Using Computational Technology to Organize, Understand, and Interpret Text

Christopher Zorn
Pennsylvania State University

March 14, 2021

LINK: **bit.ly/SU-lnk**

Ioan. Stradanus inuent.
Phls Galle excud.

Thinking of Text as Data

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 $\rightsquigarrow$ Analyzing a single straw of hay
  - Humans: amazing (e.g., the humanities)
  - Computers struggle
- Comparing, Organizing, and Classifying Text $\rightsquigarrow$ Organizing a hay stack
  - Humans: terrible. Tiny active memories
  - Computers: amazing

Four principles:

1. All models of language are wrong, but some are useful. 😕
2. Quantitative methods for text *amplify resources* and *augment humans*.
3. There is no one globally best method for automated text analysis.
4. *Validate, validate, validate*.

Analyzing Text: Basic Terminology

- Word / Term: A single collection of letters signifying some meaning(s).
- N-gram: A collection of two or more words, treated as a word / term.
- Document: A natural collection of terms.
- 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”).

Text Preprocessing

One approach:

- Remove capitalization
- Tokenize
- Remove punctuation
- 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

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 ("Ben Folds" $\neq$ "Ben folds")
    - Acronyms ("CAT" $\neq$ "cat," etc.)
- Alternative: "truecasing"...

Punctuation:

- Periods, commas, colons, semicolons are usually removed...
- 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 and Stems

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

Stemming...

- Reducing words to their "roots" $\rightarrow$ creating equivalence classes
- Valuable to reduce dimensionality of documents, *but*
- Can occasionally mislead
- Industry standard is the "snowball" stemmer (details at `http://snowballstem.org/`)

Stop Words

- We *usually* want to remove them...
- One list of standard 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 <u>can</u> be a stop word...

How Might I Do All This?

Here, R :

- "R is a free software environment for statistical computing and graphics." (from `r-project.org`)
- Available at *inter alia* CRAN
- Useful R packages for text:
  - `tm`
  - `tidytext`
  - `quanteda`
  - Also: The NLP Task View
- Other alternatives: Python, Java, commercial options

An Ink-Themed Joke

A man visits a tattoo parlor with a rather simple, but strange request. He requests a short, straight line tattooed on his upper arm.

Once the first tattoo heals, he returns, asking for another, exactly the same as the first.

After a few more visits, it becomes clear to the tattoo artist that he's tattooing tally marks on the customer's arm.

Curiosity gets the better of the tattoo artist, and she asks, "What are you counting?"

The man answers, "How many tattoos I have."

Examining Text in R

Create a text object called "Joke":

```
> Joke <- "A man visits a tattoo parlor with a rather simple, but strange request. He requests a short,
           straight line tattooed on his upper arm. Once the first tattoo heals, he returns, asking for
           another, exactly the same as the first. After a few more visits, it becomes clear to the tattoo
           artist that he's tattooing tally marks on the customer's arm. Curiosity gets the better of the
           tattoo artist, and she asks, 'What are you counting?' The man answers, 'How many tattoos
           I have.'"
```

Remove capitalization:

```
> tolower(Joke)

[1] "a man visits a tattoo parlor with a rather simple, but strange request. he requests a short,
straight line tattooed on his upper arm. once the first tattoo heals, he returns, asking for
another, exactly the same as the first. after a few more visits, it becomes clear to the tattoo
artist that he's tattooing tally marks on the customer's arm. curiosity gets the better of the
tattoo artist, and she asks, 'what are you counting?' the man answers, 'how many tattoos
i have.'"
```

Remove punctuation:

```
> removePunctuation(Joke)

[1] "A man visits a tattoo parlor with a rather simple but strange request He requests a short
straight line tattooed on his upper arm Once the first tattoo heals he returns asking for
another exactly the same as the first After a few more visits it becomes clear to the tattoo
artist that hes tattooing tally marks on the customers arm Curiosity gets the better of the
tattoo artist and she asks What are you counting The man answers How many tattoos
I have"
```

“Tokenizing”

Break into sentences:

```
> Joke.sent <- tokenize_sentences(Joke)
> Joke.sent
[[1]]
[1] "A man visits a tattoo parlor with a rather simple, but strange request."
[2] "He requests a short, straight line tattooed on his upper arm."
[3] "Once the first tattoo heals, he returns, asking for another, exactly the same as the first."
[4] "After a few more visits, it becomes clear to the tattoo artist that he’s tattooing tally marks on the customer’s arm."
[5] "Curiosity gets the better of the tattoo artist, and she asks, ’What are you counting?’"
[6] "The man answers, ’How many tattoos I have.’"

> length(Joke.sent[[1]])
[1] 6
```

