Introduction to Text Analysis

It may seem obvious but.. what is text?

- From a sociolinguistic perspective: any symbolically encoded language.

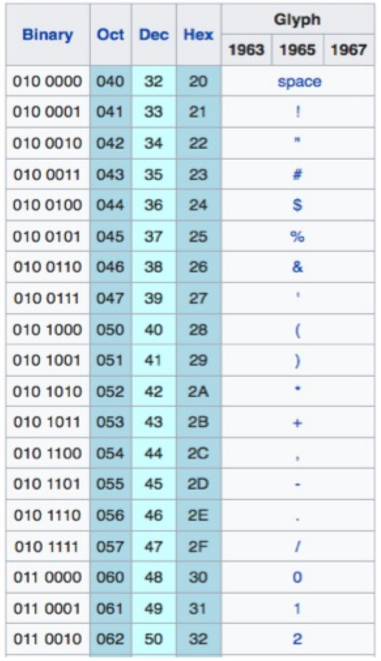
These come in many types:

  - Alphabetic – symbols (letters) to represent vowels and consonants (e.g., Latin)
  - Abjads – symbols to represent consonants, with diacritics (or reused consonants) to represent vowels (e.g., Arabic, Hebrew)
  - Syllabic – Symbol systems representing consonants plus inherent vowels (e.g., Devanagari)
  - Semanto-phonetic – Symbols that carry meaning and/or sounds (e.g., Chinese)

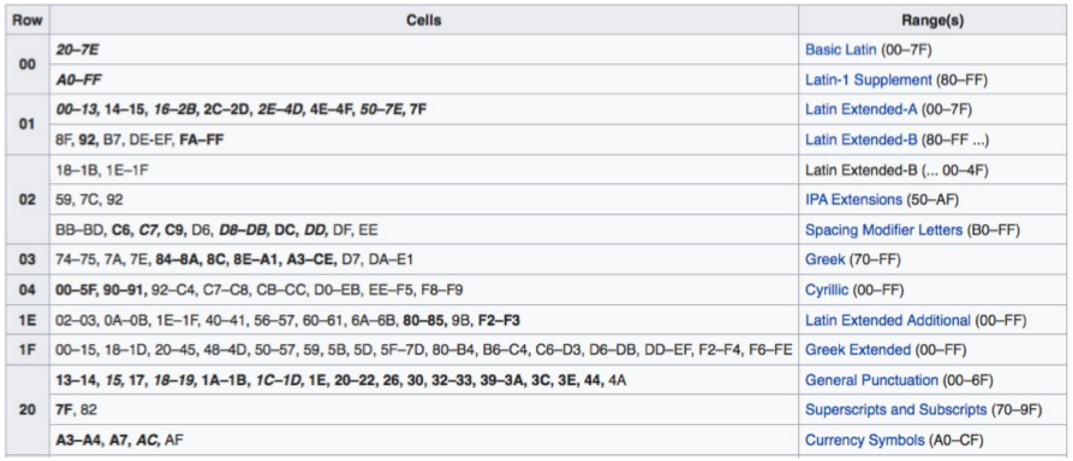
- Plus, there are additional writing systems: Braille, shorthand, phonetic alphabets

What is Text?

- From a computer perspective a binary encoding system for storing written language symbols from the world's various systems



Binary	Oct	Dec	Hex	Glyph		
				1963	1965	1967
010 0000	040	32	20	space		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043	35	23	#		
010 0100	044	36	24	$		
010 0101	045	37	25	%		
010 0110	046	38	26	&		
010 0111	047	39	27	'		
010 1000	050	40	28	(		
010 1001	051	41	29	)		
010 1010	052	42	2A	*		
010 1011	053	43	2B	+		
010 1100	054	44	2C	,		
010 1101	055	45	2D	-		
010 1110	056	46	2E	.		
010 1111	057	47	2F	/		
011 0000	060	48	30	0		
011 0001	061	49	31	1		
011 0010	062	50	32	2		

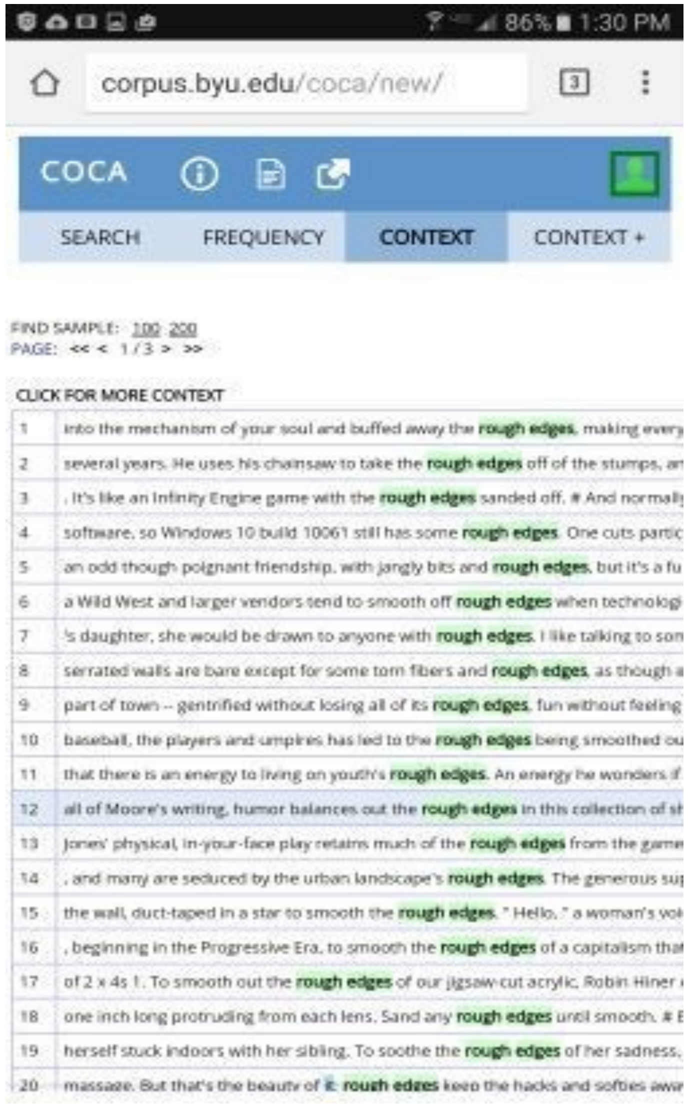


Row	Cells	Range(s)
00	***20–7E***	Basic Latin (00–7F)
	A0–FF	Latin-1 Supplement (80–FF)
01	***00–13,*** **14–15,** ***16–2B,*** **2C–2D,** ***2E–4D,*** **4E–4F,** ***50–7E,*** **7F**	Latin Extended-A (00–7F)
	8F, **92,** B7, DE-EF, **FA–FF**	Latin Extended-B (80–FF ...)
02	18–1B, 1E–1F	Latin Extended-B (... 00–4F)
	59, 7C, 92	IPA Extensions (50–AF)
	BB–BD, **C6,** ***C7,*** **C9,** D6, ***D8–DB,*** **DC,** ***DD,*** DF, EE	Spacing Modifier Letters (B0–FF)
03	74–75, 7A, 7E, **84–8A, 8C, 8E–A1, A3–CE,** D7, DA–E1	Greek (70–FF)
04	**00–5F, 90–91,** 92–C4, C7–C8, CB–CC, D0–EB, EE–F5, F8–F9	Cyrillic (00–FF)
1E	02–03, 0A–0B, 1E–1F, 40–41, 56–57, 60–61, 6A–6B, **80–85,** 9B, **F2–F3**	Latin Extended Additional (00–FF)
1F	00–15, 18–1D, 20–45, 48–4D, 50–57, 59, 5B, 5D, 5F–7D, 80–B4, B6–C4, C6–D3, D6–DB, DD–EF, F2–F4, F6–FE	Greek Extended (00–FF)
20	**13–14,** ***15,*** **17,** ***18–19,*** **1A–1B,** ***1C–1D,*** **1E, 20–22, 26, 30, 32–33, 39–3A, 3C, 3E, 44,** 4A	General Punctuation (00–6F)
	7F, 82	Superscripts and Subscripts (70–9F)
	A3–A4, A7, ***AC,*** AF	Currency Symbols (A0–CF)

From ASCII (7-bits) to Unicode: An international standard that supports up to four bytes (32 bits) per symbol and that encodes 137,439 characters from 146 different scripts.

What is text?

- From a data scientist's perspective, a set of documents organized into a corpus, where each document contains an encoded sample of written language, generally:
  - Human readable files (e.g., plain text)
  - Metadata describing data source, date of capture, speaker/author, etc.
  - Character/symbol encoding appropriate to language (Arabic, Hebrew..)
  - One file per document, or container file includes document separator characters (e.g., newline)
  - Each file contains unstructured, "natural" language content – generally the same "type" of content for each file in the corpus (i.e., don't mix tweets with book chapters)



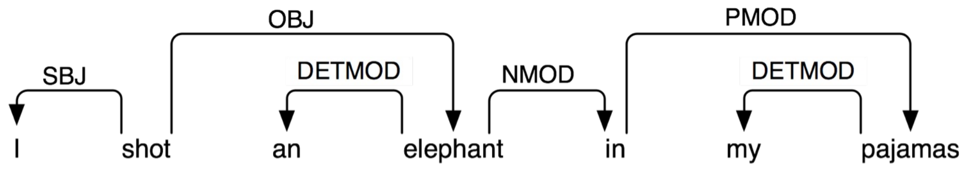
FIND SAMPLE: 100 200
PAGE: << < 1 / 3 > >>

CLICK FOR MORE CONTEXT

1	into the mechanism of your soul and buffed away the **rough edges**, making every
2	several years. He uses his chainsaw to take the **rough edges** off of the stumps, an
3	. It's like an Infinity Engine game with the **rough edges** sanded off. # And normally
4	software, so Windows 10 build 10061 still has some **rough edges**. One cuts partic
5	an odd though poignant friendship, with jangly bits and **rough edges**, but it's a fu
6	a Wild West and larger vendors tend to smooth off **rough edges** when technologi
7	's daughter, she would be drawn to anyone with **rough edges**. I like talking to son
8	serrated walls are bare except for some torn fibers and **rough edges**, as though a
9	part of town -- gentrified without losing all of its **rough edges**, fun without feeling
10	baseball, the players and umpires has led to the **rough edges** being smoothed ou
11	that there is an energy to living on youth's **rough edges**. An energy he wonders if
12	all of Moore's writing, humor balances out the **rough edges** in this collection of sh
13	Jones' physical, in-your-face play retains much of the **rough edges** from the game
14	, and many are seduced by the urban landscape's **rough edges**. The generous sup
15	the wall, duct-taped in a star to smooth the **rough edges**. " Hello, " a woman's voi
16	, beginning in the Progressive Era, to smooth the **rough edges** of a capitalism that
17	of 2 x 4s 1. To smooth out the **rough edges** of our jigsaw-cut acrylic, Robin Hiner
18	one inch long protruding from each lens. Sand any **rough edges** until smooth. # E
19	herself stuck indoors with her sibling. To soothe the **rough edges** of her sadness,
20	massage. But that's the beauty of it: **rough edges** keep the hacks and softies awa

“Unstructured” is the Key

- Natural language is notoriously flexible and ambiguous
  – even simple tasks like parsing are complicated:
- Contractions; Compound words; Misspellings; Proper Names; Preposition Attachment; Vernacular
- Transforming a set of documents into data that allow for comparisons, linkages, visualization, or other analysis, is complex and contains many decisions



