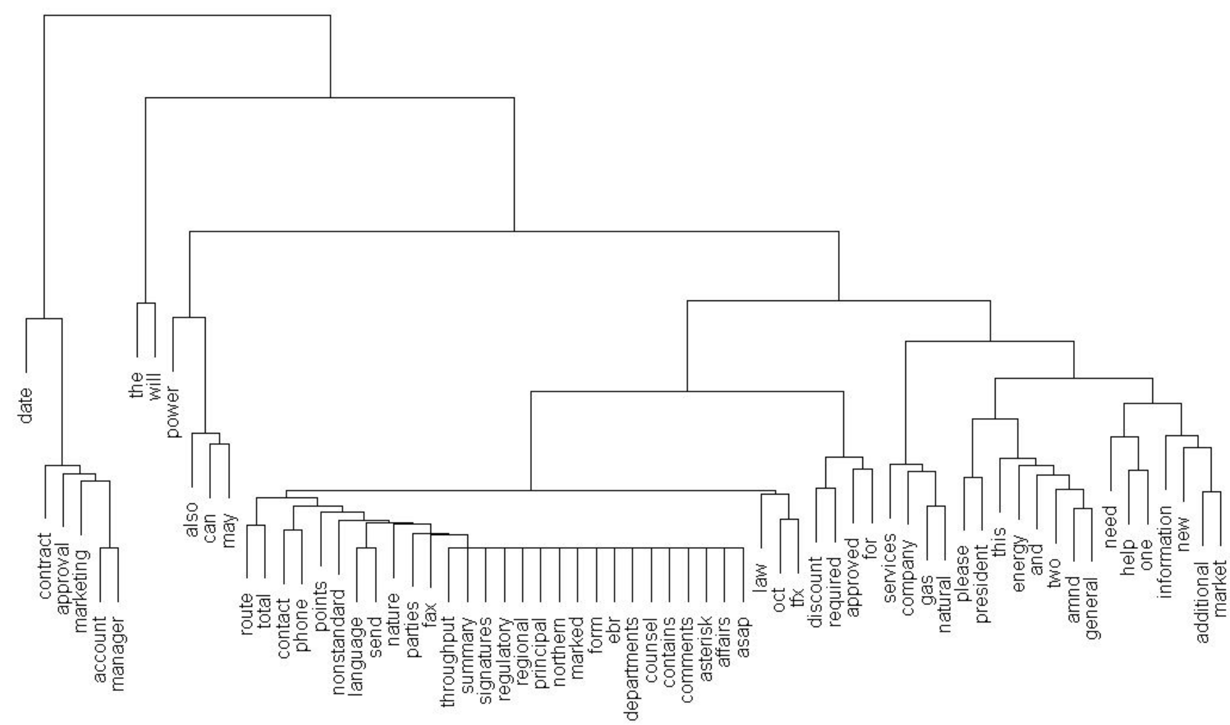
We can identify the top 25 documents and then subset to the top 5 documents. After doing so, we can plot the dendrogram of word usage. We first have to sparsify the terms (as before) in order to make the dendrogram visually interpretable.

```r
# ---------------Get top 25, top 5 documents-----------------

# subset to the longest documents
asDF = data.frame(text=unlist(sapply(cleanCorpus, `[`, "content")),
                  stringsAsFactors=F)
asDF$lengths = nchar(asDF$text)

# top 25 longets emails in a dataframe
asDFTrunc = asDF[with(asDF,order(-lengths)),][1:26,1]
```

Cluster Dendrogram



We can then provide the wordcloud of each of the top 5 documents:

Document 1:

Document 2:

Document 3:

Document 4:

Document 5:

As seen in the visualizations above, the top 5 documents shared several (non-stopword) words, including 'power' and 'date.' The word clouds give us a quick view into the overall content or topics in the emails. For example, document 4 is an email discussing football, while document 1 discusses some sort of board meeting and its outcome/agreement.

g. Prior to removing the punctuation, find the longest word and longest sentence in each document from the 25 largest documents.

Longest term: 51 characters

Longest Sentence: 1058 characters

h. There are many sentences contained within the top 25 documents. We used the openNLP implementation of a MaxEnt tokenizer model to sentence tokenize the emails. We then stripped the punctuation and determined the length of each sentence. R output cleaned up in Excel below.

Top 14 sorted by Length

Length	Freq
0	8
3	1
5	1
6	1
8	2
9	1
11	3
12	1
14	2
15	2
16	2
18	5
19	2
20	10

Top 14 sorted by Freq

Length	Freq
48	14
57	14
43	11
56	11
20	10
67	10
69	9
70	9
80	9
0	8
24	8
32	8
39	8
49	8

j. Analyze word frequency using functions from package zipfR.

Screenshots provided above of experimentation with zipfR package.

Lessons Learned about Data Science:

This assignment introduced to us the basic components of natural language processing and how to apply several R packages to our own data sets in the future. In addition to analyzing the metadata of the Enron email dataset to identify key employees, hierarchies, and cliques, this project helped us explore the content of their email communications to potentially better define the relationships between the nodes in our original graph. In this sense, Project 2 demonstrated to us the need for an iterative approach to data science, where subsequent analysis is used to enhance previous methods and refine prior assumptions. Per Lecture 1, data science is a “process of continuous discovery - not only knowledge, but of new analytics.” For example, exploring the data with text analytics packages helped us identify “administrative” email accounts that we may have previously assigned too much influence to within the Enron organization. Analyzing the content of their emails demonstrated that assessing relationships and influence with quantitative data doesn’t provide the entire story.

In addition, this project helped us understand the importance of spending a considerable amount of time up-front cleansing your data to streamline ingestion and later use with R’s text analytics packages. Troubleshooting numerous error messages related to data formatting issues and package installations/dependencies drove this point home very clearly. Every data science project requires some form of consolidating messy data, re-formatting it to extract the values of particular interest, and reading it into a tool such as R with relevant packages installed, so the lessons learned will undoubtedly be applicable in our futures.