Break into words:

```
> Joke.words <- tokenize_words(Joke)
> Joke.words
[[1]]
 [1] "a"         "man"       "visits"    "a"         "tattoo"    "parlor"    "with"      "a"         "rather"
[10] "simple"    "but"       "strange"   "request"   "he"        "requests"  "a"         "short"     "straight"
[19] "line"      "tattooed"  "on"        "his"       "upper"     "arm"       "once"      "the"       "first"
[28] "tattoo"    "heals"     "he"        "returns"   "asking"    "for"       "another"   "exactly"   "the"
[37] "same"      "as"        "the"       "first"     "after"     "a"         "few"       "more"      "visits"
[46] "it"        "becomes"   "clear"     "to"        "the"       "tattoo"    "artist"    "that"      "he’s"
[55] "tattooing" "tally"     "marks"     "on"        "the"       "customer’s" "arm"      "curiosity" "gets"
[64] "the"       "better"    "of"        "the"       "tattoo"    "artist"    "and"       "she"       "asks"
[73] "what"      "are"       "you"       "counting"  "the"       "man"       "answers"   "how"       "many"
[82] "tattoos"   "i"         "have"

> length(Joke.words[[1]]) # total word count
[1] 84
```

Counting Things

Count words per sentence:

```
> Joke.wordcount <- sapply(tokenize_words(Joke.sent[[1]]), length)

> Joke.wordcount
[1] 13 11 16 21 15  8

> mean(Joke.wordcount)
[1] 14
```

"Term frequencies":

```
> termFreq(Joke,control=list(removePunctuation=TRUE,
+                            wordLengths=c(1,Inf)))

        a     after       and   another   answers       are       arm    artist        as    asking 
        5         1         1         1         1         1         2         2         1         1 
     asks   becomes    better       but     clear  counting curiosity customers   exactly       few 
        1         1         1         1         1         1         1         1         1         1 
    first       for      gets      have        he     heals       hes       his       how         i 
        2         1         1         1         2         1         1         1         1         1 
       it      line       man      many     marks      more        of        on      once    parlor 
        1         1         2         1         1         1         1         2         1         1 
   rather   request  requests   returns      same       she     short    simple  straight   strange 
        1         1         1         1         1         1         1         1         1         1 
    tally    tattoo  tattooed tattooing   tattoos      that       the        to     upper    visits 
        1         4         1         1         1         1         8         1         1         2 
     what      with       you 
        1         1         1 
```

Eliminating Stop-Words and Basic Stemming

Eliminate stop-words:

```
> removeWords(Joke,stopwords("en"))

[1] "A man visits  tattoo parlor   rather simple,  strange request. He requests
short, straight line tattooed   upper arm. Once  first tattoo heals,  returns,
asking  another, exactly     first. After    visits,  becomes clear   tattoo artist
tattooing tally marks customer's arm. Curiosity gets  better   tattoo artist,
asks, 'What   counting?' The man answers, 'How many tattoos I .'"
```

Basic stemming (using the Snowball stemmer):

```
> stemDocument(Joke)

[1] "A man visit a tattoo parlor with a rather simple, but strang request.
He request a short, straight line tattoo on his upper arm. Once the first
tattoo heals, he returns, ask for another, exact the same as the first. After
a few more visits, it becom clear to the tattoo artist that he tattoo talli
mark on the custom arm. Curios get the better of the tattoo artist, and
she asks, What are you counting? The man answers, How mani tattoo I have."
```

A Basic Corpus (multiple documents)

Create a simple corpus:

```
> Joke.clean <- removePunctuation(Joke.sent[[1]])
> IDJ<-Corpus(VectorSource(Joke.clean))
> inspect(IDJ)

<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 6

[1] A man visits a tattoo parlor with a rather simple but strange request
[2] He requests a short straight line tattooed on his upper arm
[3] Once the first tattoo heals he returns asking for another exactly the same
    as the first
[4] After a few more visits it becomes clear to the tattoo artist that hes tattooing
    tally marks on the customers arm
[5] Curiosity gets the better of the tattoo artist and she asks What are you counting
[6] The man answers How many tattoos I have
```

Term-Document and Document-Term Matrices

A term-document matrix has:

- N rows, $i \in \{1, 2, ...N\}$, corresponding to the N unique terms in the corpus
- J columns, $j \in \{1, 2, ...J\}$, corresponding to the J documents in the corpus
- Entries N_{ij} are the *term frequencies* – the number of times term i appears in document j