Analytical Overview

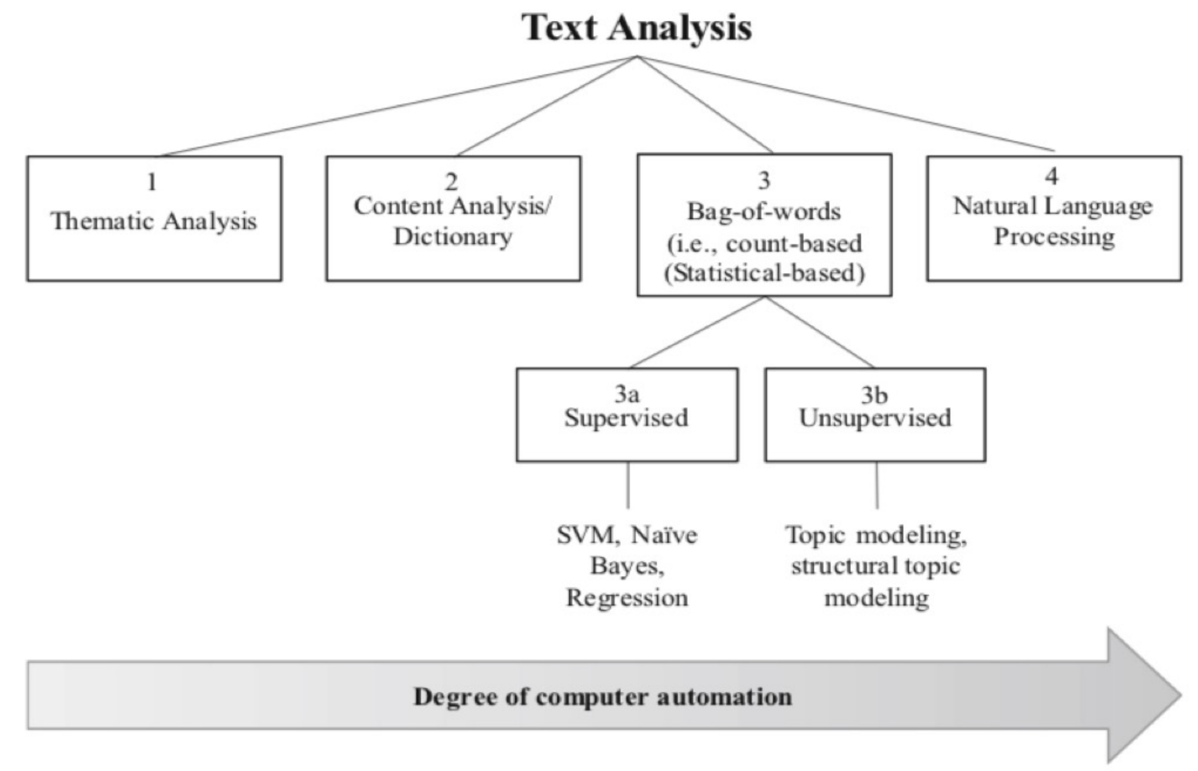
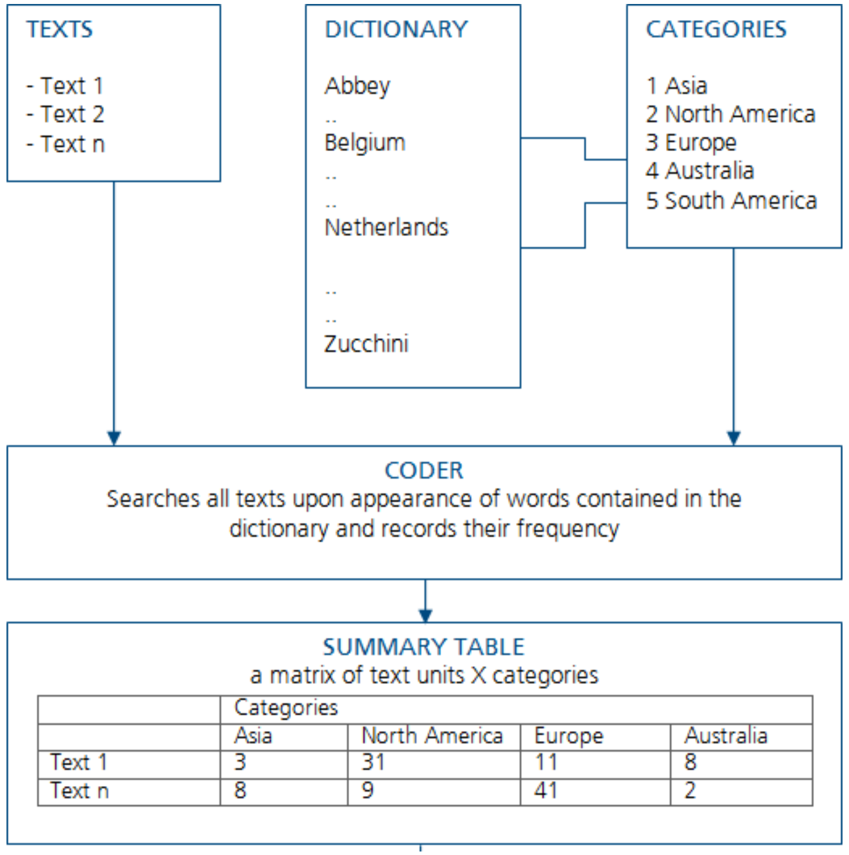

Text Analysis
1 Thematic Analysis
2 Content Analysis/ Dictionary
3 Bag-of-words (i.e., count-based (Statistical-based)
4 Natural Language Processing
3a Supervised
3b Unsupervised
SVM, Naïve Bayes, Regression
Topic modeling, structural topic modeling
Degree of computer automation

Image Credit: Banks et al. (2018), Fig. 1, p. 447

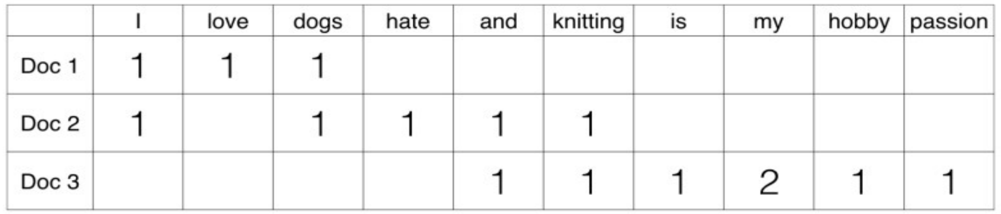
Dictionary-Based Content Analysis

- Simplest form of automated text analysis
- Reduce texts to frequency counts of various vocabulary words
- Lookup vocabulary words in appropriate dictionaries – e.g., positive/negative sentiment
- Quantify results based on connection between word frequency and dictionary connotations
- Results can be used to compare documents to one another or to summarize corpora



	Categories			
	Asia	North America	Europe	Australia
Text 1	3	31	11	8
Text n	8	9	41	2

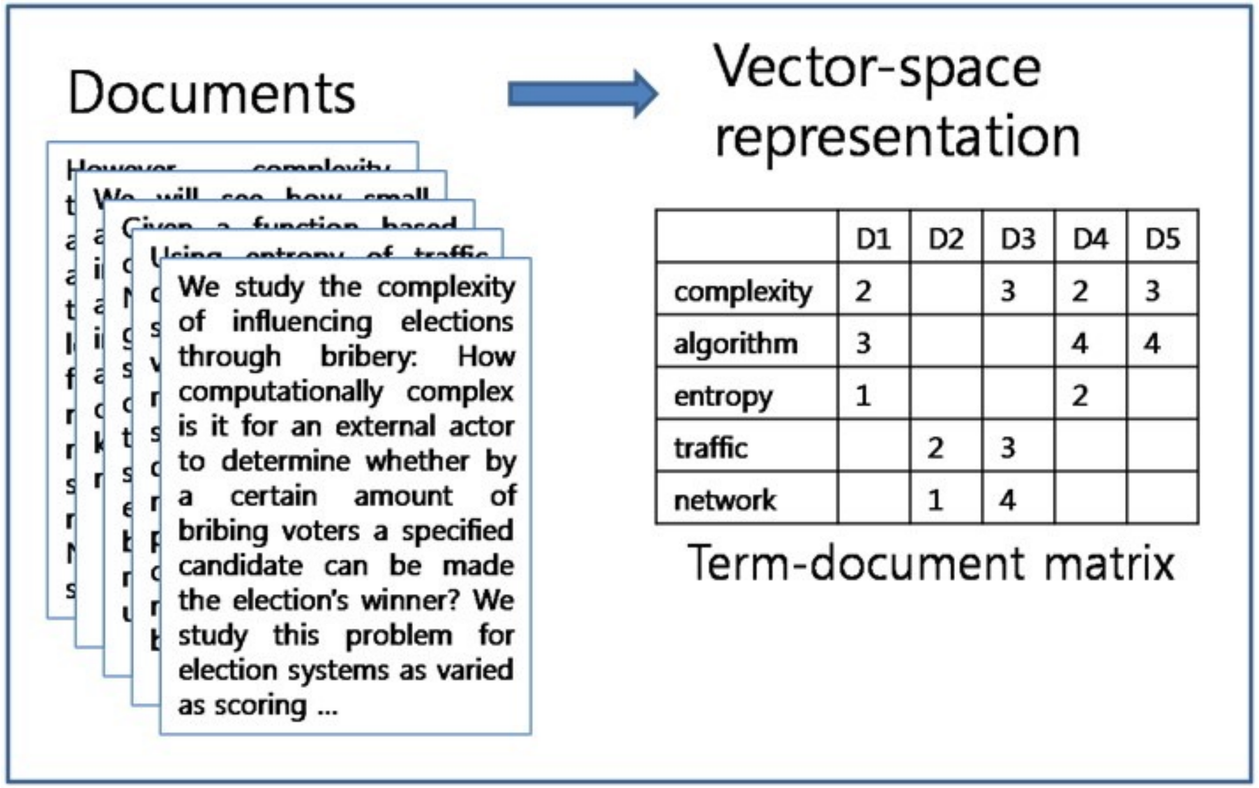
Statistical Approaches



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

- Treat each text fragment as a collection of units/words, without regard to word order
- Use many text fragments – e.g. chapters from a book; each fragment is considered a “document”
- Compute **a term-document matrix (TDM)** (or document term matrix); a sparse matrix pinning down the count of appearance of a term (aka word) in a document
- Conduct statistical or machine learning analysis on the TDM to reveal patterns

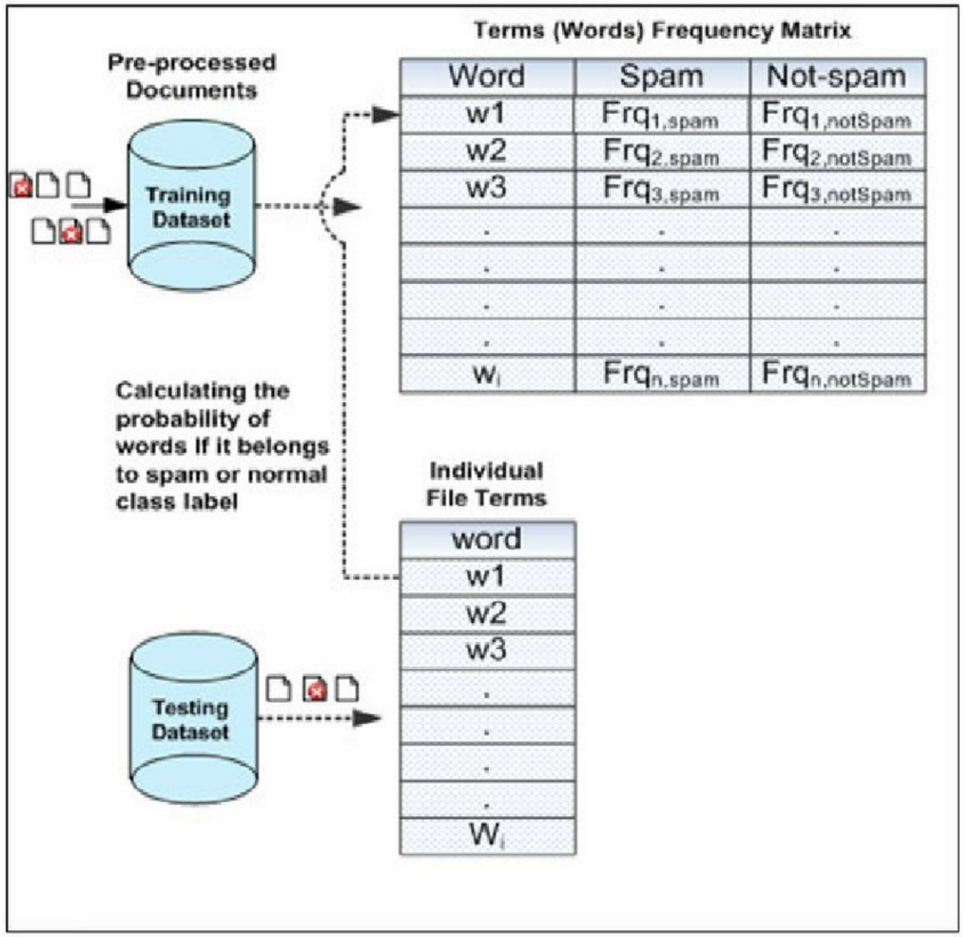
Example TDM



We study the complexity of influencing elections through bribery: How computationally complex is it for an external actor to determine whether by a certain amount of bribing voters a specified candidate can be made the election's winner? We study this problem for election systems as varied as scoring ...

