A Primer on Text Analytics

From the Perspective of a Statistics Graduate Student

Bradley Turnbull

Statistical Learning Group
North Carolina State University

March 27, 2015

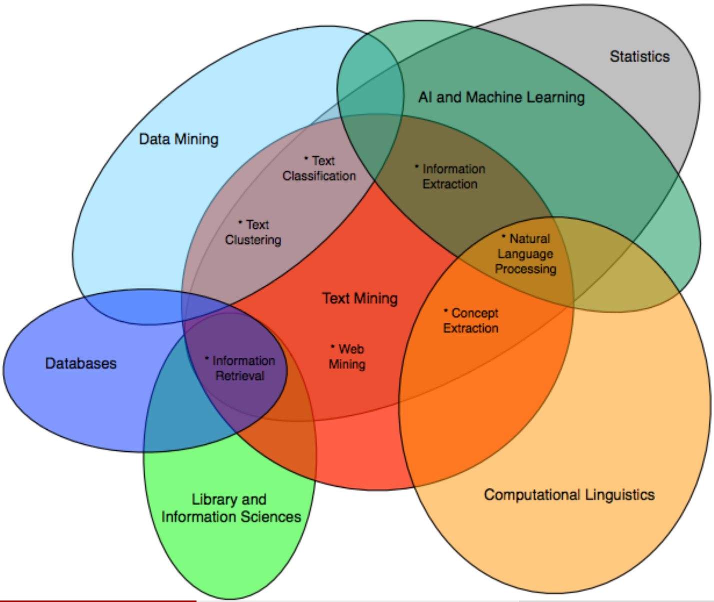
What is Text Analytics?

- A set of linguistic, analytic, and predictive techniques to extract structure and meaning from unstructured documents
- The objective of Text Mining is to exploit information contained in textual documents in various ways, including... discovery of patterns and trends in data, associations among entities, predictive rules, etc. (Grobelnik et al., 2001)

What is Text Analytics?

- A set of linguistic, analytic, and predictive techniques to extract structure and meaning from unstructured documents
- The objective of Text Mining is to exploit information contained in textual documents in various ways, including... discovery of patterns and trends in data, associations among entities, predictive rules, etc. (Grobelnik et al., 2001)
- If we get to the root of it...
  - Data analytics where some or all of the data is in an unstructured format

What is Text Analytics?



Why Text Analytics?

- Most of the "World's" data is in an unstructured format (80%)
  - Email and Messages
  - Webpages
  - Social Media (Twitter, Facebook, Blogs)
  - Surveys, feedback forms
  - Scientific literature, books, and legal documents

Why Text Analytics?

- Most of the "World's" data is in an unstructured format (80%)
  - Email and Messages
  - Webpages
  - Social Media (Twitter, Facebook, Blogs)
  - Surveys, feedback forms
  - Scientific literature, books, and legal documents
- Data is information, NOT just numbers in structured fields!

Where is it Applied?

- Intelligence and Law Enforcement
- Life Sciences and Clinical Medicine
- Social Media Analysis and Contextual Advertising
- Competitive Intelligence
- Product Management and Marketing
- Public Administration and Policy
- Recruiting

Major Challenges

- Natural human language can be ambiguous, subtle, and full of nuance

Major Challenges

- Natural human language can be ambiguous, subtle, and full of nuance
  - different words with the same meaning (Synonymy)
  - same word with different meanings (Homonomy)

Major Challenges

- Natural human language can be ambiguous, subtle, and full of nuance
  - different words with the same meaning (Synonymy)
  - same word with different meanings (Homonomy)
- Context is needed to clarify

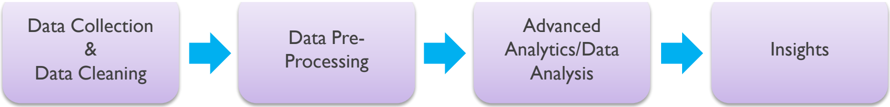
Major Challenges

- Natural human language can be ambiguous, subtle, and full of nuance
  - different words with the same meaning (Synonymy)
  - same word with different meanings (Homonomy)
- Context is needed to clarify
- Natural Language Processing – an entire lecture (or even course) unto itself

