Samruddhi Somani
Julia Wu
Jace Burton

MODULE 17: TEXT DATA

1. **Vocabulary**
2. **Importance**
3. **Representing Documents**
4. **Vector-Space/Bag of Words**

1. Vocabulary:

Corpus/Corpora:	Body of documents
Dictionary:	Set of all possible words for a particular corpus
Grok:	"Getting" it (Computers have trouble 'grokking' text documents)
n-gram:	Set of 'n' words in a row
NLP:	Natural Language Processing
Text Pipelines:	How unstructured text data becomes tidy structured data
Token:	String with an identified meaning
Tokenization:	Turning a string of symbols into a string of tokens
XML:	Text markup language—includes document metadata

2. NLP allows us to automate tasks like the following.

Information Retrieval

1. Find me similar documents
2. Find me documents about this topic
3. What is this about?

Classifying

1. Author attribution
2. Document grouping

3. Representing Documents: Representation is just as, if not more, important than subsequent analysis.

Text does not come structured, so we need to think about the below to get it in a structured form, which may look different for different types of analyses. We often only consider metadata and words in document, which leaves information on the table but still works well for many applications.

- Metadata—facts about the document (author/file type)
- Words in document
- Sentiment
- Main ideas/topics
- Meaning
- Grammar/syntax
- Language: Documents in different languages may convey different things in translation.

4. Bag of words: A list of distinct words in the document and a count for each word

- Purely textual features
- Easy to calculate/work with
- Data Structures

   i. Hash Table ("Python dictionary"): Key value store with keys: words; values: counts

ii. Vector: each component is one word, each entry is count for that word
This representation is what people usually mean when they say 'bag of words'
    - EG: Me Cookie Monster. Me want cookie.
      (Me Existentialist Cookie)
      (2 0 2)
    - Vector representation is common because there is a history of usage so we know how to think about vectors

- Document term matrix:
  i. N=number of documents; D=number of words in dictionary; x_{ij}=count for word *j* in document *i.*
  ii. $X = \begin{matrix} x_{11} & x_{12} \ldots & x_{1D} \\ \vdots & & \\ x_{n1} & x_{n2} \ldots & x_{nD} \end{matrix}$
- Tokenization: Turn a string of symbols into a string of tokens.
  i. Examples: cookie, :-), quarter back
  ii. Potential steps: Lots of choices to be made ('One person's trash is another's treasure')
    - Remove/split white spaces (doesn't work for E.E. Cummings)
    - Remove punctuation (doesn't work for emoticons)
    - Convert to lowercase
    - Remove stop words—grammatical filler—conjunctions, prepositions, articles
    - Drop numbers or combine into one token
    - Stemming: Breaking words into "roots" and "suffixes"
      Each word can only have one stem, e.g. taxing cannot be represented by 'tax' ***and*** *'hard'.*
        - Tax← taxes, taxing, taxation
        - Action← actionable

5. R Script

Load in the xml library and source in helper functions.

```
library(XML)

# Some helper functions'
source('textutils.R')
```