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Example Application: Spam Filter


Pre-processed Documents
Training Dataset
Calculating the probability of words if it belongs to spam or normal class label
Testing Dataset
Individual File Terms
Terms (Words) Frequency Matrix

Terms (Words) Frequency Matrix

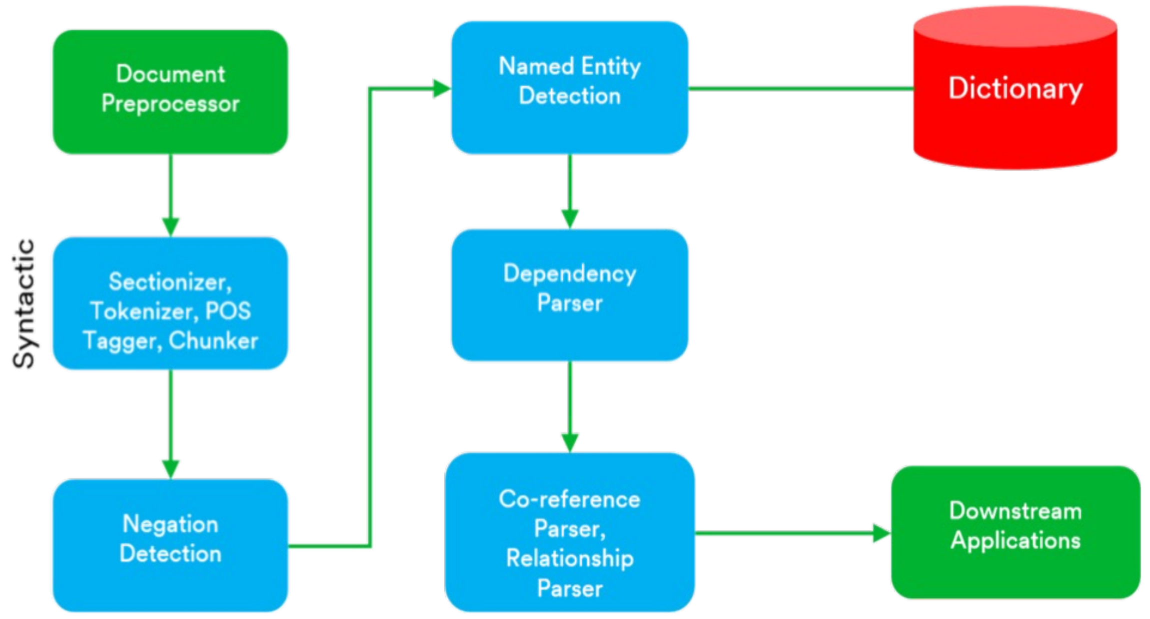
Word	Spam	Not-spam
w1	$Frq_{1,spam}$	$Frq_{1,notSpam}$
w2	$Frq_{2,spam}$	$Frq_{2,notSpam}$
w3	$Frq_{3,spam}$	$Frq_{3,notSpam}$
.	.	.
.	.	.
.	.	.
.	.	.
w_i	$Frq_{n,spam}$	$Frq_{n,notSpam}$

Individual File Terms

word
w1
w2
w3
.
.
.
.
W_i

Natural Language Processing

- Tokenizes and tags text to detect and preserve **part-of-speech**
- Post processing of tokens tries to resolve syntactic dependencies, co-references, names of entities, sequences of events, etc.
- Machine learning can then be used to solve a variety of difficult challenges such question answering, information retrieval, and machine translation, e.g., Alexa, Siri…



Research Questions to Address with Text

- Organizations: Examine the dynamics of team formation by analyzing listserv emails
- Economics: Analyze the content of job postings to estimate the demand for particular jobs across years/regions
- Sociology: Track levels of hate speech over time using text from online forums
- Politics: Detect the proportion of fear-based appeals in campaign rhetoric based on party affiliation
- Psychology: Conduct sentiment analysis on open ended comments in surveys to detect faking

Text Data Sources

- https://github.com/niderhoff/nlp-datasets

Alphabetic list of freely available text corpora

- https://www.gutenberg.org

Project Gutenberg, more than 57,000 public domain books available in various formats

- https://en.wikipedia.org/wiki/List_of_text_corpora

List of corpora for more than 30 languages including “parallel corpora” where the same text is available in two or more languages

- https://gengo.ai/datasets/the-best-25-datasets-fornatural-language-processing/ As the URL suggests. . .

Worked Example 1:

Importing Text and Dictionary Analysis

Overview

- Read in a text file, separating into paragraphs, each of which will be treated as a separate "document"
- Tokenize the text into a bag of words, representing it as a Document-Term Matrix
- Extract term frequency information
- Visualize term frequency information
- Match term frequency to dictionaries of positive and negative connoted words (merge)
- Visualize results
- Research question: In a nomination acceptance speech, do positive sentiments predominate over negative ones?

Read in a Speech

```
library(tm)
charVector <- scan("speech.txt", character(0), sep = "\n")
head(charVector)
```

```
1"I speak tonight of gratitude, achievement, and high hopes
for our country."
2  "Tonight, I think first of those who helped get me here
- starting with the people of Tennessee. Then, those who
braved the first snows of Iowa and New Hampshire -- and all of
you here, from all over this country, who have come with me
into the warm sunlight of this great city."
3  "While I can't thank each of you individually in words,
I do so in my heart."
4  "And I know you won't mind if I single out someone who
has just spoken so eloquently, someone I've loved with my whole
heart since the night of my high school senior prom -- my wife,
Tipper. We've been lucky enough to find each other all over
again at each new stage of our lives - and we just celebrated
our 30th wedding anniversary."
```

Read in Dictionaries

```
posWords <- scan("positive-words.txt", character(0), sep = "\n")  #
2006 items
negWords <- scan("negative-words.txt", character(0), sep = "\n") #
4783 items
head(posWords,15)
head(negWords,15)
```

```
> head(posWords,15)
[1] "a+"            "abound"        "abounds"      [4]
"abundance"     "abundant"      "accessable"
[7] "accessible"    "acclaim"       "acclaimed"
[10] "acclamation"   "accolade"      "accolades"    [13]
"accommodative" "accomodative"  "accomplish"
> head(negWords,15)
[1] "2-faced"     "2-faces"     "abnormal"    "abolish"
[5] "abominable"  "abominably"  "abominate"   "abomination"
[9] "abort"       "aborted"     "aborts"      "abrade"
[13] "abrasive"    "abrupt"      "abruptly"
```

Bag of words: Two class transformations

```
wordVector <- VectorSource(charVector)
wordCorpus <- Corpus(wordVector)
class(wordVector); typeof(wordVector); length(wordVector)
class(wordCorpus); typeof(wordCorpus); length(wordCorpus)
```

```
> class(wordVector); typeof(wordVector); length(wordVector)
[1] "VectorSource" "SimpleSource" "Source"
[1] "list"
[1] 166
> class(wordCorpus); typeof(wordCorpus); length(wordCorpus)
[1] "SimpleCorpus" "Corpus"
[1] "list"
[1] 166
```

Token Clean Up

```
# first step transformation: make all of the letters in "wordCorpus"
lowercase wordCorpus <- tm_map(wordCorpus, content_transformer(tolower))
```

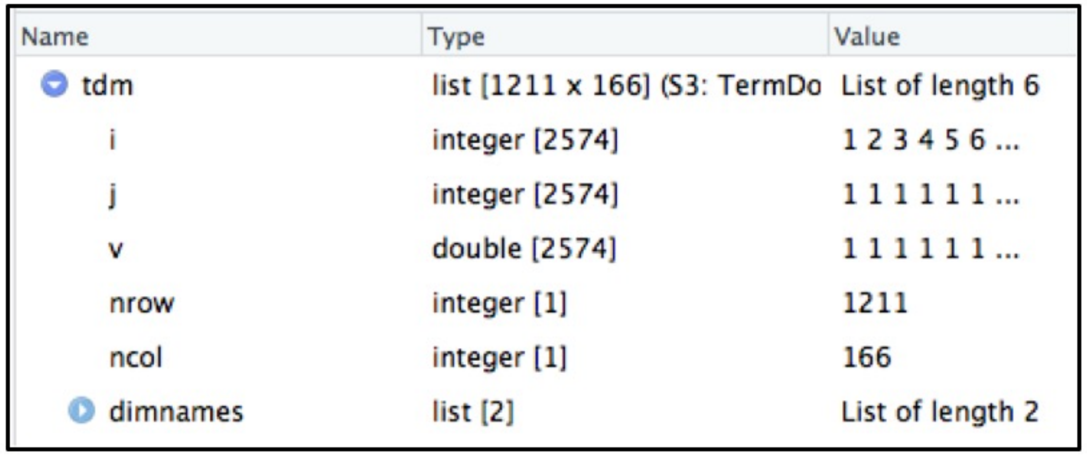
Ignore warnings about dropped documents.

```
# second step transformation:

#remove the punctuation in "wordCorpus" wordCorpus <- tm_map(wordCorpus,
removePunctuation

# third step transformation: remove numbers in "wordCorpus" wordCorpus <-
tm_map(wordCorpus, removeNumbers)

# final step transformation: take out the "stop" words, such as "the", "a"
and "at" wordCorpus <- tm_map(wordCorpus, removeWords,
stopwords("english")) wordCorpus[["1"]][["content"]] # Review what's left
of the first paragraph
```

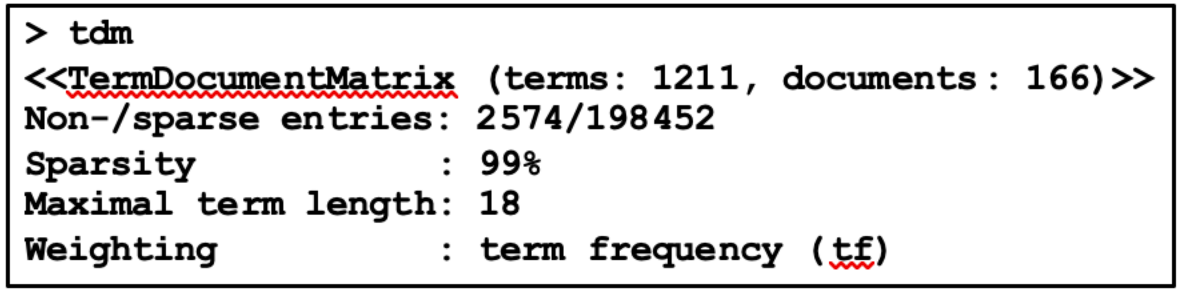
```
> wordCorpus[["1"]][["content"]]
[1] " speak tonight  gratitude achievement  high hopes   country"
```

Create a term-document matrix "tdm"

```
tdm <- TermDocumentMatrix(wordCorpus)

tdm
View(tdm)
```



Name	Type	Value
tdm	list [1211 x 166] (S3: TermDo	List of length 6
i	integer [2574]	1 2 3 4 5 6 ...
j	integer [2574]	1 1 1 1 1 1 ...
v	double [2574]	1 1 1 1 1 1 ...
nrow	integer [1]	1211
ncol	integer [1]	166
dimnames	list [2]	List of length 2



```
> tdm
<<TermDocumentMatrix (terms: 1211, documents: 166)>>
Non-/sparse entries: 2574/198452
Sparsity           : 99%
Maximal term length: 18
Weighting          : term frequency (tf)
```

Some things to do with a TDM

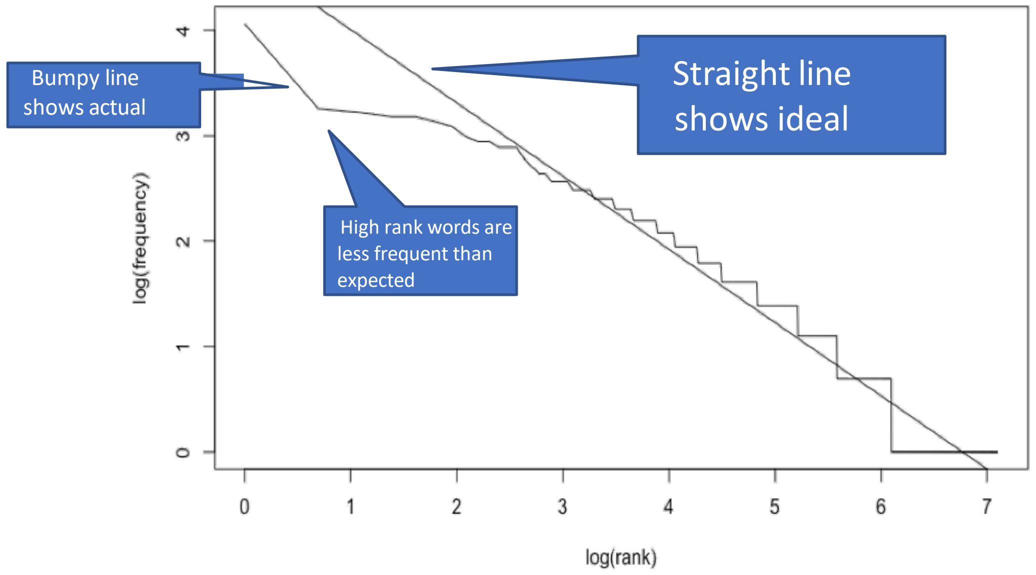
```
fTerms <- findFreqTerms(tdm, lowfreq = 20)
# Common terms: "tonight"  "new"    "people”
#  "children" "family"   "will"     "families"
"america" findAssocs(tdm, fTerms, 0.4)
```

Note: Excerpted results.

```
> findAssocs(tdm, fTerms, 0.4)
$people fighting     champion      charged constitution     district
    0.62         0.53         0.53         0.53         0.53
$children raising
  0.4
$families tax working
        0.41    0.40
```

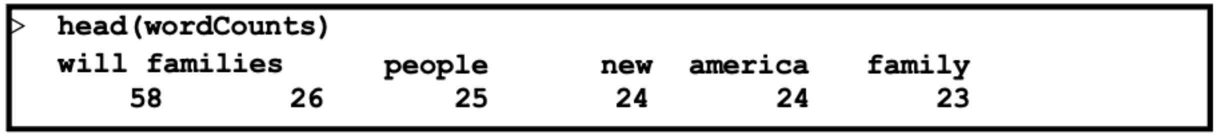
Zipf’s Law: Zipf_plot(tdm)

- The numeric frequency of any word in an NL corpus is negatively related to its frequency rank.
- Example: Most frequent word occurs 2X as much as the next most frequent word, 3X as much as the third most frequent word, etc. . .