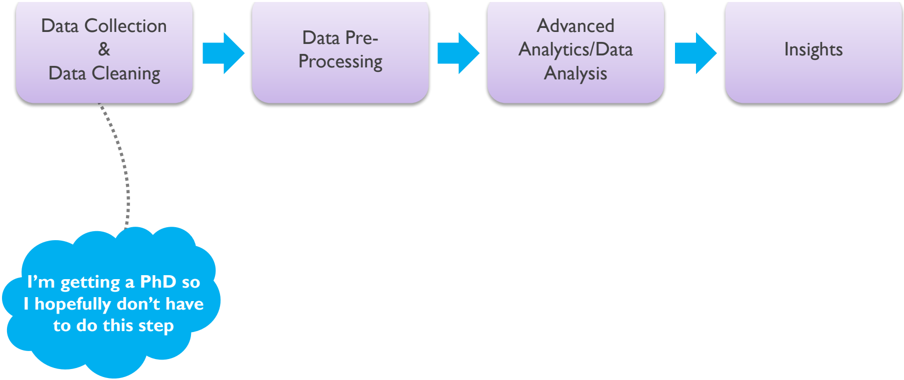
I Need to Read that Again...

- Juvenile Court to Try Shooting Defendant
- Kids Make Nutritious Snacks
- Local High School Dropouts Cut in Half
- Obesity Study Looks for Larger Test Group
- Red Tape Holds up New Bridges

The “Corporate” Flow Chart

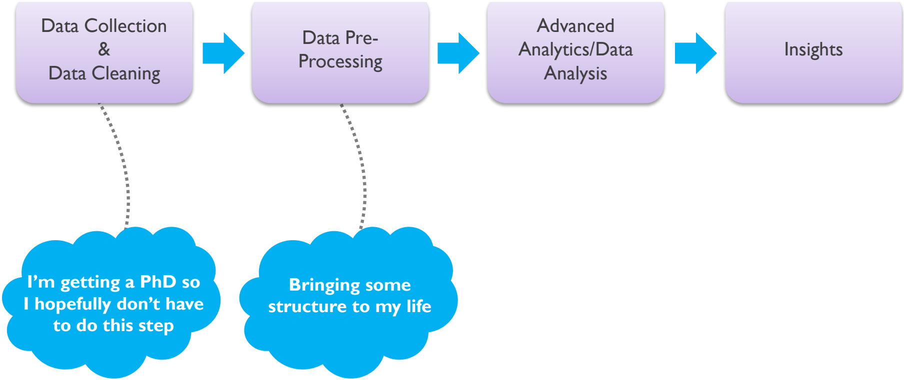


My Thoughts...

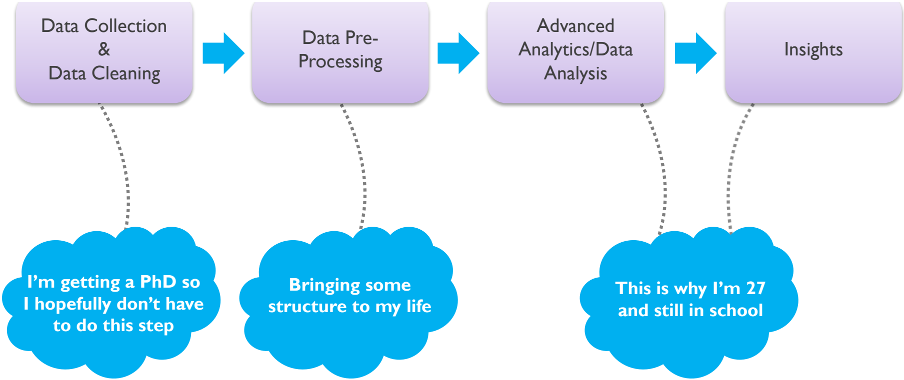


Data Collection & Data Cleaning
Data Pre-Processing
Advanced Analytics/Data Analysis
Insights
I'm getting a PhD so I hopefully don't have to do this step

My Thoughts...


Data Collection & Data Cleaning
Data Pre-Processing
Advanced Analytics/Data Analysis
Insights
I'm getting a PhD so I hopefully don't have to do this step
Bringing some structure to my life

My Thoughts...