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

Example Term-Document Matrix

```
> IDJ.TDM <- TermDocumentMatrix(IDJ,control=list(tolower=TRUE,
+                                                 stemming=TRUE))

> inspect(IDJ.TDM)

<<TermDocumentMatrix (terms: 50, documents: 6)>>
Non-/sparse entries: 64/236
Sparsity           : 79%
Maximal term length: 8
Weighting          : term frequency (tf)
Sample             :
         Docs
Terms      1 2 3 4 5 6
  after    0 0 0 1 0 0
  arm      0 1 0 1 0 0
  artist   0 0 0 1 1 0
  ask      0 0 1 0 1 0
  first    0 0 2 0 0 0
  man      1 0 0 0 0 1
  request  1 1 0 0 0 0
  tattoo   1 1 1 2 1 1
  the      0 0 3 2 2 1
  visit    1 0 0 1 0 0
```

What Can We Do?

Three examples:

- Text similarity
- Topic models
- Dictionary-based methods (e.g., sentiment analysis)

Text Similarity

What makes two documents "similar"?

- Describe similar *concepts*
- "Semantically close" (similar *style*)

<u>Idea</u>: *distance* between two documents **A** and **B**...

- In K-space: *Euclidean* distance is

$$D_{\mathbf{A}\to\mathbf{B}} = \sqrt{\sum_{k=1}^{K} (A_k - B_k)^2}$$

- Two points at the same location have $D_{\mathbf{A}\to\mathbf{B}} = 0$

Cosine Similarity / Distance

Cosine *similarity*:

$$\begin{aligned} S_{cos} &= \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \, ||\mathbf{B}||} \\ &= \frac{\sum_{k=1}^{K} A_k B_k}{\sqrt{\sum_{k=1}^{K} A_k^2} \sqrt{\sum_{k=1}^{K} B_k^2}} \end{aligned}$$

- ...is the cosine of the angle between the two vectors in K-space
- ...ranges from zero (for two documents with no terms in common) to one (for two documents with all terms in common)

Cosine *distance*:

$$D_{cos} = 1 = S_{cos}$$

Cosine Distance: Examples

“Documents”:

A: “Accidentally drank invisible ink. I’m at the hospital waiting to be seen.”
B: “Accidentally accidentally drank drank invisible invisible ink ink. I’m I’m at at the the hospital hospital waiting waiting to to be be seen seen.”
C: “Accidentally prank invisible mink. I’m cat the hospitality waiting too be scene.”
D: “Why were all the ink spots crying? Their father was in the pen.”
E: “What do you call a bear with no teeth? A gummy bear.”

Cosine distances:

A and B = 0.000
A and C = 0.553
A and D = 0.717
A and E = 1.000

Example Two: Dad Jokes

- $N = 144$ Dad jokes
- Source: a February 23, 2022 online article in *Country Living*
- Jokes contain a total of 667 “terms”
- An example: “What do a tick and the Eiffel Tower have in common? They’re both Paris sites.”

Dad Jokes: Distance

Calculate $\frac{(144 \times 143)}{2} = 10296$ "distances" between pairs of jokes...

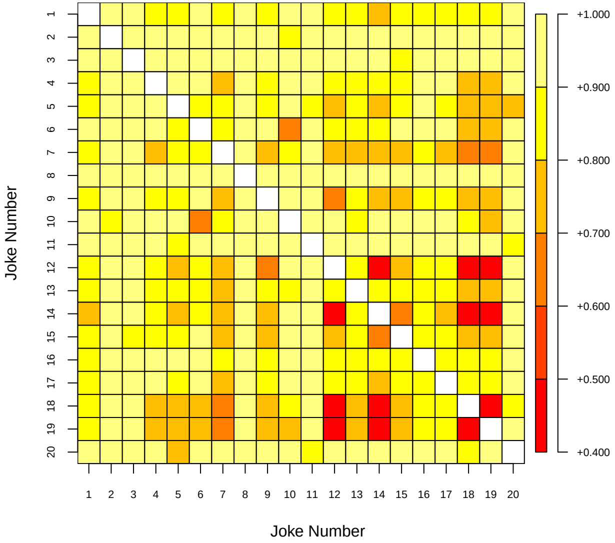
Which jokes are the most similar? There are two pairs with $D_{cos} = 0.147$:

- "How do you make a tissue dance? You put a little boogie in it."
- "How do you make a Kleenex dance? You put a little boogie in it."