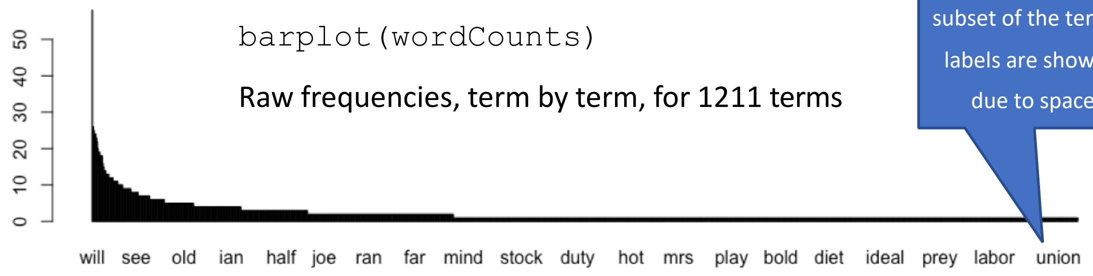
Aggregate TDM into vector of word counts

```
m <- as.matrix(tdm) # Coerce to regular matrix

# create a list of counts for each word named "wordCounts"
wordCounts <- rowSums(m)

# sort words in "wordCounts" by frequency  wordCounts
<- sort(wordCounts, decreasing=TRUE)

# check the first several items in "wordCounts" to see if it is
built correctly  head(wordCounts)
totalWords <- sum(wordCounts) # Calculate the total number of occurrences
```



```
> head(wordCounts)
will families     people      new   america    family
   58       26         25       24        24        23
```

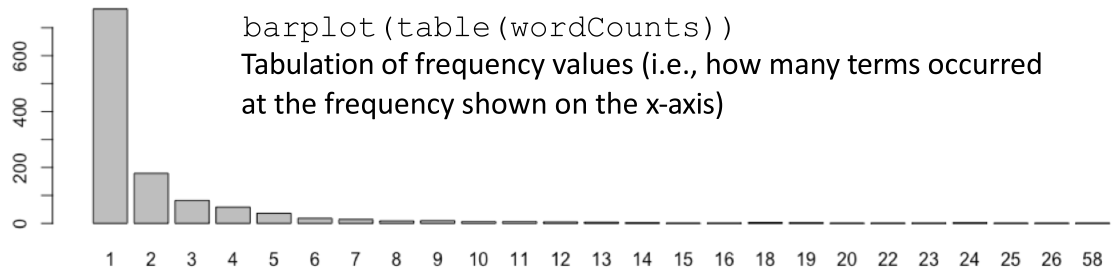
Word Count Bar Plots



`barplot(wordCounts)`

Raw frequencies, term by term, for 1211 terms

Note: only a small subset of the term labels are shown due to space

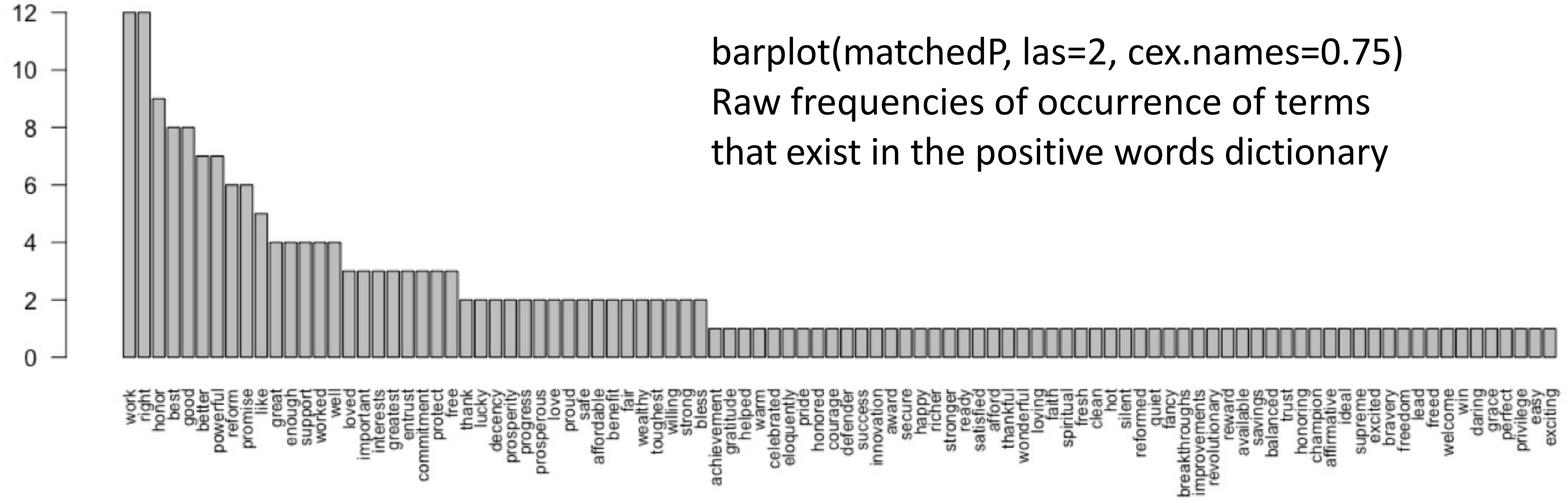


`barplot(table(wordCounts))`

Tabulation of frequency values (i.e., how many terms occurred at the frequency shown on the x-axis)

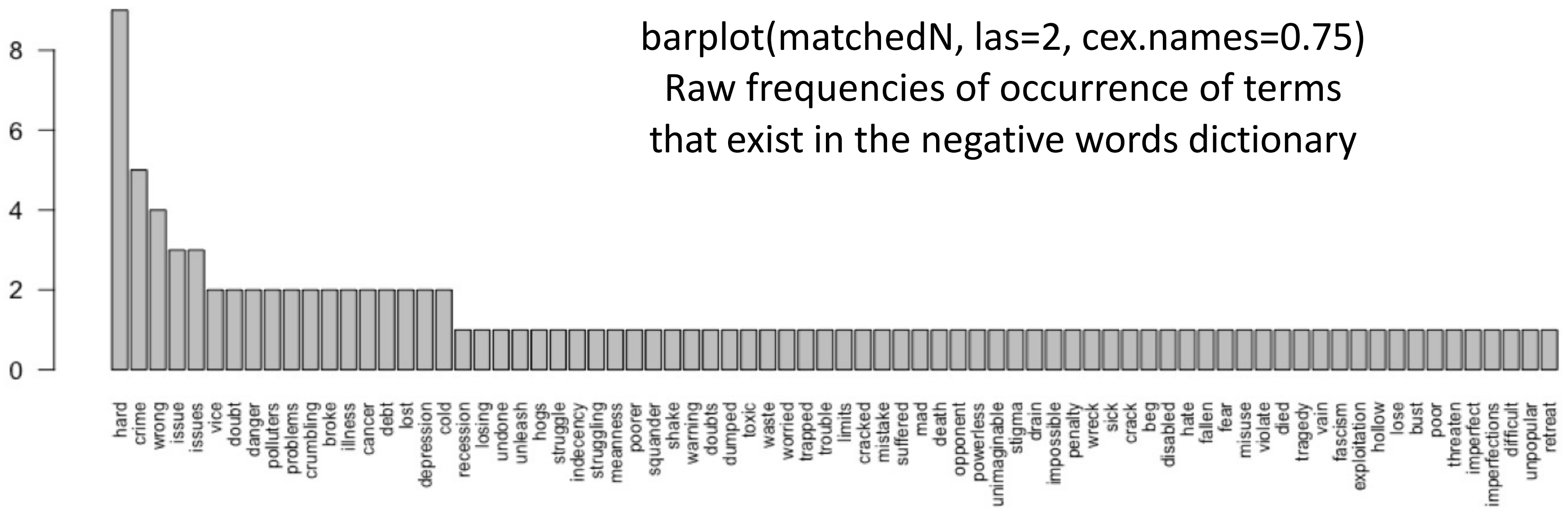
Positive Word Matches: 7.8% of total

```
matchedP <- match(names(wordCounts), posWords, nomatch = 0)
matchedP <- matchedP != 0 # Make a Boolean map
matchedP <- wordCounts[matchedP] # Apply Boolean map
barplot(matchedP,las=2,cex.names=0.75) # Small vertical labels
sum(matchedP)/totalWords # Proportion of pos words vs total
```



barplot(matchedP, las=2, cex.names=0.75)
Raw frequencies of occurrence of terms
that exist in the positive words dictionary

Exercise 14.1: Generate the Negative Word Matches and Interpret Results



barplot(matchedN, las=2, cex.names=0.75)
Raw frequencies of occurrence of terms
that exist in the negative words dictionary

Answering a Research Question

- Dictionary analysis indicated that positive terms occurred twice as frequently as negative terms in this speech
- This result affirms that idea that a nomination acceptance speech is typically an affirmative celebration of success in the primary
- Other recent nomination speeches should be similarly analyzed to see if the pattern holds

Exercise: Analyze another speech

Analyze another presidential acceptance speech using the same code

- Go to:

https://www.presidency.ucsb.edu/documents/appcategories/elections-and-transitions/conventionspeeches

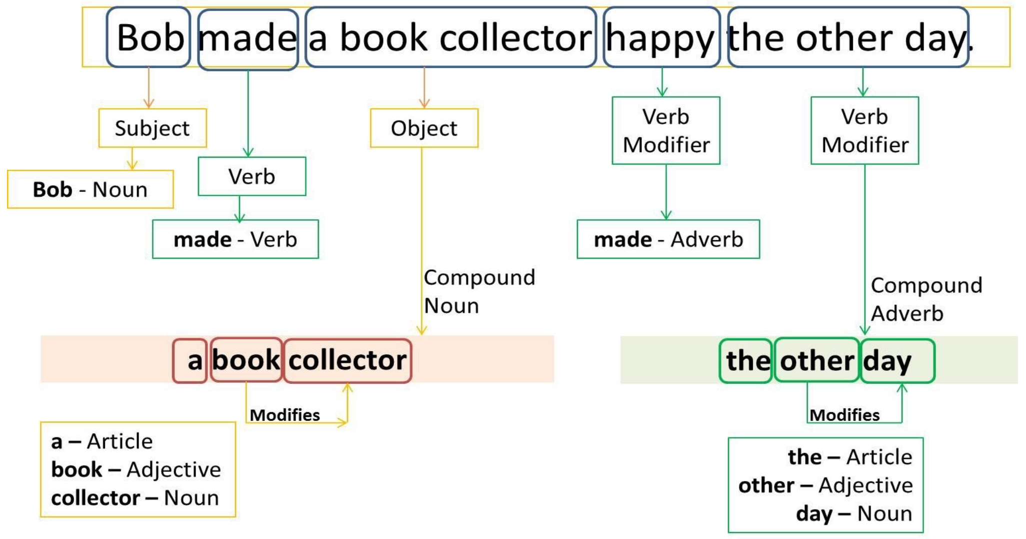
to obtain another speech

- Copy and paste the nominee's remarks from the web page into a plain text file on your computer
- Check to make sure that there are line breaks for each paragraph
- Use the same positive and negative word dictionaries

Brief Review of POS Tagging

Part of Speech Tagging

- NLP parsers contain extensive, language-specific logic for decomposing sentences (tokenizing) into parts of speech



