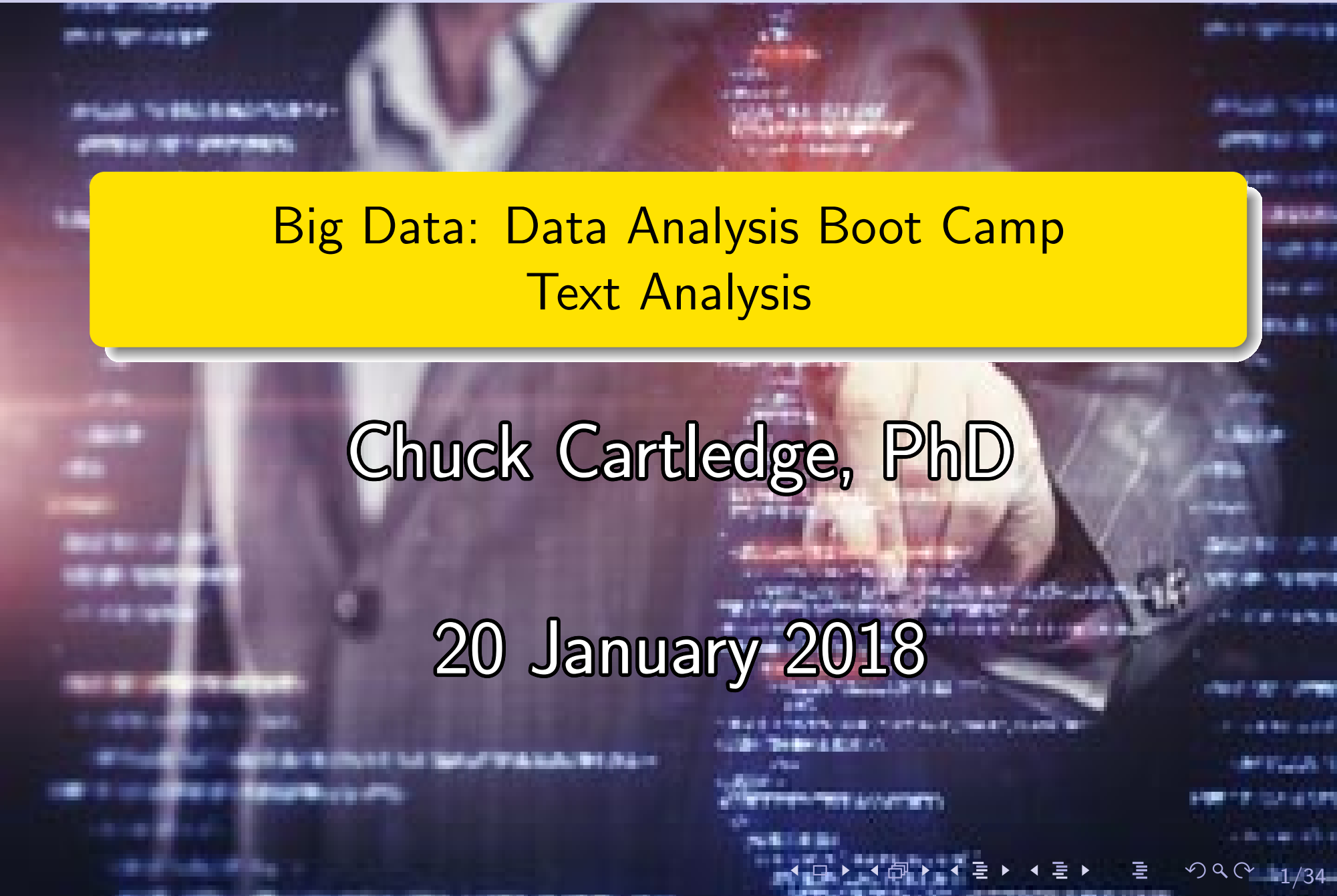


Big Data: Data Analysis Boot Camp
Text Analysis

Chuck Cartledge, PhD

20 January 2018

Table of contents (1 of 1)

What are we going to cover?