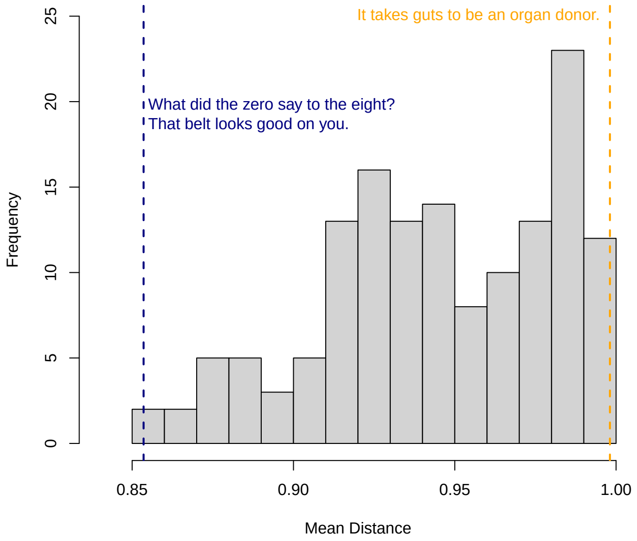
and

- "If a child refuses to sleep during nap time, are they guilty of resisting a rest?"
- "If a child refuses to nap, are they guilty of resisting a rest?"

Visualization



Distinctiveness (Mean Distance)



Topics in Text

- "Topics" / "themes" / etc.: What the document is <u>about</u>.
- How do we know?
  - Word meanings...
  - <u>Clustering</u> of words
  - Tone (sometimes)
- Complications / challenges...
  - What's a "topic"?
  - (Key)words can be ambiguous ("tennis" vs. "crane")
  - Documents are often about > one topic

Learning Topics

Dictionary-based / Supervised methods

- Predetermined "topics" (think: lists of key words)
- Document is "about" whatever topic(s) have (proportionally) the most terms

Unsupervised methods

- Extract topics from the corpus itself
- Intuition: *co-occurrence* of terms in documents
- Useful when (a) we don't know topics *a priori*, and/or (b) term meaning/usage is complex / nonstandard

Example: Latent Dirichlet Allocation (LDA)

Intuition:

- Each document comprises a mixture of one or more of *k* topics
- Each topic comprises a mixture of terms
- We observe documents and terms, but not topics; topics are *latent*
- Goals:
  - Infer the latent topic structure of the corpus
  - Assign documents (probabilistically) to topics
- Process:
  - Assign words to topics
  - Assess Pr(topic | document) and Pr(word | topic)
  - Reassign words to topic
  - Repeat...

(Reference)

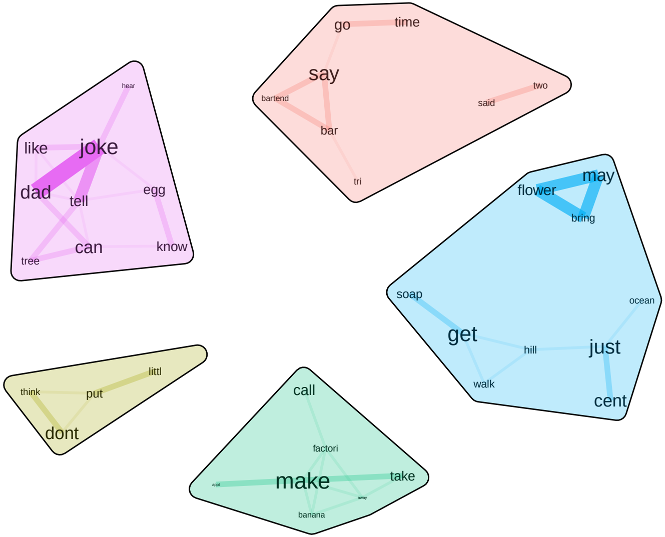
Example: Biterm Topic Model (BTM)

Intuition:

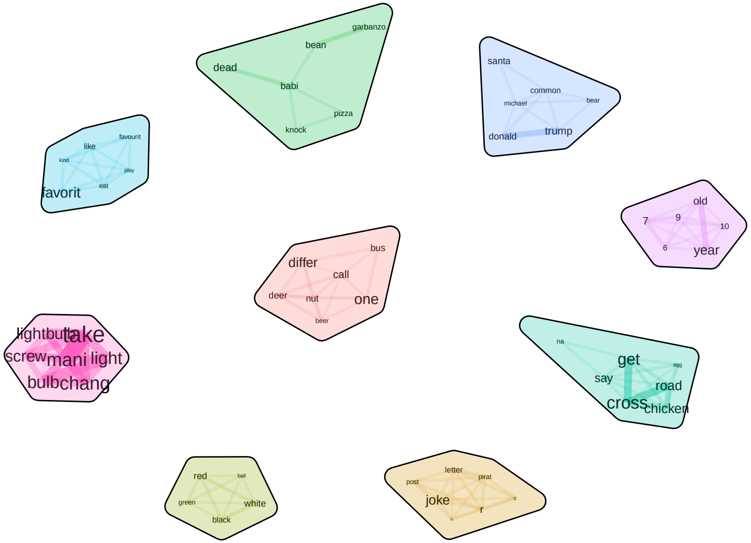
- *Biterms*: two words co-occurring in the same (short) window of text
- Premise: The corpus comprises a mixture of one or more of k topics
- Biterms are drawn (probabilistically) from the mix of topics in the corpus
- Intuition: If a corpus is (say) 50 percent about curling and 50 percent about biathlon...
  - ...the term(s) "shot" / "shoot" will appear in short texts about both, but
  - ...the biterm "shot rock" will occur in short texts only former, and
  - ... "shoot clean" will only occur in the latter
- Same goals as LDA, but...
- Advantages:
  - Works better on "short" documents ($\leq$ 100 terms)
  - Widely used for (e.g.) tweets, other "bursty" applications

(Reference)

Dad Joke Topics



More (≈ 75k) Joke Topics



Dictionary-Based Methods

Classification task:

- *Categorize* documents into classes, and/or
- *Score* documents degree of association with those classes.

Heuristic: **Dictionaries assign weights to words / terms.**

Formally: For $j \in \{1...J\}$ words in a corpus of $i = \{1...N\}$ documents, the *document score* for document i is:

$$S_i = \frac{\sum_{j=1}^{J} \omega_j X_{ij}}{\sum_{j=1}^{J} X_{ij}}$$

where

- X_{ij} is the number of instances of word j in document i, and
- ω_j is the weight assigned to each word by the dictionary.

Toy Example: “Truthiness”

- Document: {`TRUE FALSE TRUE FALSE TRUE`}
- Dictionary:

Term	ω_j
`TRUE`	1.0
`FALSE`	0.0

- Word counts:

$$\mathbf{X} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

- Score:

$$S_i = \frac{(1.0 \times 3) + (0.0 \times 2)}{(3+2)} = \frac{3}{5} = 0.6$$

Dictionary-Based Classification Tasks

- Topic(s)
  - What are documents *about*?
  - What thing(s) are *emphasized*?
- Tone / Style
  - Authorship / provenance
  - Specialization of language (e.g., "hold harmless")
- Sentiment
  - What is the *emotional valence* of the documents?
  - What are the $\underline{\text{emotions}}$ expressed? (pity, anger, jealousy, etc.)
  - Other valences...

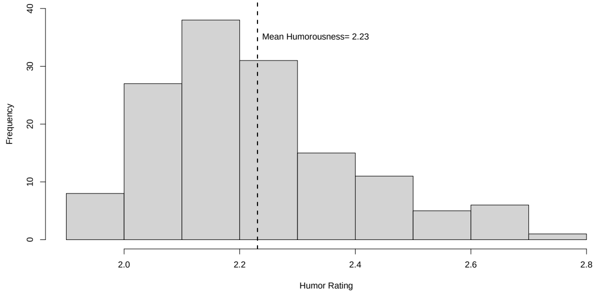
An Example: Humorousness

Englethaler, Tomas, and Thomas H. Tills. 2018. "Humor Norms for 4,997 English Words." *Behavior Research Methods* 50:1116-1124. [cite]

- Asked 821 participants to rate 211 words randomly chosen from a list of 5,000
- Rating: 1 = humorless / "not funny at all" $\longrightarrow$ 5 = "most funny"
- Calculated mean humor ratings (average MHR = 2.41)
- Among the least funny words: torture (1.26), torment (1.30), gunshot (1.31), death (1.32), deathbed (1.39)
- Among the funniest words: booty (4.32), nitwit (4.03), waddle (4.00), bebop (3.93), twerp (3.92)

$\rightarrow$ Scale our jokes by their humorousness...

Dad Joke Humorousness



Funniest: "I could tell a joke about pizza, but it's a little cheesy."
Runner-up: "I like telling Dad jokes. Sometimes he laughs."

Least funny: "What do you call a poor Santa Claus? St. Nickel-less."
Runner-up: "Dear Math: Grow up and solve your own problems."

Other Things You Might Do...

- Text Scaling
  - Related to item response theory
  - Allows "location" of texts / authors / speakers / etc.
  - Applications: political speech, sentiment intensity
- Parts of Speech / Dependency Parsing
- "Latent Semantic Analysis" – extract meaning from text
- Many other things...

Thank you!