Prepositions – in, on, at, with
Conjuctions – but, or, and, for

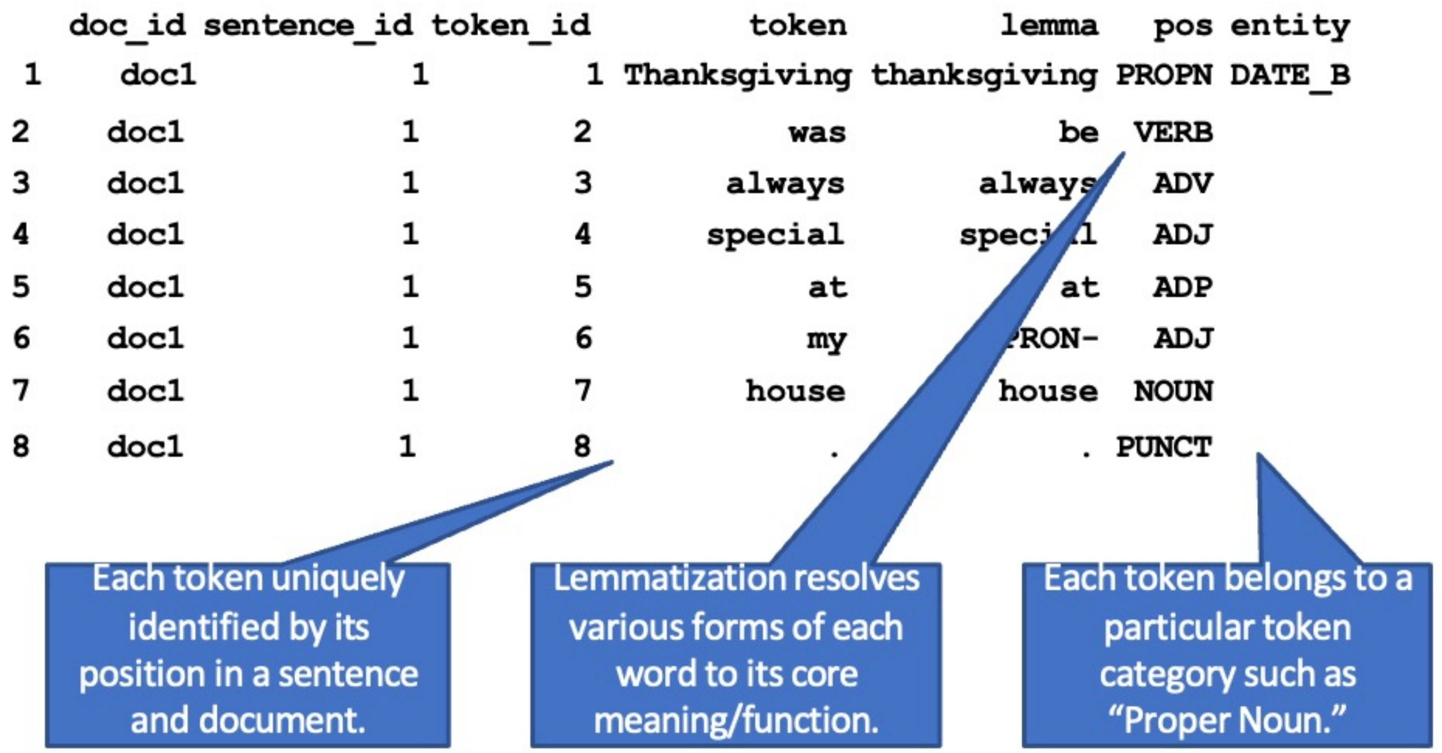
Example using spacyr

- spacyr is an R package “wrapper” around an open source Python package called spaCy

- v2.0 of spaCy is one of the most accurate and fastest parsers available

```
spacy_initialize() # Start the spacy session

inText <- c(doc1="Thanksgiving was always special at my
house. Sarah pointed to my sisters and said, 'They are
eating apples.'",
            doc2="Then Sarah pointed to the bowl that
held the fruit and said, 'They are eating apples.'")
```

Tokenized Output



```
  doc_id sentence_id token_id        token        lemma   pos entity
1   doc1           1        1 Thanksgiving thanksgiving PROPN DATE_B
2   doc1           1        2          was           be  VERB
3   doc1           1        3       always       always   ADV
4   doc1           1        4      special      special   ADJ
5   doc1           1        5           at           at   ADP
6   doc1           1        6           my       -PRON-   ADJ
7   doc1           1        7        house        house  NOUN
8   doc1           1        8            .            . PUNCT
```

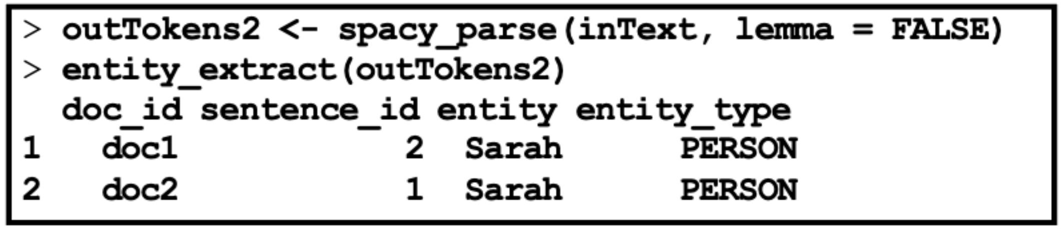
Each token uniquely identified by its position in a sentence and document.

Lemmatization resolves various forms of each word to its core meaning/function.

Each token belongs to a particular token category such as "Proper Noun."

Entity Detection

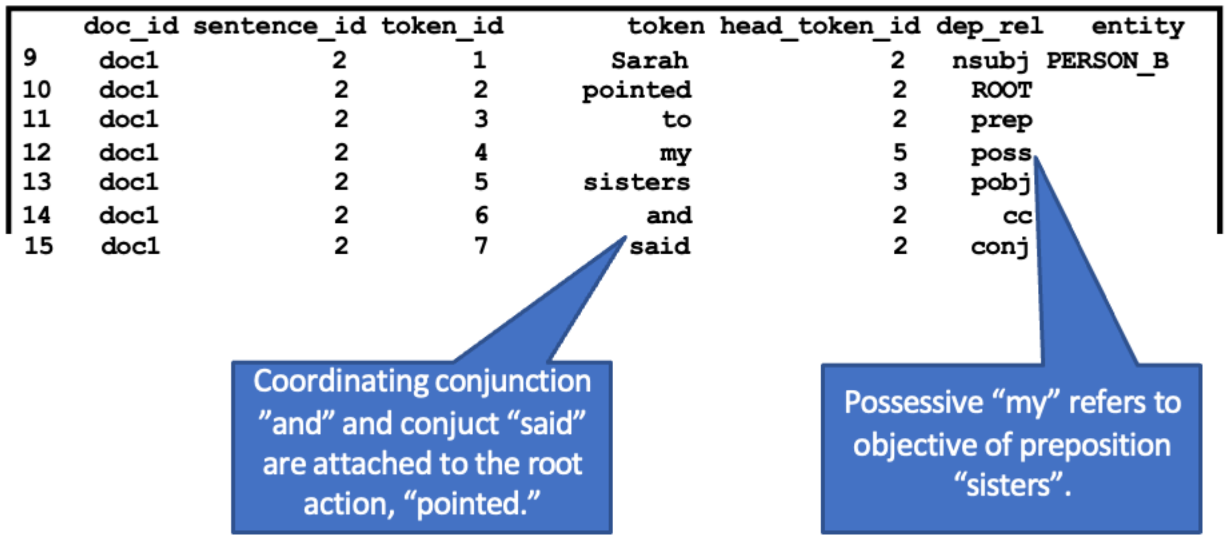
- Parsers usually contain dictionaries and rules that support detection of named entities such as people, places, and organizations



```
> outTokens2 <- spacy_parse(inText, lemma = FALSE)
> entity_extract(outTokens2)
  doc_id sentence_id entity entity_type
1   doc1           2  Sarah      PERSON
2   doc2           1  Sarah      PERSON
```

Dependency Tagging

- Syntactical dependencies show, for example, which noun an adjective modifies

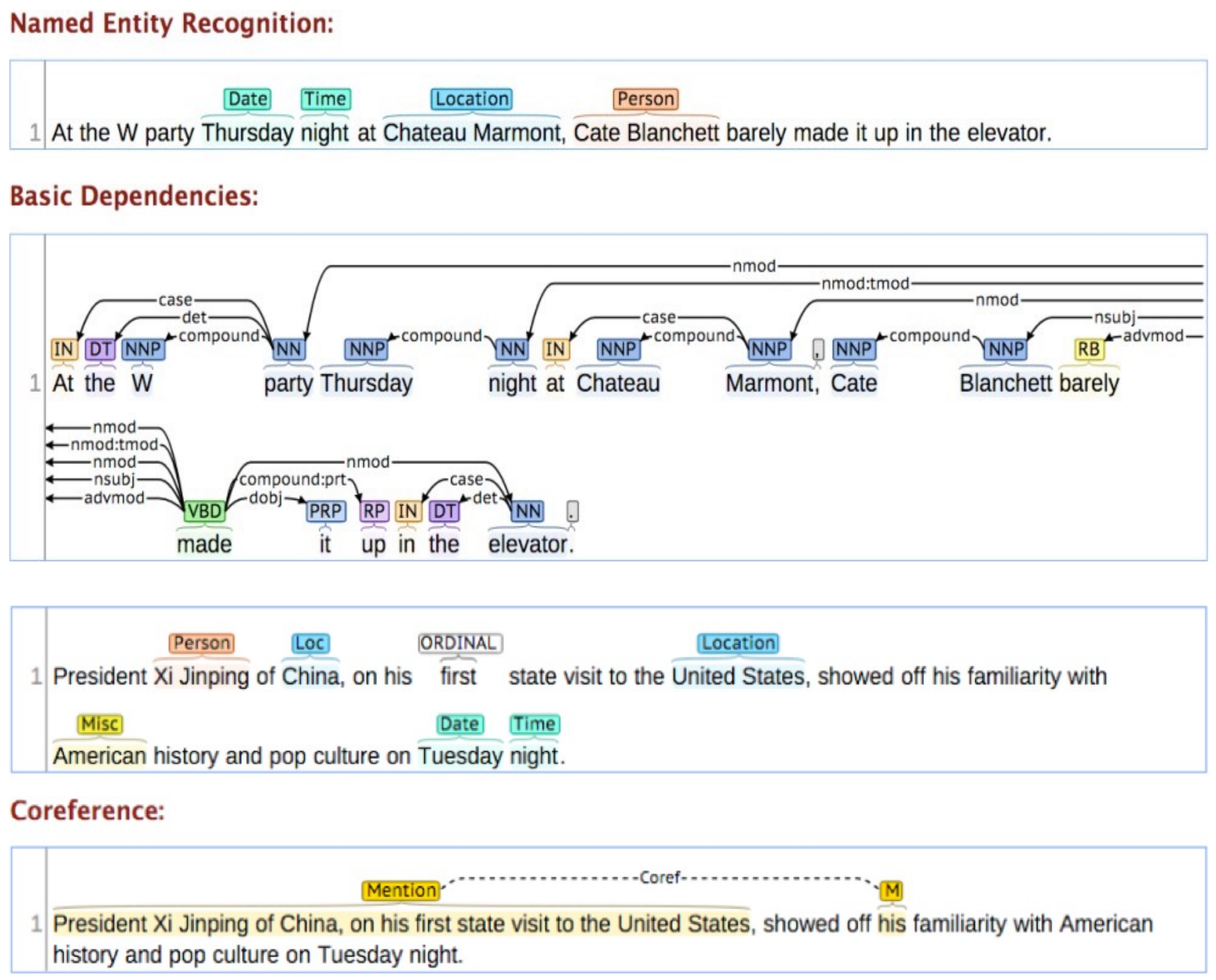


```
    doc_id sentence_id token_id        token head_token_id dep_rel   entity
9     doc1           2        1        Sarah             2   nsubj PERSON_B
10    doc1           2        2      pointed             2    ROOT
11    doc1           2        3           to             2    prep
12    doc1           2        4           my             5    poss
13    doc1           2        5      sisters             3    pobj
14    doc1           2        6          and             2      cc
15    doc1           2        7         said             2    conj
```

Coordinating conjunction ”and” and conjuct “said” are attached to the root action, “pointed.”

Possessive “my” refers to objective of preposition “sisters”.

Harder Problem: Detecting Co-Reference



Named Entity Recognition:

1 At the W party Thursday night at Chateau Marmont, Cate Blanchett barely made it up in the elevator.

Basic Dependencies:

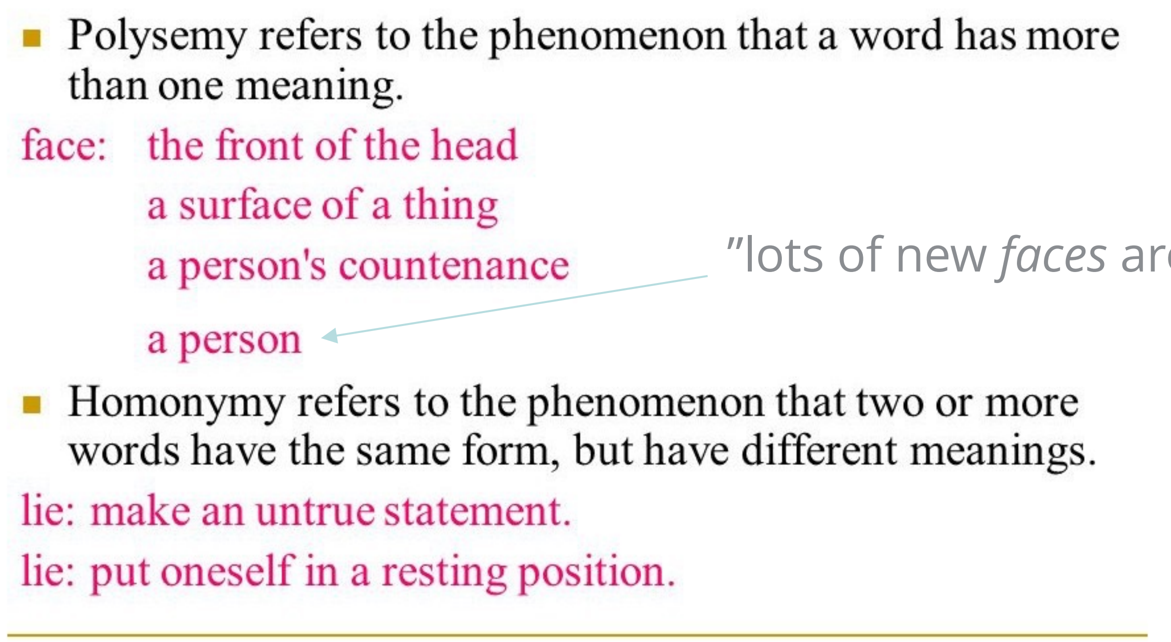
1 At the W party Thursday night at Chateau Marmont, Cate Blanchett barely made it up in the elevator.

