PLSC 504: Analyzing Text: A Super-Simple Introduction

December 2, 2020

Text as Data: Goals

Humans:

- Good at: Meaning, subtlety (irony, sarcasm, subtle negation, etc.), context, tone, etc.
- Bad at: Doing things quickly and consistently.

Computers:

- Good at: Doing things quickly and consistently.
- Bad at: Meaning, subtlety (irony, sarcasm, subtle negation, etc.), context, tone, etc.

What Can Text Methods Do?

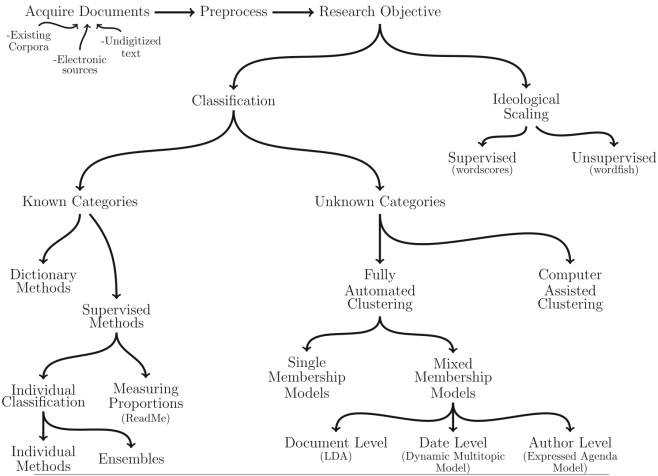
Grimmer's "haystack metaphor": Improved reading...

- Interpreting the meaning of a sentence or phrase ⇝ Analyzing a single straw of hay
  - Humans: amazing (e.g., the humanities)
  - Computers struggle
- Comparing, Organizing, and Classifying Text ⇝ Organizing a hay stack
  - Humans: terrible. Tiny active memories
  - Computers: amazing

What automated text methods don't do:

- Develop a comprehensive statistical model of language
- Replace the need to read
- Develop a single tool + evaluation for all tasks

Text as Data: Aims



Grimmer and Stewart's "Four Principles"

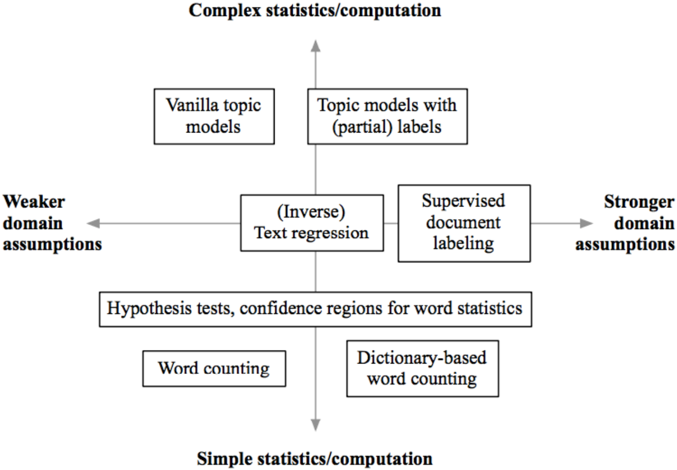
1. All quantitative models of language are wrong, but some are useful. 😕

2. Quantitative methods for text amplify resources and augment humans.

3. There is no globally best method for automated text analysis.

4. Validate, validate, validate.

Alternative Typology: O'Connor et al.


Complex statistics/computation
Vanilla topic models
Topic models with (partial) labels
Weaker domain assumptions
(Inverse) Text regression
Supervised document labeling
Stronger domain assumptions
Hypothesis tests, confidence regions for word statistics
Word counting
Dictionary-based word counting
Simple statistics/computation

Text as Data: Basic Terminology

- Word / Term: In NLP, a single collection of letters signifying some meaning(s).
- N-gram: A collection of two or more words, treated as a unit / term.
- Document: A natural collection of terms with a common theme or content.
- Tokenizing: Breaking up a document into words, N-grams, sentences, or other syntactic subunits.
- Corpus: A collection of documents.
- Stop Words: A group of extremely common words typically of little direct interest to the researcher (e.g., conjunctions).
- Normalization: The creation of *equivalence classes* of terms. Examples include:
    - Case folding: Harmonizing the case/capitalization of terms (e.g., "Work" and "work")
    - Stemming: Reducing words with common stems to those stems (e.g., "works" and "working" become "work*")
    - Lemmatization: Similar to stemming: Combining words with common roots but more diverse meanings (e.g., "democracy" and "democratization").