Data Collection
&
Data Cleaning
Data Pre-Processing
Advanced Analytics/Data Analysis
Insights
I'm getting a PhD so I hopefully don't have to do this step
Bringing some structure to my life
This is why I'm 27 and still in school

Some "Free" Open Source Pre-Processing Software Tools

- R packages: "tm", "openNLP", "Rweka", "SnowballC", "koRpus", "RTextTools", "Rstem", "wordnet", "qdap"
- Java Software: WEKA, Carrot2
- Python Software: NLTK, Gensim, CLiPS Pattern
- Even More Options: GATE, KNIME, RapidMiner

Some "Free" Open Source Pre-Processing Software Tools

- R packages: "tm", "openNLP", "Rweka", "SnowballC", "koRpus", "RTextTools", "Rstem", "wordnet", "qdap"
- Java Software: WEKA, Carrot2
- Python Software: NLTK, Gensim, CLiPS Pattern
- Even More Options: GATE, KNIME, RapidMiner
- We will of course focus on R implementation

Let's Make Some Data

```
library(tm)
mySents <- c("Why didn't the Seahawks run the ball from the
              one yard line?!",
          "The Steelers ran the ball from the 2 yard line,
           but Jerome Bettis fumbled.")

myCorp <- Corpus( VectorSource(mySents) )
myCorp[[1]]
  <<PlainTextDocument>>
  Why didn't the Seahawks run the ball from the one yard
  line?!
myCorp[[2]]
  <<PlainTextDocument>>
  The Steelers ran the ball from the 2 yard line,
  but Jerome Bettis fumbled.
```

Let's Make Some Data

```
library(tm)
mySents <- c("Why didn't the Seahawks run the ball from the
              one yard line?!",
          "The Steelers ran the ball from the 2 yard line,
           but Jerome Bettis fumbled.")

myCorp <- Corpus( VectorSource(mySents) )
myCorp[[1]]
  <<PlainTextDocument>>
  Why didn't the Seahawks run the ball from the one yard
  line?!
myCorp[[2]]
  <<PlainTextDocument>>
  The Steelers ran the ball from the 2 yard line,
  but Jerome Bettis fumbled.
```

If data from csv file:

```
myTable <- read.table("somefile.csv",sep=",")
myCorp <- Corpus( DataframeSource(myTable) )
```

Major Preprocessing Steps

1. Tokenization and Cleaning

2. Dimension Reduction

3. Frequencies

Data “Cleaning” Steps

1. Tokenization - convert streams of characters into “words”
   - Main clue in English is white space
   - Special characters can make things difficult: Dr. O’Malley
2. Text Normalization - make all lowercase

Text Normalization in R

The `tm_map` function applies the "tm" package functions across all documents in the corpus, similar to `lapply`

```
myCorp <- tm_map(myCorp, tolower)
myCorp[[1]]; myCorp[[2]]
  [1] "why didn't the seahawks run the ball from the one yard line?
      !"
  [1] "the steelers ran the ball from the 2 yard line, but jerome
      bettis fumbled."
```

Data "Cleaning" Steps

1. Tokenization - convert streams of characters into "words"
   - Main clue in English is white space
   - Special characters can make things difficult: Dr. O'Malley
2. Text Normalization - make all lowercase
3. Spell Checking
   - Dictionary look up tables
   - Bayesian spell checker:

$$\arg\max_{c \in \mathscr{C}} \mathsf{Pr}(w \mid c)\, \mathsf{Pr}(c)$$

   Edit distance - deletion, transposition (swap), alteration, insertion

Data "Cleaning" Steps

1. Tokenization - convert streams of characters into "words"
   - Main clue in English is white space
   - Special characters can make things difficult: Dr. O'Malley
2. Text Normalization - make all lowercase
3. Spell Checking
   - Dictionary look up tables
   - Bayesian spell checker:

$$\arg\max_{c \in \mathscr{C}} \mathsf{Pr}(w \mid c)\, \mathsf{Pr}(c)$$

   Edit distance - deletion, transposition (swap), alteration, insertion

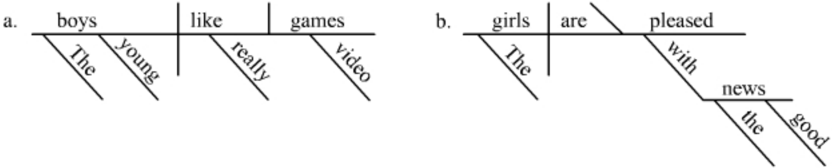
4. Part of Speech Tagging

Part of Speech Tagging

- Remember diagraming sentences in grade/middle school?