We're going to talk about:

- Differences between numerical and textual data analysis.
- Define common textual data analysis terms and ideas.
- Use different textual analysis tools (*knn*, naïve Bayes, logit, and support vector machines)

Processing textual data is messy.

With numerical data, there are a limited number of ways to get data ready for analysis:

1. Ignore records that are missing/incomplete
2. Fill in missing values (mean, mode, estimated)
3. Accept incomplete records and adjust the uncertainties

Textual data is harder. Data may be complete, but very hard to get ready for analysis.

Textual "data wrangling"

There are a few "normal" processing steps to prepare textual data from analysis:

1. Change all text to the same case (usually lower case)
2. Remove all non-textual glyphs (punctuation marks and so on)
3. Remove all numbers
4. Remove all "stop words" (stop words are language and domain specific)
5. Remove all "white space"
6. Apply stemming techniques to what remains

What does all this mean?

A sentence that starts like this[6]:

Text Mining (or Text Analytics) applies analytic tools to learn from collections of text data, like social media, books, newspapers, emails, etc.

Ends up like this:

text min text analyt appli analyt tool learn collect text data like social media book newspap email etc

A few definitions[2]

TF Term Frequency, which measures how frequently a term occurs in a document.

$$\mathrm{tf}(t,d) = \frac{\text{Number of time the term } t \text{ appears in the document}}{\text{Total number of terms in the document} d}$$

IDF Inverse Document Frequency, which measures how important a term is (whether the term is common or rare across all documents).

$$\mathrm{IDF}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- D : the corpus, a collection of documents
- N : total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t,d) \neq 0$).

TF and IDF for sample string.

The terms:

```
 [1] "analyt"  "appli"   "book"    "collect" "data"    "email"   "etc"
 [8] "learn"   "like"    "media"   "mine"    "newspap" "social"  "text"
[15] "tool"
```

The frequency of each term:

```
 [1] 2 1 1 1 1 1 1 1 1 1 1 1 1 3 1
```

IDF is not a real useful metric with only one document:

```
1 weightTfIdf(TermDocumentMatrix(corp1))$v
2 named numeric(0)
```

(See Misc. slides for code.)

Whats happening in the beginning?

We gather up a predefined set of documents, save them locally, and create a term frequency object:

```
1 tempFile <- file.path(tempdir(), "corpus.dat")
2 corpusCase <- 1
3 corpus <- getCorpus(corpusCase, useSaved=TRUE, saveFile
      =tempFile)
4 processed <- normalizeText(corpus)
5 dumpObject(processed)
```

(From the attached file.)

Afterwards we look at the corpus:

```
[1] "   -- Dumping the object: processed (of type: list, class: VCorpus)"
[2] "   -- Dumping the object: processed (of type: list, class: Corpus)"
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2000
[1] "   -- Dumping the object:
     head(Frequencies[order(Frequencies, decreasing = T)], 5)
     (of type: double, class: numeric)"
   film    movi     one    like charact
  11109    6857    5759    3998    3855
[1] "   -- Dumping the object:
    head(DocFrequencies[order(DocFrequencies, decreasing = T)], 5)
    (of type: double, class: numeric)"
   film     one    movi    like charact
   1797    1763    1642    1538    1431
```

We now know the most common terms across the 2,000 documents in the corpus.

Gathering a few corpus statistics.

It is easy to think about how terms and documents create a 2-dimensional array.

```
[1] "   -- Dumping the object: moreThanOnce (of type: integer, class: integer)"
[1] 9748
[1] "   -- Dumping the object: total (of type: integer, class: integer)"
[1] 30585
[1] "   -- Dumping the object: prop (of type: double, class: numeric)"
[1] 0.3187183
[1] "   -- Dumping the object: ncol(SparseRemoved) (of type: integer, class: integer)"
[1] 202
[1] "   -- Dumping the object: sum(rowSums(as.matrix(SparseRemoved)) == 0)
    (of type: integer, class: integer)"
[1] 0
[1] "   -- Dumping the object: colnames(SparseRemoved) (of type: character, class: character)"
  [1] "act"       "action"    "actor"     "actual"    "almost"    "along"
```

Columns that have only one entry are assumed to not be too interesting.