Notation

- N unique terms/words/tokens T_i in the corpus...
- ...indexed by $i = \{1, 2, ...N\}$
- J documents D_j, $j = \{1, 2, ...J\}$
- X_{ij} = the ith unique term in the jth document

Text Preprocessing: **One** Recipe

Preprocessing a la Grimmer:

- Remove capitalization, punctuation
- Tokenize / define N-grams
- Discard Word Order (Bag of Words Assumption)
- Discard stop words
- Create equivalence classes: stem, lemmatize, or synonym
- Discard less useful features $\rightsquigarrow$ depends on application
- Other reduction, specialization

Output: Count vector, each element counts occurrence of terms / stems

Capitalization and Punctuation

Capitalization / case-folding:

- Generally best removed (*Ferrari* and *ferrari* mean the same thing in English)
- Exceptions / potential pitfalls:
  - Proper nouns ("Mark Cuban" ≠ "mark" "cuban")
  - Acronyms ("CAT" ≠ "cat," etc.)
- Alternative: "truecasing"...

Punctuation:

- Periods, commas, colons, semicolons can usually go...
- Occasionally question marks and exclamation points are useful (e.g., sentiment analysis)
- **Order is important!** Don't remove punctuation prior to (say) tokenizing sentences...

Terms, Stems, and N-grams

Terms are the "lowest-level unit;" can be words, stems/roots, synonym groups, etc.

Stemming...

- Industry standard is the "snowball" stemmer...
- Details at `http://snowballstem.org/`

N-grams:

- Can be specified/user-defined ("Utah Jazz," "Orlando Magic," etc.)
- Useful for proper nouns, terms of art, etc.
- Can also be built from the corpus ("shingled")

Stop Words

- We *usually* want to remove them...
- Standard R stop words:

```
> stopwords("en")
 [1] "a"      "an"     "and"    "are"    "as"
 [6] "at"     "be"     "but"    "by"     "for"
[11] "if"     "in"     "into"   "is"     "it"
[16] "no"     "not"    "of"     "on"     "or"
[21] "such"   "that"   "the"    "their"  "then"
[26] "there"  "these"  "they"   "this"   "to"
[31] "was"    "will"   "with"
```

- Other lists are much longer (e.g. `https://github.com/stopwords-iso/stopwords-iso/`)
- Potential issues:
  - Proper nouns ("The Who," "That Was Then")
  - Stop word lists often have gendered pronouns (Monroe, Colaresi, and Quinn 2008)
  - Any word can be a stop word...

Term-Document and Document-Term Matrices

A term-document matrix has:

- N rows, corresponding to the N unique terms in the corpus
- J columns, corresponding to the J documents in the corpus
- Entries N_{ij} that represent the number of times term i appears in document j

A document-term matrix is a transposed term-document matrix.

Weighting (TF v. TF-IDF)

Term frequency:

$$N_{ij} = \text{The number of times term } i \text{ appears in document } j$$

Term frequency (normalized for document length):

$$TF_{ij} = \frac{N_{ij}}{\sum_{i=1}^{N} N_{ij}},$$

the fraction of all terms in D_j that are term T_i.

Inverse document frequency (normalized):

$$IDF_i = \log_2 \frac{J}{J_i}$$

where J_i is the number of documents in which T_i occurs.

TF-IDF$_{ij}$ is then simply $TF_{ij} \times IDF_i$

TF-IDF Examples

Three “documents”:

$$\begin{aligned} A &= \{\text{red}, \text{blue}, \text{red}\} \\ B &= \{\text{green}, \text{blue}, \text{orange}\} \\ C &= \{\text{yellow}, \text{blue}, \text{yellow}\} \end{aligned}$$

Example one:

- In document A “red” appears twice ($TF_{ij} = 2$), and
- “red” is two of the three total terms in that document (normed $TF_{ij} = 0.67$)
- “red” appears in only one of the three documents ($IDF_i = log_2[3/1] = 1.6$)
- The TF-IDF for “red” in document A is $0.67 \times 1.6 = 1.1$

Example two:

- In document C “blue” appears once ($TF_{ij} = 1$), and
- “blue” is one of the three total terms in that document (normed $TF_{ij} = 0.33$)
- “blue” appears in all three documents ($IDF_i = log_2[3/3] = 0$)
- The TF-IDF for “blue” in document C is $0.33 \times 0 = 0$

TF-IDF Intuition

In general:

- (Normalized) TF indicates the prevalence of a term in a document
- IDF reflects how common or rare the word is across documents
- IDF is thus a measure of the level of “informativeness” (or “document-specificity”) of a word
- TF-IDF is thus a measure of a term’s **“importance”** (in some respects)

Text Analysis in R: Toy (/ toe) Example

```
> # Raw text:
>
> Walter <- "You want a toe? I can get you a toe, believe me. There are ways, Dude.
You don’t wanna know about it, believe me."
>
> # Basic operations:
> #
> # Replace capitals (all-caps is "toupper"):
>
> tolower(Walter)
[1] "you want a toe? i can get you a toe, believe me. there are ways, dude.
you don’t wanna know about it, believe me."
>
> # Replace characters (ex: "a" with "A"):
>
> chartr("a","A",Walter)
[1] "You wAnt A toe? I cAn get you A toe, believe me. There Are wAys, Dude.
You don’t wAnnA know About it, believe me."
```

Basics, continued

```
> # Punctuation removal:
>
> removePunctuation(Walter)
[1] "You want a toe I can get you a toe believe me There are ways Dude
You dont wanna know about it believe me"
>
> # Remove words:
>
> removeWords(Walter, "toe")
[1] "You want a ? I can get you a , believe me. There are ways, Dude.
You don’t wanna know about it, believe me."
>
> # From a list:
>
> wordsGone<-c("toe","Dude","believe")
> removeWords(Walter, wordsGone)
[1] "You want a ? I can get you a ,  me. There are ways, . You don’t wanna know about it,  me."

>
> # Can also removeNumbers and stripWhitespace...
```

Tokenizing

```
> # Tokenize: Break into sentences:
>
> Walter.sent <- tokenize_sentences(Walter)
> Walter.sent
[[1]]
[1] "You want a toe?"
[2] "I can get you a toe, believe me."
[3] "There are ways, Dude."
[4] "You don’t wanna know about it, believe me."

> length(Walter.sent[[1]])
[1] 4
>
> # Tokenize II: Break into words:
>
> Walter.words <- tokenize_words(Walter)
> Walter.words
[[1]]
 [1] "you"     "want"    "a"       "toe"     "i"       "can"
 [7] "get"     "you"     "a"       "toe"     "believe" "me"
[13] "there"   "are"     "ways"    "dude"    "you"     "don’t"
[19] "wanna"   "know"    "about"   "it"      "believe" "me"

> length(Walter.words[[1]]) # total word count
[1] 24
```

Tokenize, continued

```
> # Tokenize III: Break sentences into words:
>
> Walter.sw <- tokenize_words(Walter.sent[[1]])
> Walter.sw
[[1]]
[1] "you"  "want" "a"    "toe"

[[2]]
[1] "i"       "can"     "get"     "you"     "a"       "toe"     "believe"
[8] "me"

[[3]]
[1] "there" "are"   "ways"  "dude"

[[4]]
[1] "you"     "don’t"   "wanna"   "know"    "about"   "it"      "believe"
[8] "me"
```

Counting Things

```
> # Count words per sentence:
>
> Walter.wordcount <- sapply(Walter.sw, length)
> Walter.wordcount
[1] 4 8 4 8
```

```
> # Term frequencies:
>
> termFreq(Walter, control=list(removePunctuation=TRUE))
  about     are believe     can    dont    dude     get    know   there
      1       1       2       1       1       1       1       1       1
    toe   wanna    want    ways     you
      2       1       1       1       3
attr(,"class")
[1] "term_frequency" "integer"
```