1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.

Coreference:

1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.

Major Challenges in NLP: Polysemy and Homonymy



- Polysemy refers to the phenomenon that a word has more than one meaning.

face: the front of the head
a surface of a thing
a person's countenance
a person

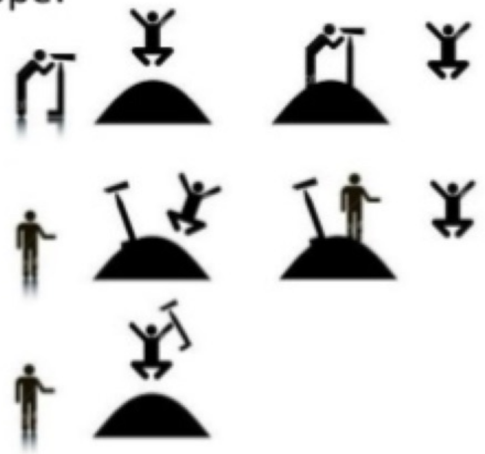
"lots of new *faces* around here!"

- Homonymy refers to the phenomenon that two or more words have the same form, but have different meanings.

lie: make an untrue statement.
lie: put oneself in a resting position.

Major Challenges in NLP: Preposition Attachment

I saw the man on the hill with a telescope.



1. I saw the man. The man was on the hill. I was using a telescope.
2. I saw the man. I was on the hill. I was using a telescope.
3. I saw the man. The man was on the hill. The hill had a telescope.
4. I saw the man. I was on the hill. The hill had a telescope.
5. I saw the man. The man was on the hill. I saw him using a telescope.

Elliptical Construction

- Elliptical construction occurs when words are left out because they are generally understood to be there. Several types:

Type	Example
Verb Gapping	Jeff can play the bass, and Larry . . . the guitar.
Verb Phrase	Jeff wanted to buy the instrument and did . . .
Nominal	Larry played one song, then Jeff played two . . .
	. . . and many more!

- Parsers have difficulty inferring the missing phrases, making later steps (like dependency or co-reference) more difficult/less accurate.

Philosophical Pitfalls:

Semantics Versus Pragmatics

- Authentic human communication works in an interpersonal context of shared meanings, personal histories, and prevailing circumstances
- In this perspective, many texts, on their own, do not contain sufficient information to truly convey the underlying meaning of the utterances
- Non-propositional speech acts (when captured in text) lose meaning when separated from their related performances. Example: “I’ll finish the job now.”
- Give an example of an utterance whose meaning varies depending on context. . .

Worked Example #2:
Topic Modeling of Text

Topic Modeling

- Considered an *unsupervised* data/text mining technique: Does a “bottom-up” analysis of documents, topics, and words with no “criterion” (ground truth) to train against
- Extracts topics that are characterized by a combination of more frequent words
- Each document in a corpus comprises a mixture of topics

Example Using National Anthems

```
> anthems <- read.csv(‘anthems.csv')
> charVector <- anthems$Anthem
> head(charVector, 1)
```

```
[1] "Around our flag we stand united, With one wish
and one goal, A sacred oath we bestow upon it
Proclaiming loyalty for our salvation. From war
abstains only he, Who a traitor is born, He who is a
true man is not frightened, But dies a warrior to the
cause. With weapons in our hands a-brandished, We will
defend our fatherland, Our sacred rights
weÃƒÂ¢Ã¢â€šÂ¬Ã¢â€žÂ¢ll not relinquish …
```

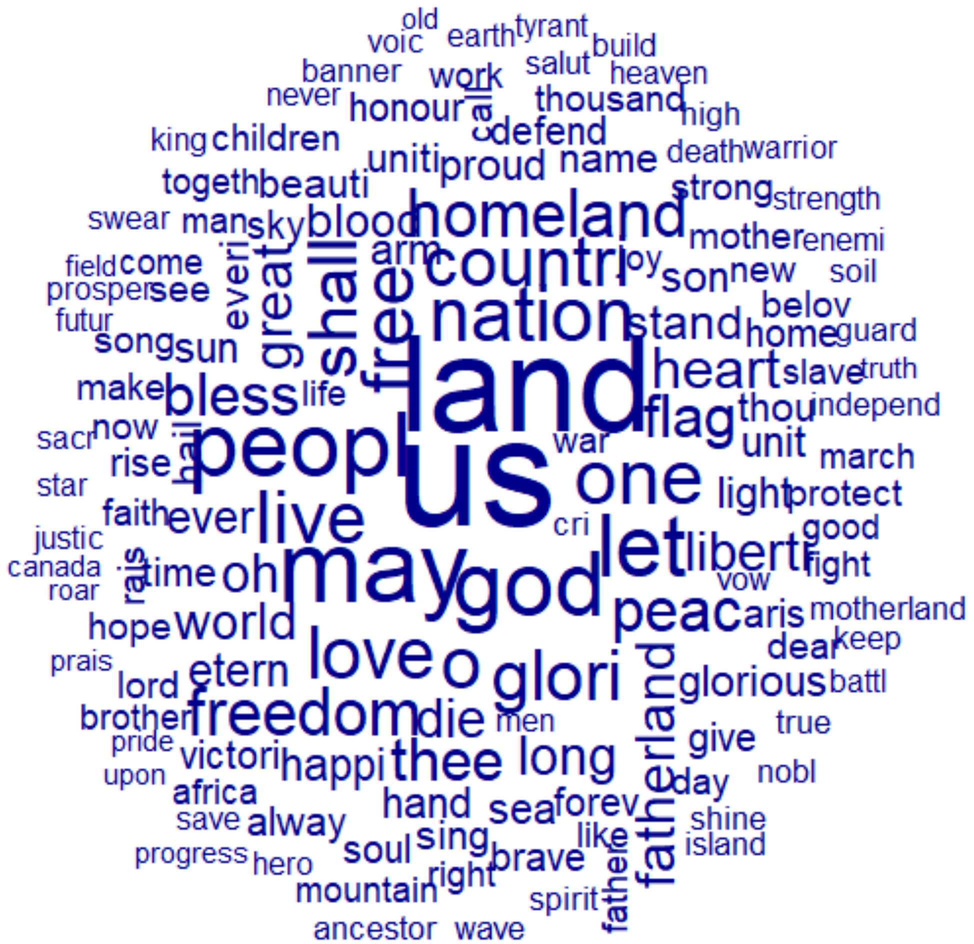
library("quanteda")

- The quanteda package was developed by Kenneth Benoit (http://kenbenoit.net) at the London School of Economics
- A comprehensive text processing package that uses some C++ and Fortran – but unlike Stanford Core NLP tools no Java; mostly runs faster and with less memory than Stanford Core NLP
- Includes readtext(), a wrapper for a wide range of text files (PDF, CSV, text, XML, JSON) that simplifies reading in folders of text files
- Directly supports document-level variables for machine learning

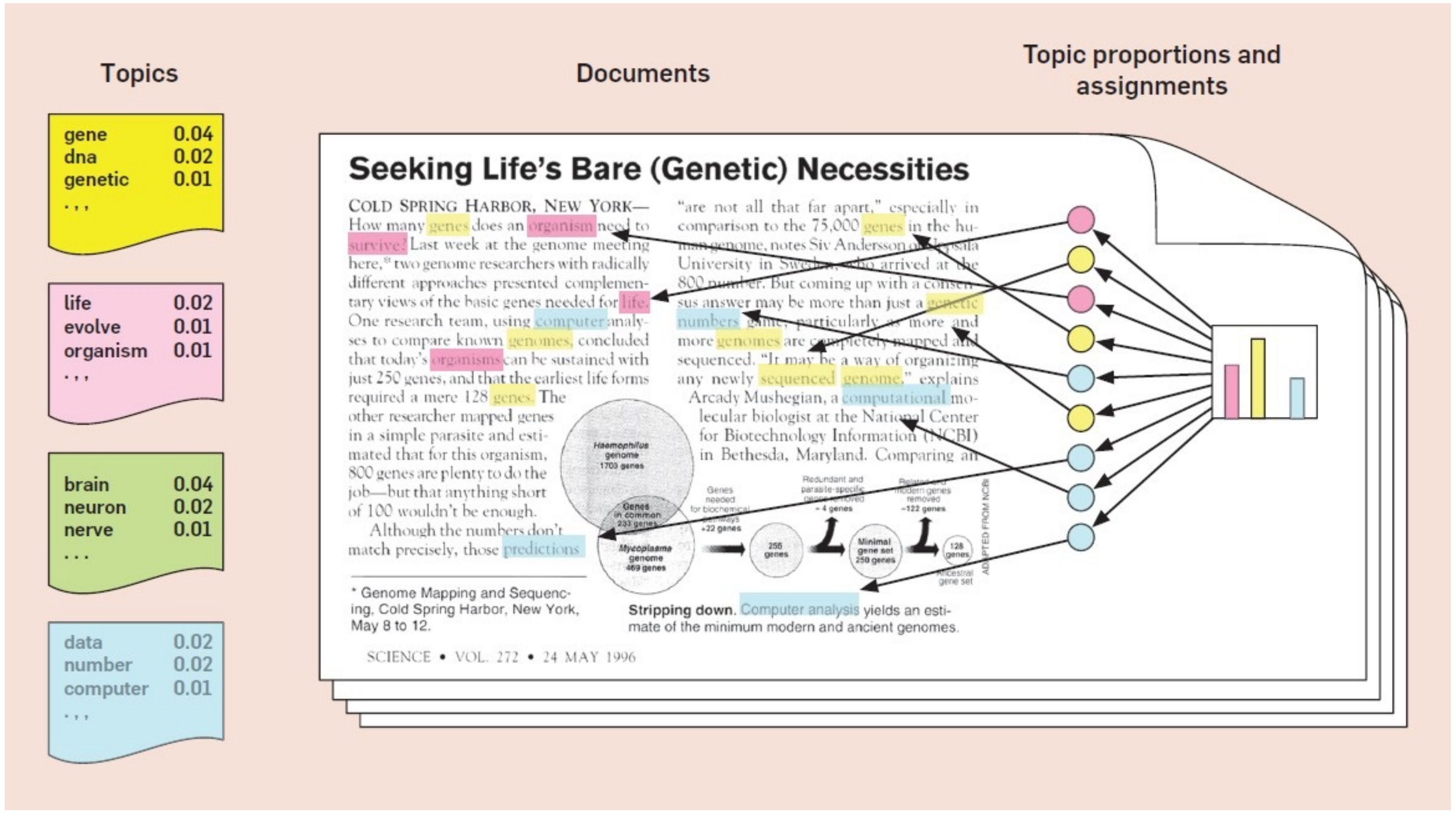
Preparing and Visualizing Corpus

```
anthemcorpus <- corpus(charVector, docnames=anthems$country)
paras <- corpus_reshape(anthemcorpus, to="paragraphs")
anthem_dtm <- dfm(paras, stem=TRUE, remove_punct=TRUE, remove_symbols =
TRUE, remove_numbers = TRUE, remove=c(stopwords("english")))
anthem_dtm <- dfm_remove(anthem_dtm, c("s","ã","â",'thi'))
anthem_dtm <- dfm_trim(anthem_dtm, min_termfreq=20)
anthem_dtm
colnames(anthem_dtm)
library("quanteda.textplots")
textplot_wordcloud(anthem_dtm) # visualize words
```

```
>textplot_wordcloud(anthem_dtm)
```



Using Latent Dirichlet Allocation to Infer Topics from Words and Documents


Topics
Documents
Topic proportions and assignments
gene 0.04
dna 0.02
genetic 0.01
. . .
life 0.02
evolve 0.01
organism 0.01
. . .
brain 0.04
neuron 0.02
nerve 0.01
. . .
data 0.02
number 0.02
computer 0.01
. . .
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

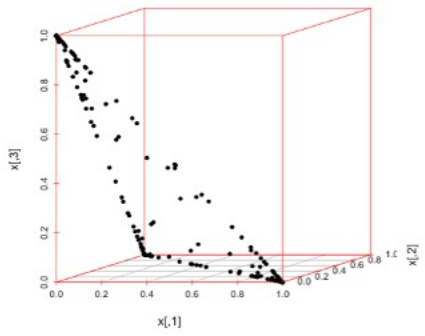
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

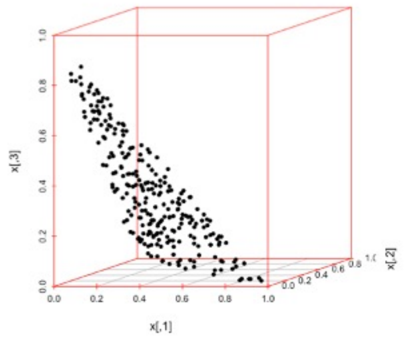
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