Create a dataframe with all the data

```
quality <- c(rep(1,1000),rep(0,1000))
lengths <- colSums(as.matrix(TermDocumentMatrix(
    processed)))
DF <- as.data.frame(cbind(quality, lengths,
    SparseRemoved))
```

DF is a dataframe with review lengths and positive and negative reviews. The reviews are coded as 1 or 0 to support logistic regression (logit).

How well do *knn* classifiers do? (1 of 2)

The code:

```
1 Class3n <- knn(TrainDF[,-1], TrainDF[,-1], TrainDF[,1],
      k = 3)
2 Class5n <- knn(TrainDF[,-1], TrainDF[,-1], TrainDF[,1],
      k = 5)
3 dumpObject(confusionMatrix(Class3n, as.factor(TrainDF$
    quality)))
4 dumpObject(confusionMatrix(Class5n, as.factor(TrainDF$
    quality)))
```

How well do *knn* classifiers do? (2 of 2)

The results:

```
[1] "   -- Dumping the object: confusionMatrix(Class3n,
     as.factor(TrainDF$quality)) (of type: list,
     class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 358 126
         1 134 382

               Accuracy : 0.74
...
[1] "   -- Dumping the object: confusionMatrix(Class5n,
     as.factor(TrainDF$quality)) (of type: list,
     class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 336 162
         1 156 346

               Accuracy : 0.682
```

How well will a naïve Bayes classifier do? (1 of 2)

The code:

```
model <- naiveBayes(TrainDF[-1], as.factor(TrainDF
    [[1]]))
classifNB <- predict(model, TrainDF[,-1])
dumpObject(confusionMatrix(as.factor(TrainDF$quality),
    classifNB))
classifNB <- predict(model, TestDF[,-1])
dumpObject(confusionMatrix(as.factor(TestDF$quality),
    classifNB))
```

How well will a naïve Bayes classifier do? (2 of 2)

The results:

```
[1] "   -- Dumping the object: confusionMatrix(
           as.factor(TrainDF$quality), classifNB)
           (of type: list, class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 353 139
         1  74 434

               Accuracy : 0.787
...
[1] "   -- Dumping the object: confusionMatrix(
           as.factor(TestDF$quality), classifNB)
            (of type: list, class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 335 173
         1 120 372

               Accuracy : 0.707
```

How well will logistic regression (logit) do? (1 of 2)

The code:

```
model <- glm(quality~ lengths, family = binomial)
dumpObject(summary(model))
...
classif <- Prob1
classif[classif >0.5] = 1
classif[classif <=0.5] = 0
tbl <- table(classif, quality)
dumpObject(tbl)
dumpObject(cohen.kappa(tbl))
model2 <- glm(quality ~ ., family = binomial, data = TrainDF)
TrainDF$classif <- fitted.values(model2, type="response")
TrainDF$classif <- assignBinary(TrainDF$classif, 0.5)
dumpObject(confusionMatrix(TrainDF$quality, TrainDF$classif))
TestDF$classif <- predict(model2, TestDF, type = "response")
TestDF$classif[TestDF$classif >0.5] <- 1
TestDF$classif[TestDF$classif <=0.5] <- 0
dumpObject(confusionMatrix(TestDF$quality,TestDF$classif))
```

How well will logistic regression (logit) do? (2 of 2)