N-grams

```
> # N-grams: Basic N-grams of length 2:
>
> Walter.Ng2<-tokenize_ngrams(Walter,n=2)
> Walter.Ng2
[[1]]
 [1] "you want"    "want a"      "a toe"       "toe i"
 [5] "i can"       "can get"     "get you"     "you a"
 [9] "a toe"       "toe believe" "believe me"  "me there"
[13] "there are"   "are ways"    "ways dude"   "dude you"
[17] "you don’t"   "don’t wanna" "wanna know"  "know about"
[21] "about it"    "it believe"  "believe me"

> # Count of unique N-grams of length 2:
>
> table(Walter.Ng2)
Walter.Ng2
      a toe    about it    are ways  believe me     can get don’t wanna
          2           1           1           2           1           1
   dude you     get you       i can  it believe  know about    me there
          1           1           1           1           1           1
  there are toe believe       toe i  wanna know      want a   ways dude
          1           1           1           1           1           1
      you a   you don’t    you want
          1           1           1
```

Skip N-grams

```
> # Skip N-grams: length=4, skip=1:
>
> tokenize_skip_ngrams(Walter,n=4,k=1)
[[1]]
 [1] "you a i get"              "want toe can you"
 [3] "a i get a"                "toe can you toe"
 [5] "i get a believe"          "can you toe me"
 [7] "get a believe there"      "you toe me are"
 [9] "a believe there ways"     "toe me are dude"
[11] "believe there ways you"   "me are dude don’t"
[13] "there ways you wanna"     "are dude don’t know"
[15] "ways you wanna about"     "dude don’t know it"
[17] "you wanna about believe"  "don’t know it me"
[19] "you want a toe"           "want a toe i"
[21] "a toe i can"              "toe i can get"
[23] "i can get you"            "can get you a"
[25] "get you a toe"            "you a toe believe"
[27] "a toe believe me"         "toe believe me there"
[29] "believe me there are"     "me there are ways"
[31] "there are ways dude"      "are ways dude you"
[33] "ways dude you don’t"      "dude you don’t wanna"
[35] "you don’t wanna know"     "don’t wanna know about"
[37] "wanna know about it"      "know about it believe"
[39] "about it believe me"
```

Eliminating Stop-Words and Basic Stemming

```
> # Eliminate stop-words:
>
> stopwords("en")
 [1] "a"     "an"    "and"   "are"   "as"    "at"    "be"    "but"
 [9] "by"    "for"   "if"    "in"    "into"  "is"    "it"    "no"
[17] "not"   "of"    "on"    "or"    "such"  "that"  "the"   "their"
[25] "then"  "there" "these" "they"  "this"  "to"    "was"   "will"
[33] "with"

> removeWords(Walter,stopwords("en"))
[1] "You want  toe? I can get you  toe, believe me. There  ways, Dude.
You don’t wanna know about , believe me."


> # Basic stemming (uses the Snowball stemmer):
>
> stemDocument(Walter)
[1] "You want a toe? I can get you a toe, believ me. There are ways, Dude.
You don’t wanna know about it, believ me."
```

Creating a NLP Document

```
> # Create a basic document (NLP package):
>
> WS <- PlainTextDocument(Walter, author="Walter Sobchak",
+                          description="Get you a toe",
+                          language="en",
+                          origin="The Big Lebowski")
>
> str(WS)
List of 2
 $ content: chr "You want a toe? I can get you a toe, believe me. There are ways, Dude. You don’t wanna kn
 $ meta   :List of 7
  ..$ author       : chr "Walter Sobchak"
  ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-14 16:39:41"
  ..$ description  : chr "Get you a toe"
  ..$ heading      : chr(0)
  ..$ id           : chr(0)
  ..$ language     : chr "en"
  ..$ origin       : chr "The Big Lebowski"
  ..- attr(*, "class")= chr "TextDocumentMeta"
 - attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
```

A Basic Corpus (multiple documents)

```
> # Creating a (simple) corpus from sentences/words (NLP package):
>
> Walter.clean <- removePunctuation(Walter.sent[[1]])
> WSC<-Corpus(VectorSource(Walter.clean))
>
> inspect(WSC)
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 4

[1] You want a toe
[2] I can get you a toe believe me
[3] There are ways Dude
[4] You dont wanna know about it believe me
>
> str(WSC)
List of 4
 $ 1:List of 2
  ..$ content: chr "You want a toe"
  ..$ meta   :List of 7
  .. ..$ author       : chr(0)
  .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-14 18:09:21"
  .. ..$ description  : chr(0)
  .. ..$ heading      : chr(0)
  .. ..$ id           : chr "1"
  .. ..$ language     : chr "en"
  .. ..$ origin       : chr(0)
  .. ..- attr(*, "class")= chr "TextDocumentMeta"
  ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
 .
 .
 .
```