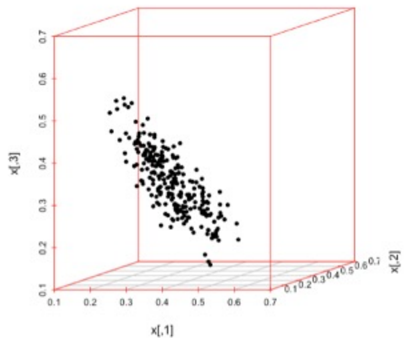
Dirichlet Distributions at Different Alpha Levels



```
# 255 docs, where each belongs to
# one or two topics
x = rdirichlet(255,
c(0.1,0.1,0.1)) # Alpha = 0.1
scatterplot3d(x,scale.y=0.5,angle
=20,col.axis="red",pch=20)
```



```
# 255 docs, with mixing of 1,2,3
# topics across docs
x = rdirichlet(255, c(1,1,1))
# Alpha = 1
scatterplot3d(x,scale.y=0.5,angle
=20,col.axis="red",pch=20)
```

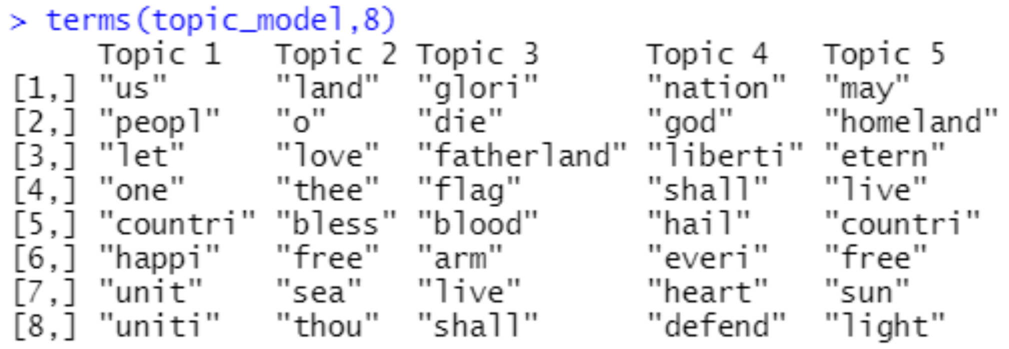


```
# 255 docs, all docs include
# mix of all three topics
x = rdirichlet(255, c(10,10,10))
# Alpha = 10
scatterplot3d(x,scale.y=0.5,angle
=20,col.axis="red",pch=20)
```

LDA is a Bayesian technique, guided by two hyperparameters of the Dirichlet distribution, alpha and beta. Low “prior” alpha value has each document composed of only a few key topics. (good separation) Low “prior” beta has each topic composed of only a few key words. (more dense)

Estimate LDA Model

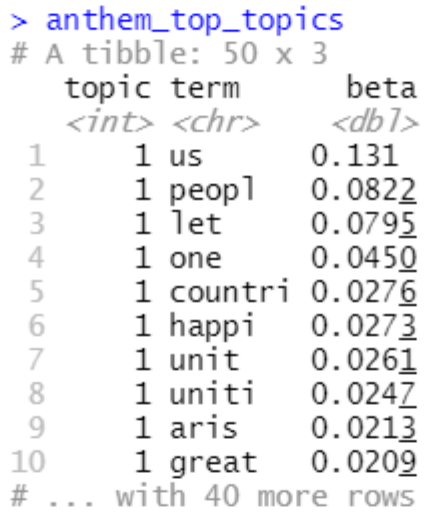
```
# topic model with 5 total topics (more on how to choose the number of topics)
topic_model <- LDA(anthem_topics, method = "VEM", k=5)  # Latent Dirichlet Allocation
termsOut<- terms(topic_model, 8) # Descriptive stems for each topic
```



```
> terms(topic_model,8)
     Topic 1   Topic 2 Topic 3       Topic 4   Topic 5
[1,] "us"      "land"  "glori"       "nation"  "may"
[2,] "peopl"   "o"     "die"         "god"     "homeland"
[3,] "let"     "love"  "fatherland"  "liberti" "etern"
[4,] "one"     "thee"  "flag"        "shall"   "live"
[5,] "countri" "bless" "blood"       "hail"    "countri"
[6,] "happi"   "free"  "arm"         "everi"   "free"
[7,] "unit"    "sea"   "live"        "heart"   "sun"
[8,] "uniti"   "thou"  "shall"       "defend"  "light"
```

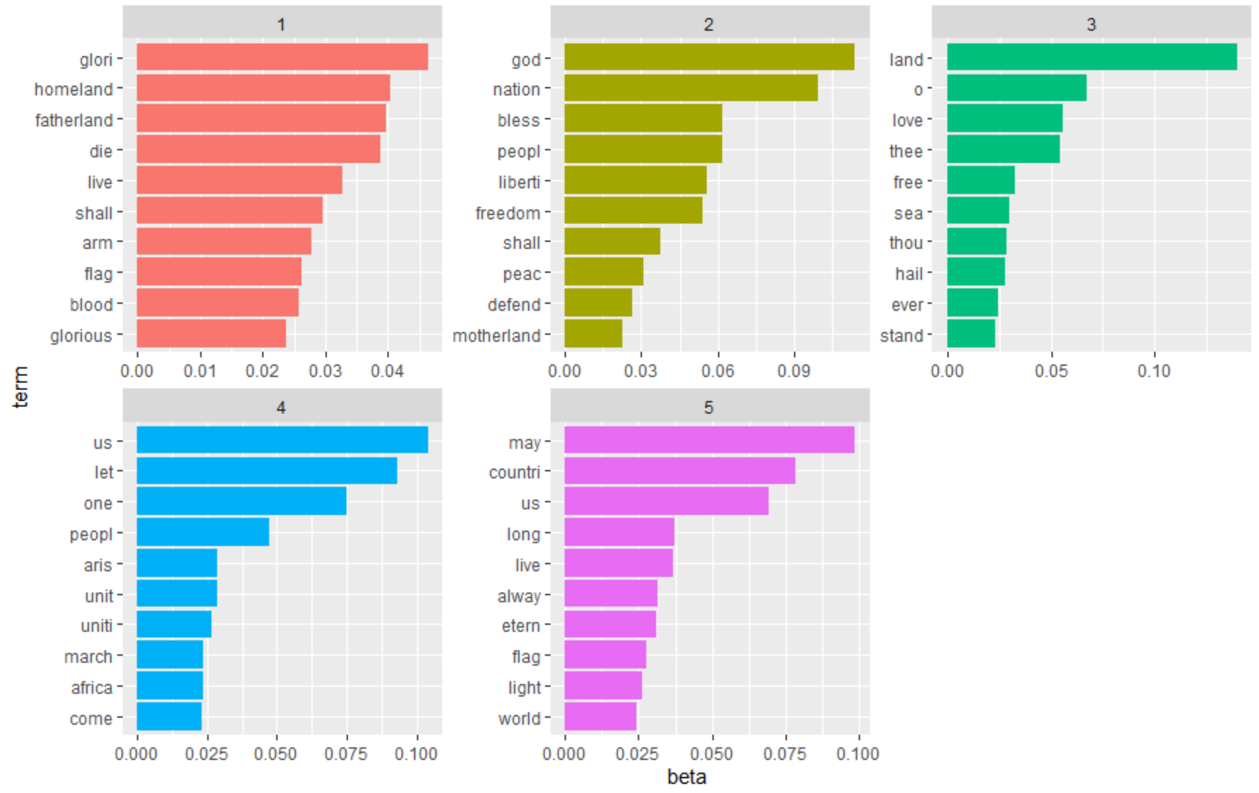
Examine beta: The per-topic-per-word probability

```
library(tidytext) # All three packages: “tidyverse”
library(ggplot2) # by Hadley Wickham library(tidyverse)

# A custom “tidier” for LDA output!

tidyTopics <- tidy(topic_model, matrix="beta")

# The %>% is “pipe” notation
anthem_top_topics <- tidyTopics %>%
 group_by(topic) %>%
 top_n(10, beta) %>%
 ungroup() %>%
 arrange(topic, -beta)
```



```
> anthem_top_topics
# A tibble: 50 x 3
   topic term      beta
   <int> <chr>    <dbl>
 1     1 us      0.131
 2     1 peopl   0.0822
 3     1 let     0.0795
 4     1 one     0.0450
 5     1 countri 0.0276
 6     1 happi   0.0273
 7     1 unit    0.0261
 8     1 uniti   0.0247
 9     1 aris    0.0213
10     1 great   0.0209
# ... with 40 more rows
```

Use ggplot to visualize topics and terms



Beta is the probability that a given term appears in a particular topic. Higher probability terms "define" the topic best.

Note: your results will vary, but should reach to similar conclusions.

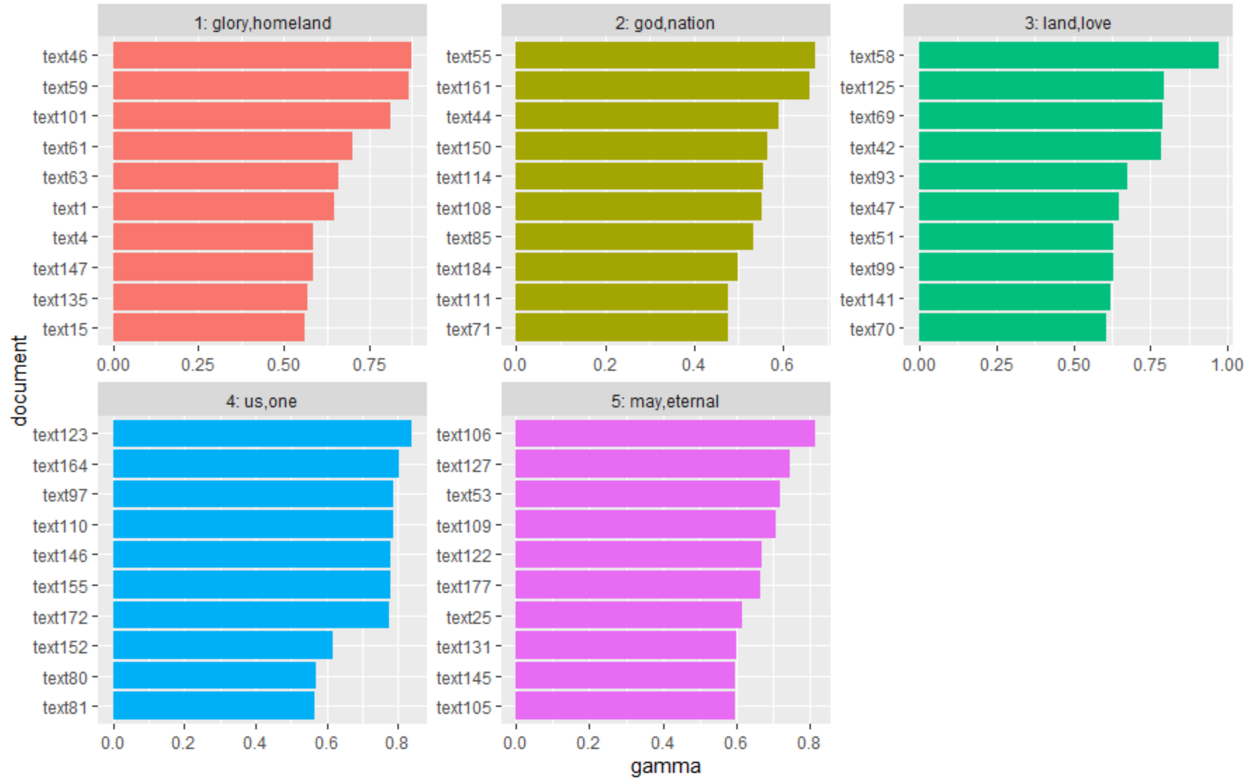
```
anthem_top_topics %>%
 mutate(term = reorder(term, beta)) %>%
 ggplot(aes(term, beta, fill = factor(topic)))
 + geom_col(show.legend = FALSE) +
 facet_wrap (~ topic, scales = "free") +
 coord_flip()
```

Try to name topics?

- Topic 1 top keywords: glory, homeland, fatherland
- My title for topic 1:

  Glory to homeland

Use ggplot to visualize documents and topics



Gamma is the probability that a given topic appears in a particular document. Higher probability documents show the main topics.

```
anthem_top_topics %>%
 mutate(document = reorder(document, gamma)) %>%
 ggplot(aes(document, gamma, fill = factor(topic))) +
 geom_col(show.legend = FALSE) +
 facet_wrap (~ topic, scales = "free") +
 coord_flip()
```

Try to name topics?

- Topic 1 appears strongly in 3 anthems:
    - What countries are they?: Bolivia, Mexico, Iraq

Of the homeland, the lofty name. In glorious splendour may we keep [it] And in its honour let's swear once again: To die before living as slaves!