The results:

```
[1] "   -- Dumping the object: summary(model) (of type: list, class: summary.glm)"
glm(formula = quality ~ lengths, family = binomial)
...
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.6383373  0.1171536  -5.449 5.07e-08 ***
lengths      0.0018276  0.0003113   5.871 4.32e-09 ***
...
[1] "   -- Dumping the object: tbl (of type: integer, class: table)"
       quality
classif   0   1
      0 614 507
      1 386 493
...
[1] "   -- Dumping the object: confusionMatrix(TrainDF$quality, TrainDF$classif) (of type: list, class: co
Confusion Matrix and Statistics
          Reference
Prediction   0   1
         0 418  74
         1  69 439
               Accuracy : 0.857
...
[1] "   -- Dumping the object: confusionMatrix(TestDF$quality, TestDF$classif) (of type: list, class: conf
Confusion Matrix and Statistics
          Reference
Prediction   0   1
         0 377 131
         1 145 347
               Accuracy : 0.724
```

How well will a support vector machine (svm) do? (1 of 2)

> *"The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear."*
>
> *James, et al. [1]*

The code:

```
1  modelSVM <- svm (quality ~ ., data = TrainDF)
2  probSVMtrain <- predict(modelSVM, TrainDF[,-1])
3  classifSVMtrain <- probSVMtrain
4  classifSVMtrain <- assignBinary(classifSVMtrain, 0.5)
5  dumpObject(confusionMatrix(TrainDF$quality, classifSVMtrain))
6  probSVMtest <- predict(modelSVM, TestDF[,-1])
7  classifSVMtest <- probSVMtest
8  classifSVMtest[classifSVMtest>0.5] <- 1
9  classifSVMtest[classifSVMtest<=0.5] <- 0
10 dumpObject(confusionMatrix(TestDF$quality, classifSVMtest))
```

How well will a support vector machine (svm) do? (2 of 2)

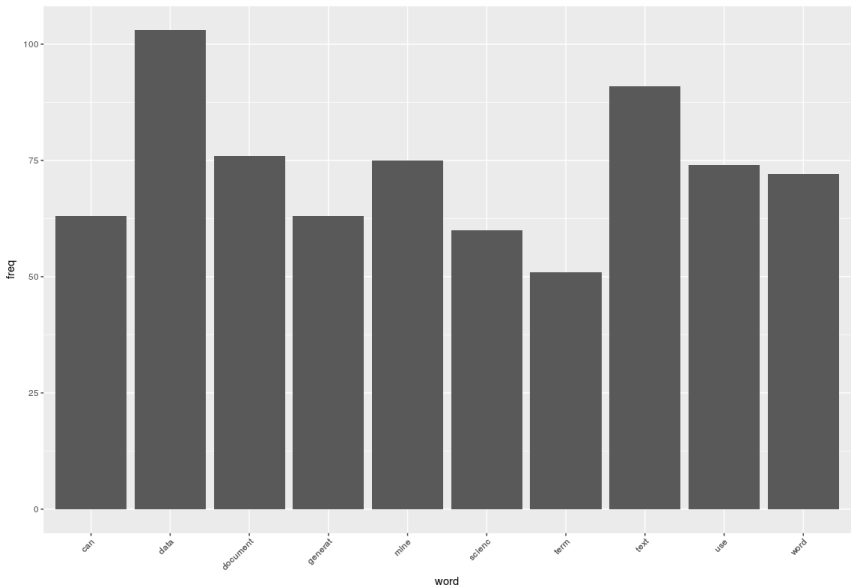
The results:

```
[1] "   -- Dumping the object: confusionMatrix(
        TrainDF$quality, classifSVMtrain)
        (of type: list, class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 449  43
         1  38 470

               Accuracy : 0.919
...
[1] "   -- Dumping the object: confusionMatrix(
        TestDF$quality, classifSVMtest)
        (of type: list, class: confusionMatrix)"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 378 130
         1 146 346

               Accuracy : 0.724
```

Looking at term frequency in a PDF.