Term-Document Matrix

```
> # Term-Document Matrix:
>
> WS.TDM <- TermDocumentMatrix(WSC,control=list(tolower=TRUE,
+                                               stemming=TRUE))
>
> inspect(WS.TDM)
<<TermDocumentMatrix (terms: 14, documents: 4)>>
Non-/sparse entries: 18/38
Sparsity           : 68%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms    1 2 3 4
  about  0 0 0 1
  are    0 0 1 0
  believ 0 1 0 1
  can    0 1 0 0
  dont   0 0 0 1
  dude   0 0 1 0
  get    0 1 0 0
  know   0 0 0 1
  toe    1 1 0 0
  you    1 1 0 1
```

Document-Term Matrix

```
> # Document-Term Matrix:
>
> WS.DTM <- DocumentTermMatrix(WSC,control=list(tolower=TRUE,
+                                                stemming=TRUE))
>
> inspect(WS.DTM)
<<DocumentTermMatrix (documents: 4, terms: 14)>>
Non-/sparse entries: 18/38
Sparsity           : 68%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs about are believ can dont dude get know toe you
   1     0   0      0   0    0    0   0    0   1   1
   2     0   0      1   1    0    0   1    0   1   1
   3     0   1      0   0    0    1   0    0   0   0
   4     1   0      1   0    1    0   0    1   0   1

> as.matrix(WS.DTM)
    Terms
Docs about are believ can dont dude get know there toe wanna want way you
   1     0   0      0   0    0    0   0    0     0   1     0    1   0   1
   2     0   0      1   1    0    0   1    0     0   1     0    0   0   1
   3     0   1      0   0    0    1   0    0     1   0     0    0   1   0
   4     1   0      1   0    1    0   0    1     0   0     1    0   0   1
```

Associations

```
> # Associations:
>
> cor(as.matrix(WS.DTM))
       about   are believ   can  dont  dude   get  know there   toe wanna  want   way   you
about   1.00 -0.33   0.58 -0.33  1.00 -0.33 -0.33  1.00 -0.33 -0.58  1.00 -0.33 -0.33  0.33
are    -0.33  1.00  -0.58 -0.33 -0.33  1.00 -0.33 -0.33  1.00 -0.58 -0.33 -0.33  1.00 -1.00
believ  0.58 -0.58   1.00  0.58  0.58 -0.58  0.58  0.58 -0.58  0.00  0.58 -0.58 -0.58  0.58
can    -0.33 -0.33   0.58  1.00 -0.33 -0.33  1.00 -0.33 -0.33  0.58 -0.33 -0.33 -0.33  0.33
dont    1.00 -0.33   0.58 -0.33  1.00 -0.33 -0.33  1.00 -0.33 -0.58  1.00 -0.33 -0.33  0.33
dude   -0.33  1.00  -0.58 -0.33 -0.33  1.00 -0.33 -0.33  1.00 -0.58 -0.33 -0.33  1.00 -1.00
get    -0.33 -0.33   0.58  1.00 -0.33 -0.33  1.00 -0.33 -0.33  0.58 -0.33 -0.33 -0.33  0.33
know    1.00 -0.33   0.58 -0.33  1.00 -0.33 -0.33  1.00 -0.33 -0.58  1.00 -0.33 -0.33  0.33
there  -0.33  1.00  -0.58 -0.33 -0.33  1.00 -0.33 -0.33  1.00 -0.58 -0.33 -0.33  1.00 -1.00
toe    -0.58 -0.58   0.00  0.58 -0.58 -0.58  0.58 -0.58 -0.58  1.00 -0.58  0.58 -0.58  0.58
wanna   1.00 -0.33   0.58 -0.33  1.00 -0.33 -0.33  1.00 -0.33 -0.58  1.00 -0.33 -0.33  0.33
want   -0.33 -0.33  -0.58 -0.33 -0.33 -0.33 -0.33 -0.33 -0.33  0.58 -0.33  1.00 -0.33  0.33
way    -0.33  1.00  -0.58 -0.33 -0.33  1.00 -0.33 -0.33  1.00 -0.58 -0.33 -0.33  1.00 -1.00
you     0.33 -1.00   0.58  0.33  0.33 -1.00  0.33  0.33 -1.00  0.58  0.33  0.33 -1.00  1.00
>
> findAssocs(WS.TDM,"toe",0.3)
$toe
 can  get want  you
0.58 0.58 0.58 0.58
```