Part of Speech Tagging

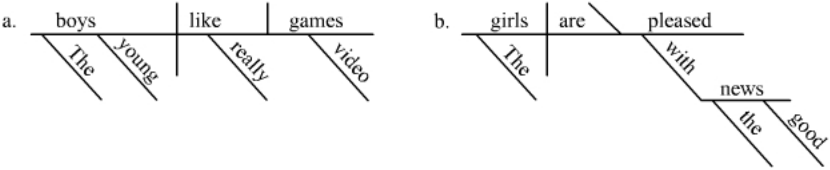
- Remember diagraming sentences in grade/middle school?



subject, predicate (verb), direct object, adjective, adverb,...

Part of Speech Tagging

- Remember diagraming sentences in grade/middle school?

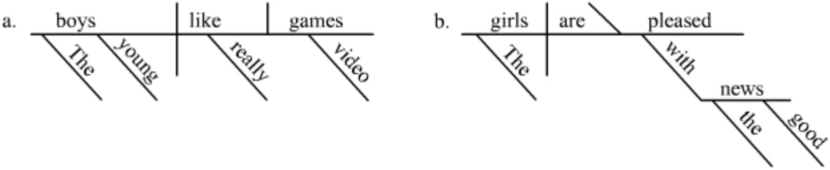


subject, predicate (verb), direct object, adjective, adverb,...

- How hard is it?

Part of Speech Tagging

- Remember diagraming sentences in grade/middle school?

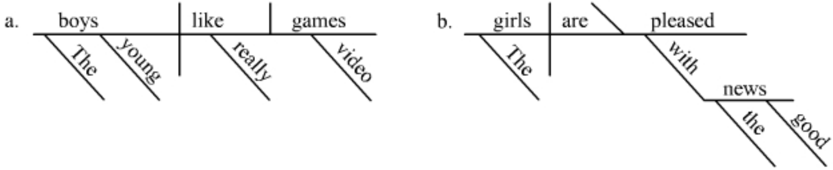


subject, predicate (verb), direct object, adjective, adverb,...

- How hard is it?
  - Approx. 89% of English words have only one part of speech
  - However, many common words in English are ambiguous

Part of Speech Tagging

- Remember diagraming sentences in grade/middle school?



  subject, predicate (verb), direct object, adjective, adverb,...

- How hard is it?
  - Approx. 89% of English words have only one part of speech
  - However, many common words in English are ambiguous
- Taggers can be rule-based, use probability models, or combination
  - Brill Tagger, from Eric Brill's 1995 PhD Thesis
- Process is similar to writing a compiler for programming language

Part of Speech Tagging in R

Use the "openNLP" package,

```
library("openNLP")

tagPOS <-  function(x){
  s <- as.String(x)
  word_token_annotator <- Maxent_Word_Token_Annotator()
  a2 <- Annotation(1L, "sentence", 1L, nchar(s))
  a2 <- annotate(s, word_token_annotator, a2)
  a3 <- annotate(s, Maxent_POS_Tag_Annotator(), a2)
  a3w <- a3[a3$type == "word"]
  POStags <- unlist(lapply(a3w$features, `[[`, "POS"))
  paste(sprintf("%s/%s", s[a3w], POStags), collapse = " ")
}

myCorp2 <- tm_map( myCorp, tagPOS )
myCorp2[[1]];myCorp2[[2]]
  [1] "why/WRB did/VBD n't/RB the/DT seahawks/NNS run/VBP the/DT
       ball/NN from/IN the/DT one/CD yard/NN line/NN ?/. !/."
  [1] "the/DT steelers/NNS ran/VBD the/DT ball/NN from/IN the/DT
       2/CD yard/NN line/NN ,/, but/CC jerome/NN bettis/NN
       fumbled/VBD ./."
```