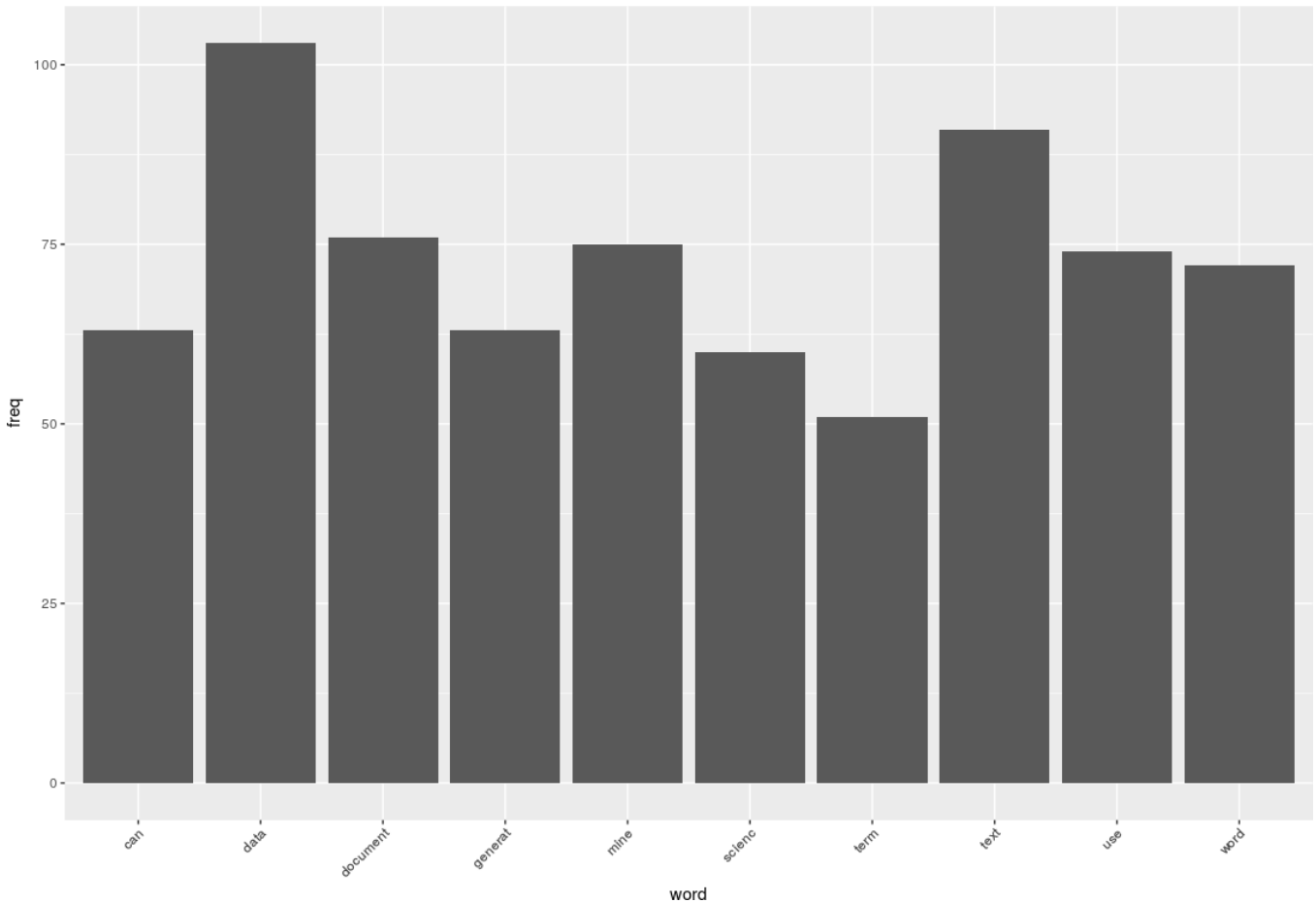
We will do a few things:

1. Read text directly from a PDF.
2. "Normalize" the text.
3. Look at the text in different ways.



Attached file.