Example Two: The 2016 Presidential Debates

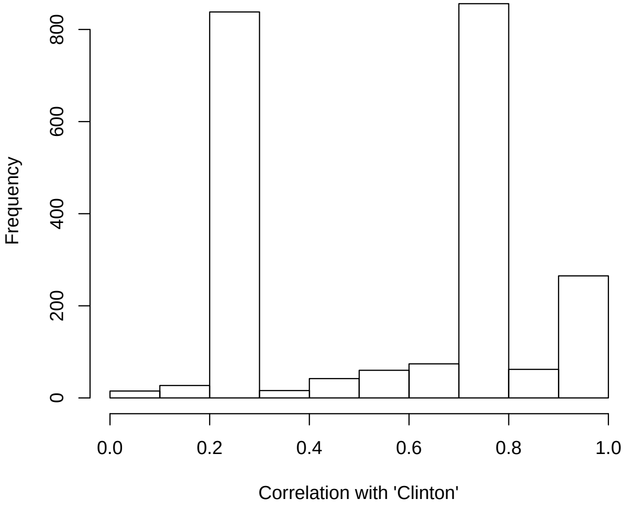
```
> Dfiles <- list.files(path="Data/Debates/",
+                     pattern="pdf")
> Dpdf<-readPDF(control = list(text = "-layout"))
>
> D16<-VCorpus(URISource(paste0("Data/Debates/",Dfiles)),
+             readerControl = list(reader = Dpdf))
>
> # Now clean that mess up while creating the TDM:
>
> D16.TDM <- TermDocumentMatrix(D16,
+             control=list(removePunctuation = TRUE,
+             stopwords=TRUE,tolower=TRUE,
+             stemming=TRUE,removeNumbers=FALSE))
> inspect(D16.TDM)
<<TermDocumentMatrix (terms: 2728, documents: 3)>>
Non-/sparse entries: 4810/3374
Sparsity           : 41%
Maximal term length: 21
Weighting          : term frequency (tf)
Sample             :
         Docs
Terms     Debate2016-1.pdf Debate2016-2.pdf Debate2016-3.pdf
  clinton              134               82              119
  countri               83               65               74
  get                   50               73               69
  it?                   97               64               44
  peopl                 72              101               93
  say                   61               57               67
  think                 84               55               71
  trump                151              111              141
  want                  54               88               95
  will                  65               74               92
```

Associations

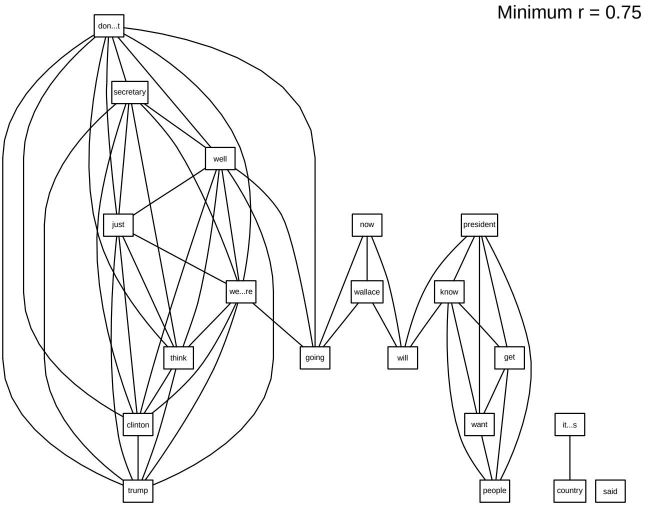
```
> # Associations:
>
> findAssocs(D16.TDM,"clinton",0.98)
$clinton
   attacks    benefit      build      built        buy    created      deals
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
  donald?s experience       iran     issues       jobs       lots     matter
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
 negotiate   prepared        say  secretary    segment      trump      world
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
   biggest      birth    company     defend       he?s       home      japan
      0.99       0.99       0.99       0.99       0.99       0.99       0.99
      just      nafta       next      wrong
      0.99       0.99       0.99       0.99

> findAssocs(D16.TDM,"trump",0.98)
$trump
   attacks    benefit      birth      build      built        buy    clinton
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
   created      deals     defend   donald?s experience       iran     issues
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
     japan       jobs       lots     matter      nafta  negotiate   prepared
      1.00       1.00       1.00       1.00       1.00       1.00       1.00
       say  secretary      world    biggest  countries       just    segment
      1.00       1.00       1.00       0.99       0.99       0.99       0.99
  they?ve      wrong    company    economy       home
      0.99       0.99       0.98       0.98       0.98
```