Read in the stories and split on spaces. Examine the structure of the directory. This is organized as a list of lists where the inner list is all the words in the document in order.

```
art_stories = read.directory('./nyt_corpus/art/')
str(art_stories)

## List of 57
##  $ : chr [1:2043] "lead" "artists" "thinking" "about" ...
##  $ : chr [1:471] "lead" "the" "first" "major" ...
##  $ : chr [1:1225] "lead" "to" "see" "one" ...
##  $ : chr [1:1393] "lead" "with" "daring" "and" ...
##  $ : chr [1:1148] "lead" "from" "jailhouse" "to" ...
##  $ : chr [1:1113] "lead" "lorna" "simpsons" "conceptual" ...
##  $ : chr [1:2952] "the" "art" "dealer" "larry" ...
##  $ : chr [1:158] "sport" "and" "art" "have" ...
##  $ : chr [1:468] "stuart" "diamond" "mckee" "gallery" ...
##  $ : chr [1:1174] "visitors" "to" "davison" "art" ...
##  $ : chr [1:920] "in" "catholicism" "at" "progressive" ...
##  $ : chr [1:275] "elizabeth" "murraypaula" "cooper" "gallery" ...
##  $ : chr [1:1140] "a" "leading" "picasso" "scholar" ...
##  $ : chr [1:733] "after" "a" "walk" "through" ...
##  $ : chr [1:190] "fred" "tomaselli" "jack" "tilton" ...
##  $ : chr [1:78] "an" "article" "in" "the" ...
##  $ : chr [1:292] "laura" "newman" "tenri" "cultural" ...
##  $ : chr [1:710] "mellie" "cooper" "was" "getting" ...
##  $ : chr [1:293] "here" "is" "a" "sampling" ...
##  $ : chr [1:1191] "in" "a" "china" "that" ...
##  $ : chr [1:202] "hows" "this" "for" "an" ...
##  $ : chr [1:247] "ilan" "averbuch" "nancy" "hoffman" ...
##  $ : chr [1:1076] "eight" "year" "old" "molly" ...
##  $ : chr [1:475] "news" "days" "of" "hopeas" ...
##  $ : chr [1:214] "brian" "tolle" "common" "consent" ...
##  $ : chr [1:264] "doug" "aitken" "#" "gallery" ...
##  $ : chr [1:1172] "the" "painter" "j" "alden" ...
##  $ : chr [1:327] "lombard" "freid" "fine" "arts" ...
##  $ : chr [1:1131] "push" "through" "the" "black" ...
##  $ : chr [1:200] "xl" "xavier" "laboulbenne" "gallery" ...
##  $ : chr [1:294] "behind" "honore" "daumier" "lies" ...
##  $ : chr [1:1429] "on" "saturday" "the" "rothko" ...
##  $ : chr [1:1197] "reality" "is" "tricky" "business" ...
##  $ : chr [1:370] "newsneedleworkisnt" "that" "jimmy" "carter" ...
##  $ : chr [1:1876] "for" "some" "in" "hollywood" ...
##  $ : chr [1:880] "when" "jonathan" "larson" "first" ...
##  $ : chr [1:459] "when" "miguel" "oks" "encountered" ...
##  $ : chr [1:1446] "beyond" "the" "easel" "decorative" ...
##  $ : chr [1:258] "one" "of" "the" "worlds" ...
##  $ : chr [1:244] "bellwether" "#" "grand" "street" ...
##  $ : chr [1:895] "hand" "held" "computers" "have" ...
##  $ : chr [1:300] "how" "to" "turn" "work" ...
##  $ : chr [1:275] "gagosian" "gallery" "#" "west" ...
##  $ : chr [1:386] "skylight" "gallery" "#" "fulton" ...
##  $ : chr [1:266] "chambers" "fine" "art" "#" ...
##  $ : chr [1:179] "to" "the" "editor" "a" ...
##  $ : chr [1:1083] "the" "mid" "#th" "century" ...
##  $ : chr [1:1673] "red" "groomsby" "arthur" "c" ...
##  $ : chr [1:986] "a" "famed" "raphael" "will" ...
##  $ : chr [1:236] "not" "all" "art" "must" ...
##  $ : chr [1:764] "members" "of" "the" "landmarks" ...
##  $ : chr [1:514] "art" "politics" "and" "piratesron" ...
##  $ : chr [1:531] "nicole" "klagsbrun" "#" "west" ...
##  $ : chr [1:1247] "they" "were" "painted" "in" ...
##  $ : chr [1:1479] "directors" "and" "trustees" "of" ...
##  $ : chr [1:1239] "a" "lot" "of" "the" ...
##  $ : chr [1:1206] "joseph" "barbera" "an" "innovator" ...
```

Turn these stories in vectors of counts of words also called a "bag of words" (list of lists). This object lists the distinct words and their counts for each story.

```
art_stories_vec = LoW_to_countvector(art_stories)
```

Pad out each vector to include NA's for words not seen in that story but part of the corpus vocabulary. This standardizes all the vectors to the same length with the same index of words. We also remove words that only appear in one document to reduce the size of our final document-term matrix.

```
art_stories_vec_std = standardize.ragged(art_stories_vec)
art_stories_vec_std = remove.singletons.ragged(art_stories_vec_std)
#remove words that only showed up in one document
```

Turn these lists into a document-term matrix.

```
art_stories_DTM = make.BoW.frame(art_stories_vec_std)
```