Part of Speech Labels - Penn English Treebank

CC = Coordinating conjunction
CD = Cardinal number
DT = Determiner
EX = Existential there
FW = Foreign word
IN = Preposition or subordinating conjunction
JJ = Adjective
JJR = Adjective, comparative
JJS = Adjective, superlative
LS = List item marker
MD = Modal
NN = Noun, singular or mass
NNS = Noun, plural
NNP = Proper noun, singular
NNPS = Proper noun, plural
PDT = Predeterminer
POS = Possessive ending
PRP = Personal pronoun
PRP$ = Possessive pronoun
RB = Adverb
RBR = Adverb, comparative
RBS = Adverb, superlative
RP = Particle
SYM = Symbol
TO = to
UH = Interjection
VB = Verb, base form
VBD = Verb, past tense
VBG = Verb, gerund or present participle
VBN = Verb, past participle
VBP = Verb, non3rd person singular present
VBZ = Verb, 3rd person singular present
WDT = Whdeterminer
WP = Whpronoun
WP$ = Possessive Whpronoun
WRB = Whadverb

Dimension Reduction

Dimension Reduction

1. Remove Stop Words - words that rarely provide useful information
   - examples: a, and, of, the, to, etc.
2. Remove Punctuation

Remove Stop Words and Punctuation in R

```
myCorp[[1]];myCorp[[2]]
 [1] "Why didn't the Seahawks run the ball from the one yard
      line?!"
 [1] "The Steelers ran the ball from the 2 yard line,
      but Jerome Bettis fumbled."

myCorp <- tm_map( myCorp, removeWords, stopwords("english"))
myCorp <- tm_map( myCorp, removePunctuation)

myCorp[[1]]; myCorp[[2]]
[1] "seahawks run  ball   one yard line"
[1] "steelers ran  ball   2 yard line  jerome bettis fumbled"
```

Dimension Reduction

1. Remove Stop Words - words that rarely provide useful information
    - examples: a, and, of, the, to, etc.
2. Remove Punctuation
3. Stemming - unifies variations of the "same idea"
    - Remove plurals
    - Normalize verb tenses
    - Reduce word to most basic element

Dimension Reduction

1. Remove Stop Words - words that rarely provide useful information
   - examples: a, and, of, the, to, etc.
2. Remove Punctuation
3. Stemming - unifies variations of the "same idea"
   - Remove plurals
   - Normalize verb tenses
   - Reduce word to most basic element
   - walking, walks, walked $\rightarrow$ walk
   - apply, applications, reapplied $\rightarrow$ apply

Stemming

- General Approaches:
  - Lookup tables
  - Suffix-stripping
  - Lemmatisation - uses part of speech of word
  - Probability models (same idea as spell-checker)

Stemming

- General Approaches:
  - Lookup tables
  - Suffix-stripping
  - Lemmatisation - uses part of speech of word
  - Probability models (same idea as spell-checker)
- Two popular stemming algorithms
  - Lovins Stemmer (Julie Beth Lovins 1968)
  - Porter Stemmer (Martin Porter 1980)

Stemming

- General Approaches:
  - Lookup tables
  - Suffix-stripping
  - Lemmatisation - uses part of speech of word
  - Probability models (same idea as spell-checker)
- Two popular stemming algorithms
  - Lovins Stemmer (Julie Beth Lovins 1968)
  - Porter Stemmer (Martin Porter 1980)
- Both Lovins and Porter are rule-based suffix-stripping algorithms
  - Lovins - "3 steps"
  - Porter - "6 steps"

Stemming in R

```
stemMe <- Corpus(VectorSource(c(
                     "walking walks walked",
                     "apply applied applications reapplied",
                     "fishing fished fisher")))
#Porter
stemMe1 <- tm_map(stemMe, stemDocument, "english")
stemMe1[[1]];stemMe1[[2]];stemMe1[[3]]
 [1]  "walk walk walk"
 [1]  "appli appli applic reappli"
 [1]  "fish fish fisher"

#Lovins
stemMe2 <- tm_map(stemMe, RWeka::IteratedLovinsStemmer)
stemMe2[[1]];stemMe2[[2]];stemMe2[[3]]
 [1]  "walking walks walked"
 [1]  "apply applied applications reappl"
 [1]  "fishing fished f"
```

Dimension Reduction

1. Remove Stop Words - words that rarely provide useful information
   - examples: a, and, of, the, to, etc.
2. Remove Punctuation
3. Stemming - unifies variations of the “same idea”
   - Remove plurals
   - Normalize verb tenses
   - Reduce word to most basic element
   - walking, walks, walked $\rightarrow$ walk
   - apply, applications, reapplied $\rightarrow$ apply
4. Apply Thesauri - synonyms
   - Can be a very involved process
   - Subject specific thesauri are best (may have to build your own)