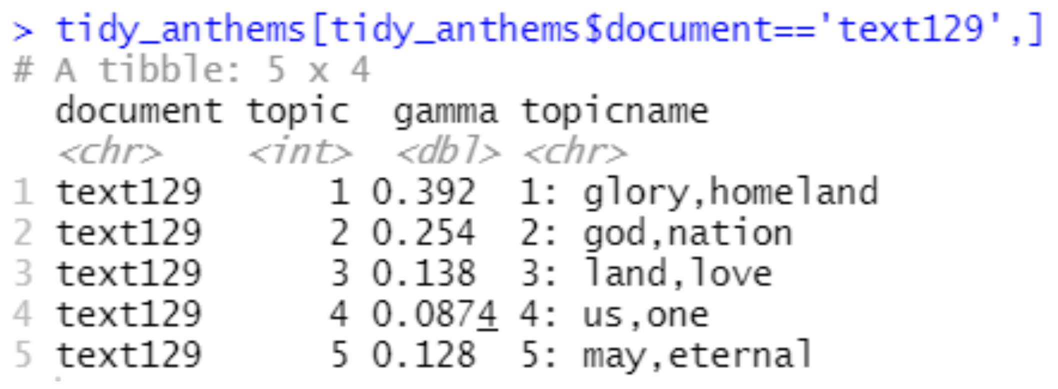
Try to name topics?

- Topic 3?: *Land of love*
  - Sri Lanka:

    Plenteous in prosperity, Thou, Beauteous in grace and love, Laden with grain and luscious fruit, And fragrant flowers of radiant hue, Giver of life and all good things, Our land of joy and victory, Receive our grateful praise sublime, we worship, worship Thee.

Use ggplot to visualize documents and topics

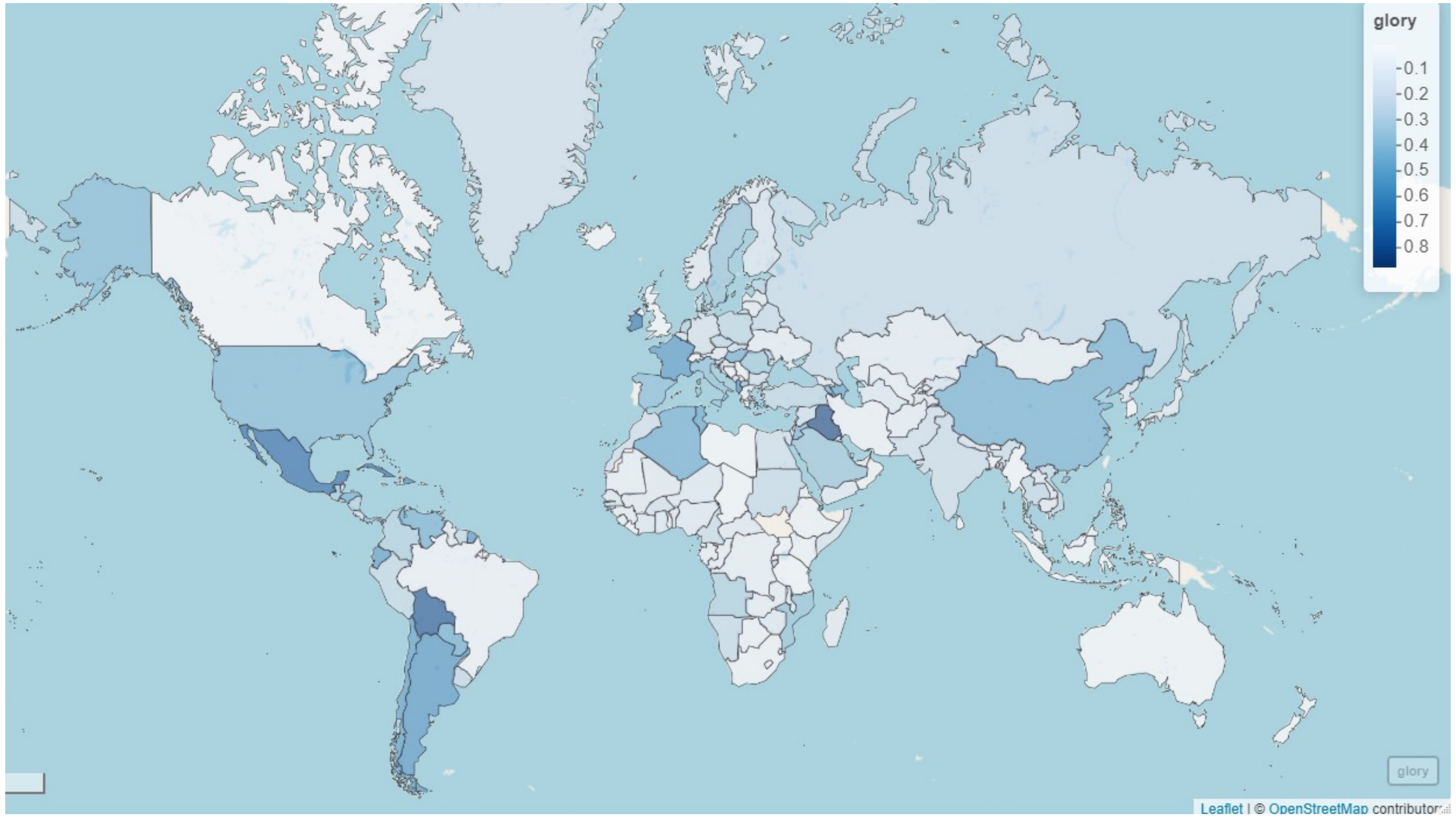
- Note: For a given country, probabilities add up to 1:



```
> tidy_anthems[tidy_anthems$document=='text129',]
# A tibble: 5 x 4
  document topic  gamma topicname
  <chr>    <int>  <dbl> <chr>
1 text129      1 0.392  1: glory,homeland
2 text129      2 0.254  2: god,nation
3 text129      3 0.138  3: land,love
4 text129      4 0.0874 4: us,one
5 text129      5 0.128  5: may,eternal
```

Try mapview()!!

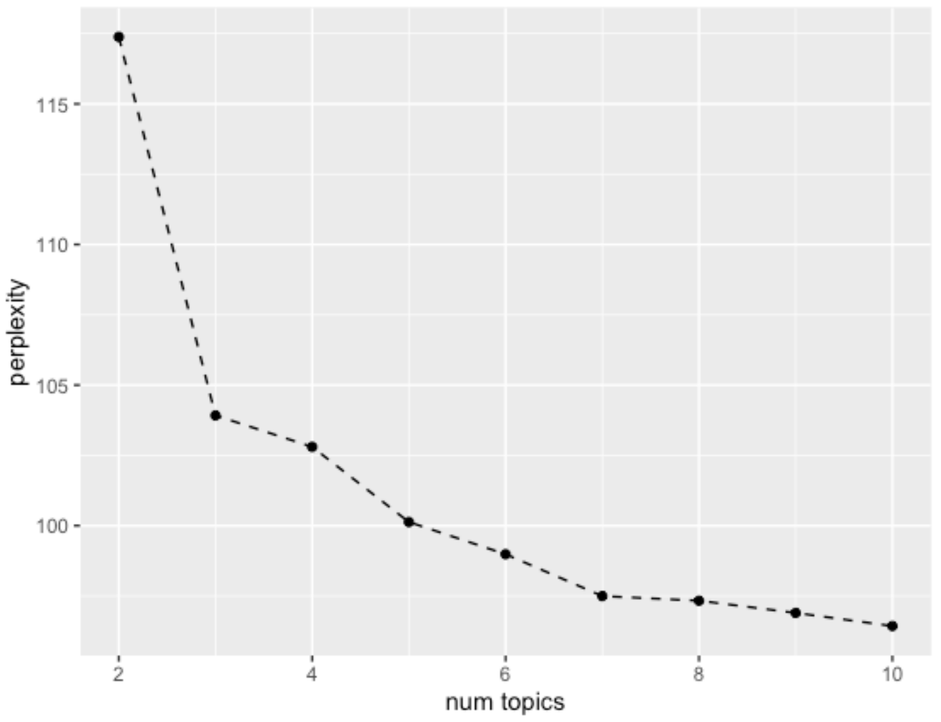
```
library(mapview)
mapview(out,zcol='gamma', col.regions = blues9,
label=paste(out$CountryCode ,out$gamma),layer.name ='glory')
```



Well.. with some cleaning

```
tidy_anthems <- tidy_anthems %>%
mutate(countryid=as.numeric(str_extract(tidy_anthems$document, "[0-9]+"))) # need the codes
anthems <- anthems %>%
  mutate(countryid = factor(rownames(anthems)))

df <- merge(tidy_anthems,anthems,on='countryid’)
#######################################
#install.packages("rnaturalearth")
#install.packages("rnaturalearthdata")

library("rnaturalearth")
library("rnaturalearthdata")

world <- ne_countries(scale = "medium", returnclass = "sf")
world$CountryCode = world$su_a3
df$CountryCode = df$Alpha.3

out <- merge(df, world, on = 'CountryCode')
out <- out %>% filter(topic==1) %>% mutate(gamma=round(gamma,2))
# convert to spatial object
out <- st_as_sf(out)

library(mapview)
mapview(out,zcol='gamma', col.regions = blues9, label=paste(out$CountryCode
,out$gamma),layer.name ='glory')
```

Use Perplexity to Determine Number of Topics



```
library("topicmodels")
interview_topics <- convert(anthem_dtm, to = "topicmodels")
maxTopics <- 10 # Max number of topics we will allow to form
set.seed(1)
perList <- NULL # We are going to make a list of perplexity values
# Loop: Try every number of topics from 2 up to maxTopics
for (i in 2:maxTopics) {
 topic_model <- LDA(anthem_topics, method = "VEM", k=i) # Latent Dirichlet
 Allocation
 perList[i-1] <- perplexity(topic_model)
}
names(perList) <- as.character(2:maxTopics)
perList_df <- as.data.frame(perList)
library(ggplot2)
ggplot(data=perList_df,aes(x=as.numeric(rownames(perList_df)),
y=perList)) + geom_point() + geom_line(linetype='dashed') +
labs(x='num topics',y='perplexity')
```

This is an elbow plot!

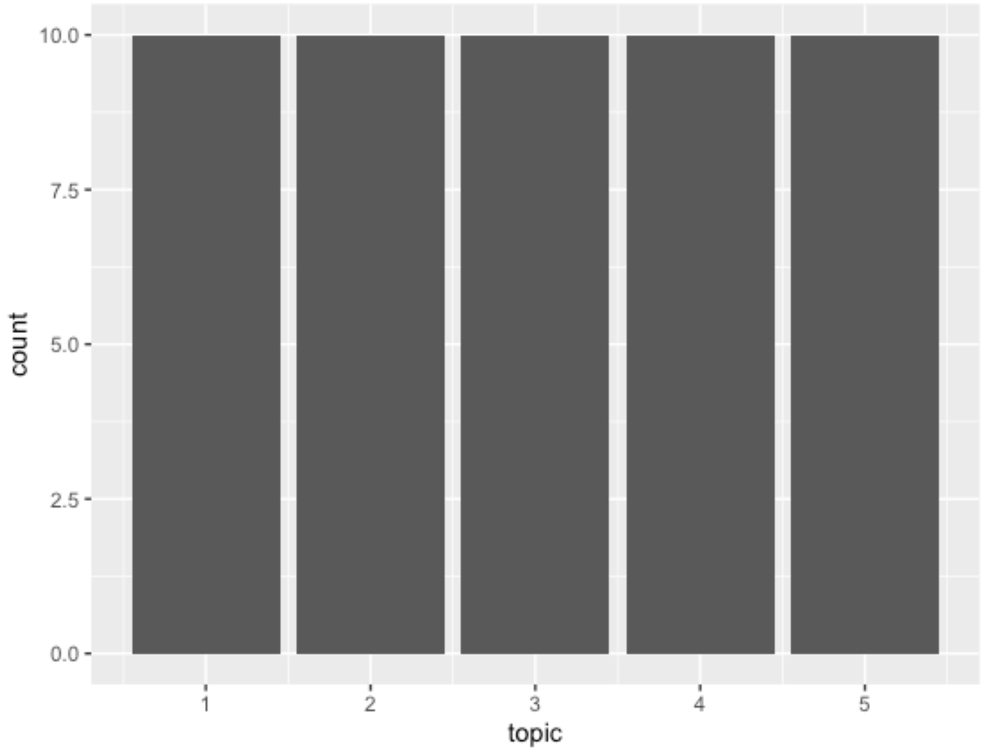
Exercise: re-do the analysis with optimal k

How Many Documents for Each Topic?

```
> top_anthems %>% group_by(topic) %>%
summarize(count=n(),av_gamma = mean(gamma))
# A tibble: 5 × 3
  topic count av_gamma
  <int> <int>    <dbl>
1     1    10    0.677
2     2    10    0.608
3     3    10    0.615
4     4    10    0.685
5     5    10    0.661
```

Ideally, the documents should be evenly distributed across topics.

```
top_anthems %>% group_by(topic) %>%
summarize(count=n(),av_gamma = mean(gamma)) %>%
ggplot(.,aes(x=topic,y=count)) + geom_col()
```



Exercise: Repeat analysis with nomination speech

- Conduct topic analysis using the nominee text that you used for the sentiment analysis exercise
- Use paragraph numbers instead of descriptive titles for each of the paragraphs to use in place of anthems
- Generate LDA topic models and examine perplexity
- Visualize the topics and paragraphs and draw conclusions