Same image.



Attached file.

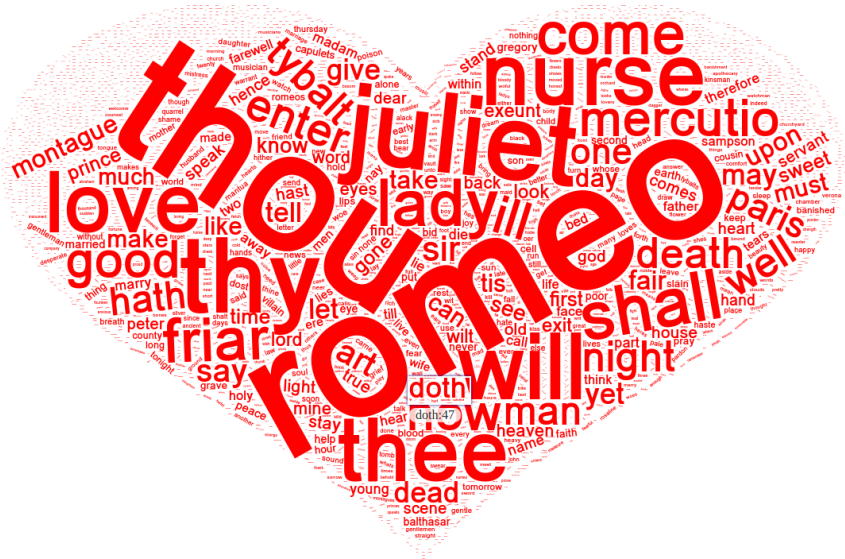
Look at text frequency as a B&W word cloud

Attached file.

Look at text frequency as a color word cloud

Attached file.

More colorful examples from Romeo and Juliet



Attached file.

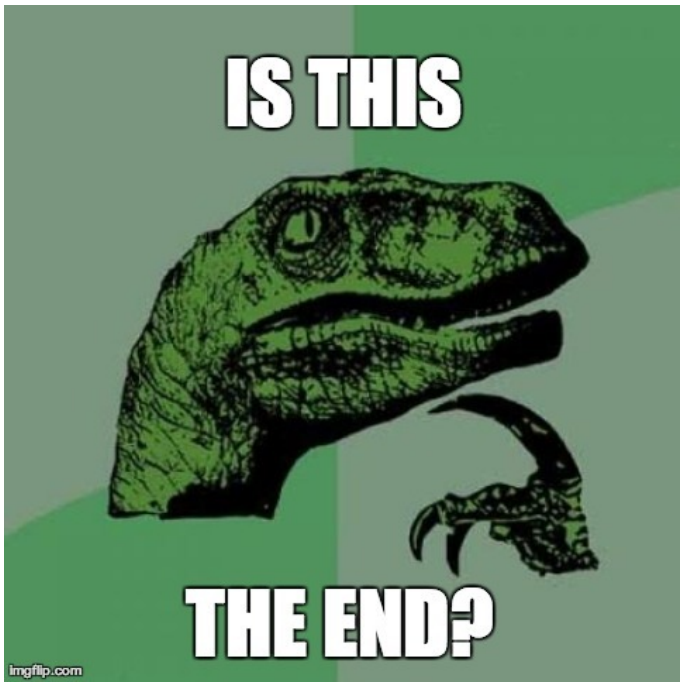
Q & A time.

Q: How do you catch a unique rabbit?
A: Unique up on it!

Q: How do you catch a tame rabbit?
A: The tame way!

What have we covered?

- Compared and contrasted numerical and textual data analysis
- Provided a few numerical definitions (TF, IDF) that are fundamental to textual analysis
- Applied different textual analysis tools and techniques (*knn*, naïve Bayes, logit, and support vector machine)
- Looked at different graphical ways textual data could be displayed



Next: Serial vs. parallel processing

References (1 of 2)

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, An Introduction to Statistical Learning, vol. 6, Springer, 2013.