Dimension Reduction

1. Remove Stop Words - words that rarely provide useful information
   - examples: a, and, of, the, to, etc.
2. Remove Punctuation
3. Stemming - unifies variations of the "same idea"
   - Remove plurals
   - Normalize verb tenses
   - Reduce word to most basic element
   - walking, walks, walked $\rightarrow$ walk
   - apply, applications, reapplied $\rightarrow$ apply
4. Apply Thesauri - synonyms
   - Can be a very involved process
   - Subject specific thesauri are best (may have to build your own)
   - WordNet - large robust "database" of the English language compiled by Princeton University Linguistics department (FREE)

Synonyms in R

```
replaceWords <- function(object, words, by ) {
   pattern <- paste(words, collapse = "|")
   gsub(pattern, by, object)
}

myCorp2 <- tm_map(myCorp, replaceWords,
             words=c("seahawks","steelers"), by="football team")
myCorp2[[1]]; myCorp2[[2]]
 [1] "   football team run  ball   one yard line"
 [1] "football team ran  ball   2 yard line  jerome bettis fumbled"

myCorp <- tm_map(myCorp, replaceWords, words=c("ran"), by="run")
myCorp[[1]]; myCorp[[2]]
 [1] "   seahawks run  ball   one yard line"
 [1] " steelers run  ball   2 yard line  jerome bettis fumbled"

library("wordnet")
synonyms("company", pos="NOUN")
 [1] "caller"   "companionship"    "company"   "fellowship"
 [5] "party"    "ship's company"   "society"   "troupe"
synonyms("data", pos="NOUN")
 [1] "data"   "information"
```

Synonyms in R

```
#Very robust
synonyms("run", pos="VERB")
 [1] "be given"            "black market"       "bleed"
 [4] "break away"          "bunk"               "campaign"
 [7] "carry"               "consort"            "course"
[10] "die hard"            "draw"               "endure"
[13] "escape"              "execute"            "extend"
[16] "feed"                "flow"               "fly the coop"
[19] "function"            "go"                 "guide"
[22] "head for the hills"  "hightail it"        "hunt"
[25] "hunt down"           "incline"            "ladder"
[28] "lam"                 "lead"               "lean"
[31] "melt"                "melt down"          "move"
[34] "operate"             "pass"               "persist"
[37] "play"                "ply"                "prevail"
[40] "race"                "range"              "run"
[43] "run away"            "run for"            "scarper"
[46] "scat"                "take to the woods"  "tend"
[49] "track down"          "turn tail"          "unravel"
[52] "work"
```

Frequencies

Frequencies

- Characterizing the subject matter of documents by "counting" terms/words in and across documents

Frequencies

- Characterizing the subject matter of documents by "counting" terms/words in and across documents
- Going back to the "streets"

Frequencies

- Term Frequency (TF) - Number times term occurs in a document

Frequencies

- Term Frequency (TF) - Number times term occurs in a document
  - Assumes if term occurs more often, it measures something important
  - 2 times as many occurrences $\Rightarrow$ 2 times as important,
    common fix - use log transform, $\log(1 + \text{TF})$

Frequencies

- Term Frequency (TF) - Number times term occurs in a document
  - Assumes if term occurs more often, it measures something important
  - 2 times as many occurrences $\Rightarrow$ 2 times as important, common fix - use log transform, $\log(1 + \text{TF})$
- Document Frequency (DF) - Number of documents term occurs in

Frequencies

- Term Frequency (TF) - Number times term occurs in a document
  - Assumes if term occurs more often, it measures something important
  - 2 times as many occurrences $\Rightarrow$ 2 times as important, common fix - use log transform, $\log(1 + \text{TF})$
- Document Frequency (DF) - Number of documents term occurs in
  - Assumes terms that occur in fewer documents are more specified to a document and more descriptive of the content in that document - i.e. rarity $\Rightarrow$ more important and descriptive
  - Terms that occur in a lot of documents are common words, not as descriptive
  - Is this true? - may just reflect synonym variations, regional differences, or personal style

Frequencies

- Inverse Document Frequency (IDF)
  - DF - smaller is better, invert so "bigger" is better
  - Often too severe if we simply invert