(Positive) Correlations with "clinton"



Term-Document Matrix Plot (using Rgraphviz)



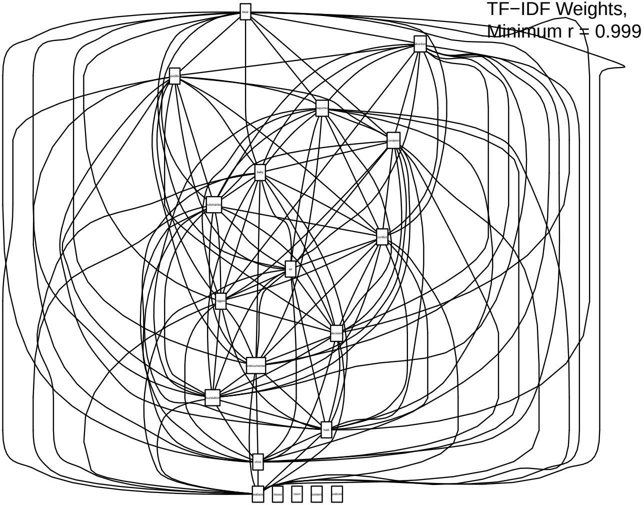
TF vs. TF-IDF Weighting

```
> # Weighting:
>
> D16.TFW <- weightTf(D16.TDM)
> D16.TFIDFW <- weightTfIdf(D16.TDM)
>
> as.matrix(D16.TFW)[1:8,]
       Docs
Terms   Debate2016-1.pdf Debate2016-2.pdf Debate2016-3.pdf
  ?have                0                1                0
  ?his                 0                0                1
  ?let                 0                0                1
  ?mr                  0                1                0
  ?your                0                2                0
  ?04                  1                0                0
  ?13                  1                0                0
  ?14                  1                0                0
> as.matrix(D16.TFIDFW)[1:8,]
       Docs
Terms   Debate2016-1.pdf Debate2016-2.pdf Debate2016-3.pdf
  ?have          0.00000          0.00021           0.0000
  ?his           0.00000          0.00000           0.0002
  ?let           0.00000          0.00000           0.0002
  ?mr            0.00000          0.00021           0.0000
  ?your          0.00000          0.00042           0.0000
  ?04            0.00019          0.00000           0.0000
  ?13            0.00019          0.00000           0.0000
  ?14            0.00019          0.00000           0.0000
```

TF vs. TF-IDF (continued)

```
> TFs<-findMostFreqTerms(D16.TFW,n=20) # top-20 terms
> TFIDFs<-findMostFreqTerms(D16.TFIDFW,n=20) # in each
> cbind(names(TFs$‘Debate2016-1.pdf‘),c(names(TFIDFs$‘Debate2016-1.pdf‘)))
      [,1]        [,2]
 [1,] "trump"     "holt"
 [2,] "clinton"   "interruption"
 [3,] "it?s"      "lester"
 [4,] "going"     "police"
 [5,] "holt"      "percent"
 [6,] "think"     "black"
 [7,] "people"    "frisk"
 [8,] "country"   "sean"
 [9,] "will"      "hannity"
[10,] "we?re"     "stamina"
[11,] "just"      "website"
[12,] "look"      "learn"
[13,] "said"      "losing"
[14,] "that?s"    "leaving"
[15,] "well"      "nato"
[16,] "want"      "certificate"
[17,] "know"      "concerned"
[18,] "one"       "crosstalk"
[19,] "secretary" "fed"
[20,] "get"       "murders"
```

TDM Plot, using TF-IDF Weights



Wrap-Up / Takeaways

- Things we didn't talk much about:
  - Data sources (web scraping, APIs, OCRing scans, etc.)
  - Text data formats (HTML, XML, JSON, etc.)
  - Regular expressions
  - R alternatives (mostly Python, also others)
- Always start with a goal
- Conduct sensitivity analyses
- Text analysis: statistics < programming