[2] TF-IDF Staff, What does tf-idf mean?, http://www.tfidf.com/, 2017.

[3] Wikipedia Staff, Logistic function, https://en.wikipedia.org/wiki/Logistic_function, 2017.

[4] _____, Naive Bayes classifier, https://en.wikipedia.org/wiki/Naive_Bayes_classifier, 2017.

[5] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Introduction to Data Mining, Pearson Education India, 2006.

References (2 of 2)

[6] G Williams, Hands-On Data Science with R: Text Mining, 2016.

Files of interest

1. Revised textual analysis script
2. Silly textual analysis script
3. PDF file used with silly textual analysis script
4. R library script file
5. Other ways to display word clouds

R script to create sample text "normalization"

```
1  library(NLP)
2
3  library(tm)
4
5  a <- "Text Mining (or Text Analytics) applies analytic
6  tools to learn from collections of text data, like
7  social media, books, newspapers, emails, etc."
8
9  corp1<- tm_map(Corpus(VectorSource(removeWords(
10 removePunctuation(tolower(a)), stopwords("english")))),
11 stemDocument)
12
13 corp1[[1]]$content
14
15 TermDocumentMatrix(corp1)$dimnames$Terms
16
17 TermDocumentMatrix(corp1)$v
```

Logistic function[3]

A logistic function or logistic curve is a common "S" shape (sigmoid curve), with equation:
$f(x) = \frac{L}{1+e^{-k(x-x_0)}}$
Where:

- e = the natural logarithm base
- x_0 = the x-value of the sigmoid's midpoint,
- L = the curve's maximum value, and
- k = the steepness of the curve

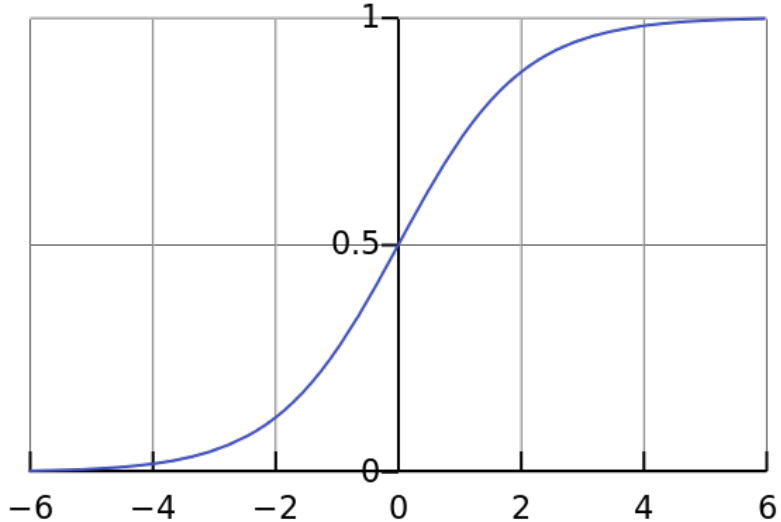


Image from [3].

Logistic regression[1]

Linear regression is:

$$p(X) = \beta_0 + \beta_1 X$$

Incorporating the logistic regression:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

"After a bit of manipulation":

$$\frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 X$$

Taking the logarithm of both sides:

$$\log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X$$

The left-hand side is called the *log-odds* or **logit**.
logit for both minimum and maximum X values must be computed and the larger one assumed most likely.

Naïve Bayes[5]

"Naïve Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set."

W. Staff [4]

Term	Expression
Joint probability	$p(x, y)$
Independent probability	$P(X, Y) = P(X) * P(Y)$
Conditional probability	$P(X \mid Y) = \frac{P(X,Y)}{P(X)}$
Bayes theorem	$P(X \mid Y) = \frac{P(X\|Y)*P(Y)}{P(X)}$
Prior probability	$P(X)$
Posterior probability	$P(X \mid Y)$