Frequencies

- Inverse Document Frequency (IDF)
  - DF - smaller is better, invert so “bigger” is better
  - Often too severe if we simply invert
  - $\text{IDF} = \log(1 + 1/\text{DF})$

Frequencies

- Inverse Document Frequency (IDF)
  - DF - smaller is better, invert so “bigger” is better
  - Often too severe if we simply invert
  - $\text{IDF} = \log(1 + 1/\text{DF})$
- Term Frequency Inverse Document Frequency (TF-IDF)

Frequencies

- Inverse Document Frequency (IDF)
  - DF - smaller is better, invert so "bigger" is better
  - Often too severe if we simply invert
  - IDF $= \log(1 + 1/\text{DF})$
- Term Frequency Inverse Document Frequency (TF-IDF)
  - TF-IDF $=$ TF $\times$ IDF
  - Higher frequency of terms that are rare indicate a very important concept
  - Often standardize TF for each document
    TF/(Total Number of Terms in Doc)

Multi-Word Frequencies: N-Grams

- Count combinations of adjacent words
  - 2-grams (bigrams, digrams) - “vice president”, “not happy”, “statistics department”
  - 3-grams (trigrams) - “central intelligence agency”
  - 4-grams - “united states of america”, “laboratory for analytic sciences”

Multi-Word Frequencies: N-Grams

- Count combinations of adjacent words
  - 2-grams (bigrams, digrams) - "vice president", "not happy", "statistics department"
  - 3-grams (trigrams) - "central intelligence agency"
  - 4-grams - "united states of america", "laboratory for analytic sciences"
- Can simply enumerate all n-grams and then keep those which meet a frequency threshold
- Sophisticated methods use probability models and/or allow gaps between words

Document Term Matrix

Document Term Matrix

- The structured design matrix we've been yearning for
  - rows - documents
  - columns - words/terms (and n-grams)
- cells populated with TF, TF-IDF, or some custom variation

Document Term Matrix

- The structured design matrix we've been yearning for
  - rows - documents
  - columns - words/terms (and n-grams)
- cells populated with TF, TF-IDF, or some custom variation
- often large and sparse
- can concatenate with other structured data
- Term Document Matrix - simply the transpose

DTM in R

```
myCorp <- tm_map(myCorp, stripWhitespace)
myCorp <- tm_map(myCorp, PlainTextDocument)

dtm <- DocumentTermMatrix(myCorp)

inspect(dtm)
  <<DocumentTermMatrix (documents: 2, terms: 11)>>
  Non-/sparse entries: 15/7
  Sparsity           : 32%
  Maximal term length: 8
  Weighting          : term frequency (tf)
              Terms
  Docs    ball bettis fumbled jerome line one run seahawks
     1       1      0       0      0    1   1   1        1
     2       1      1       1      1    1   0   1        0
              Terms
  Docs    steelers two yard
     1           0   0    1
     2           1   1    1
```

DTM in R - Minimum Word Frequency

```
dtm2 <- DocumentTermMatrix(myCorp,
                  control=list(bounds=list(global=c(2,Inf))))

inspect(dtm2)
  <<DocumentTermMatrix (documents: 2, terms: 4)>>
  Non-/sparse entries: 8/0
  Sparsity           : 0%
  Maximal term length: 4
  Weighting          : term frequency (tf)

             Terms
  Docs   ball line run yard
     1      1    1   1    1
     2      1    1   1    1
```

DTM in R - Minimum Word Frequency

```
dtm2 <- DocumentTermMatrix(myCorp,
                    control=list(bounds=list(global=c(2,Inf))))

inspect(dtm2)
  <<DocumentTermMatrix (documents: 2, terms: 4)>>
  Non-/sparse entries: 8/0
  Sparsity           : 0%
  Maximal term length: 4
  Weighting          : term frequency (tf)

              Terms
  Docs   ball line run yard
     1      1    1   1    1
     2      1    1   1    1

#Another option
#Max 50% sparsity
dtm3 <-  removeSparseTerms( dtm, 0.5 )
dim(dtm3)
  [1] 2 4
```

Weighted DTM in R

```
wDTM <- weightTfIdf(dtm)

inspect(wDTM)
  <<DocumentTermMatrix (documents: 2, terms: 11)>>
  Non-/sparse entries: 7/15
  Sparsity           : 68%
  Maximal term length: 8
  Weighting          : term frequency - inverse document frequency
      (normalized) (tf-idf)

           Terms
  Docs   ball    bettis   fumbled    jerome line       one run
     1      0 0.0000000 0.0000000 0.0000000    0 0.1666667   0
     2      0 0.1111111 0.1111111 0.1111111    0 0.0000000   0
           Terms
  Docs    seahawks  steelers       two yard
     1   0.1666667 0.0000000 0.0000000    0
     2   0.0000000 0.1111111 0.1111111    0
```

N-Grams in R

```
biGramToken <- function(x){
   RWeka::NGramTokenizer(x, RWeka::Weka_control(min=1,max=2))
}
dtm.bigram <- DocumentTermMatrix(myCorp,
                   control=list(tokenize=biGramToken))
inspect(dtm.bigram)
  <<DocumentTermMatrix (documents: 2, terms: 22)>>
  Non-/sparse entries: 28/16
  Sparsity           : 36%
  Maximal term length: 14
  Weighting          : term frequency (tf)
              Terms
  Docs  ball ball one ball two bettis bettis fumbled fumbled jerome
    1     1         1        0      0              0       0      0
    2     1         0        1      1              1       1      1
  Docs  jerome bettis line line jerome one one yard run run ball
    1                0    1          0   1        1   1        1
    2                1    1          1   0        0   1        1
  Docs  seahawks seahawks run steelers steelers run two two yard
    1          1            1        0            0   0        0
    2          0            0        1            1   1        1
  Docs  yard  yard line
    1      1          1
    2      1          1
```

Training and Test Data

```
#Training Data
spam.train <- Corpus( VectorSource(
         c("i am spam", "buy this item", "party this weekend",
           "want to get coffee", "best product deal ever",
           "get outside this weekend", "beach trip details",
           "sale of the century", "buy buy buy",
           "bovine steroid pills") ))

#Test Data
spam.test <- Corpus( VectorSource(
         c("buy more of this item on sale",
           "beach trip room reservation",
           "buy it all", "deal of the century") ))
```

Training and Test Data

```
dtm.train <- DocumentTermMatrix(spam.train,
                                control=list(stopwords=T))

Terms(dtm.train)
  [1] "beach"   "best"    "bovine"  "buy"     "century" "coffee"
  [7] "deal"    "details" "ever"    "get"     "item"    "outside"
 [13] "party"   "pills"   "product" "sale"    "spam"    "steroid"
 [19] "trip"    "want"    "weekend"
```

Training and Test Data

```
dtm.test <- DocumentTermMatrix(spam.test,
                   control=list(dictionary=Terms(dtm.train)))
inspect(dtm.test)
  <<DocumentTermMatrix (documents: 4, terms: 21)>>
  Non-/sparse entries: 8/76
  Sparsity           : 90%
  Maximal term length: 7
  Weighting          : term frequency (tf)
      Terms
  Docs beach best bovine buy century coffee deal details ever get
     1     0    0      0   1       0      0    0       0    0   0
     2     1    0      0   0       0      0    0       0    0   0
     3     0    0      0   1       0      0    0       0    0   0
     4     0    0      0   0       1      0    1       0    0   0
  Docs item outside party pills product sale spam steroid trip want
     1    1       0     0     0       0    1    0       0    0    0
     2    0       0     0     0       0    0    0       0    1    0
     3    0       0     0     0       0    0    0       0    0    0
     4    0       0     0     0       0    0    0       0    0    0
  Docs weekend
     1       0
     2       0
     3       0
     4       0
```

Next Steps

- Topic Identification - clustering documents
- Document Classification - training classification model based on some response
- Sentiment Analysis - classification with positive and negative response variable

Next Steps

- Topic Identification - clustering documents
- Document Classification - training classification model based on some response
- Sentiment Analysis - classification with positive and negative response variable
    - Often performed by simply "summing" the weights of words from a sentiment dictionary

Next Steps

- Topic Identification - clustering documents
- Document Classification - training classification model based on some response
- Sentiment Analysis - classification with positive and negative response variable
  - Often performed by simply "summing" the weights of words from a sentiment dictionary
- Related Topics
  - Latent Dirichlet Allocation Models (LDA)
  - Temporal Theme Analysis
  - Latent Semantic Analysis