Data Science for Public Policy

Aaron R. Williams - Georgetown University

# Text Analysis

## Motivation

**Text Corpus:** A set of text. A corpus generally has theme such as "The Federalist Papers" or Jane Austin novels.

**Text analysis:** The process of deriving information from text using algorithms and statistical analysis.

Text analysis has broad applications that extend well beyond policy analysis. We will briefly focus on four applications that are related to policy analysis.

### 1. Document Summarization

The process of condensing the amount of information in a document into a useful subset of information. Techniques range from counting words to using complex machine learning algorithms.

**Example:** Frederick Mosteller and David Wallace used Bayesian statistics and the frequency of certain words to identify the authorship of the twelve unclaimed Federalist papers. (blog) Before estimating any models, they spent months cutting out each word of the Federalist Papers and the counting the frequency of words.

### 2. Text Classification (supervised)

The process of labeling documents with a predetermined set of labels.

**Example:** Researchers in the Justice Policy Center at the Urban Institute classified millions of tweets with the word "cop" or "police" as "positive", "negative", "neutral", or "not applicable" for sentiment and "crime or incident information", "department or event information", "traffic or weather updates", "person identification", or "other" for topic. The researchers created content and metadata features, manually labeled a few thousand tweets for training data, and used gradient boosting for supervised machine learning for the task. (blog)

**Example:** Frederick Mosteller and David Wallace used Bayesian statistics and the frequency of certain words to identify the authorship of the twelve unclaimed Federalist papers. (blog)

### 3. Document Grouping (unsupervised)

The algorithmic grouping or clustering of documents using features extracted from the documents. This includes unsupervised classification of documents into meaningful groups. Techniques like topic modeling result in lists of important words that can be used to summarize and label the documents while techniques like K-means clustering result in arbitrary cluster labels.

**Example:** Pew Research used *unsupervised* and *semi-supervised* methods to create topic models of open-ended text responses about where Americans find meaning in their lives. (blog)

### 4. Text Extraction

Text often contains important unstructured information that would be useful to have in a structured format like a table. Text extraction is the process of searching and identifying key entities or concepts from unstructured text and then placing them in a structured format.

**Example:** Researchers from Cornell counted sources of misinformation from 38 million articles. NYTimes. Methodology.

### Other

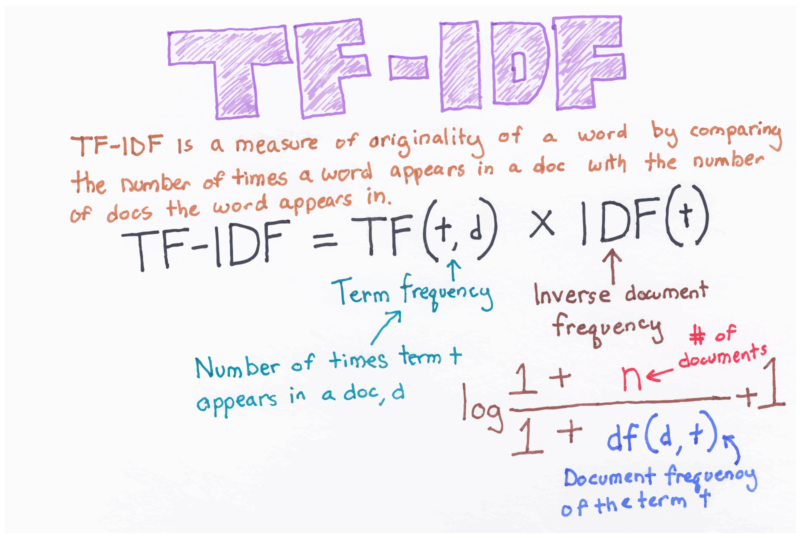
Speech recognition, machine translation, question answering, and text autocompletion are other forms of text analysis and language processing that are common but not implemented with R packages.

## Tools

### Frequency

**Term Frequency:** A count or relative frequency of the most common words in a document or documents.

**Term Frequency-Inverse Document Frequency (TF-IDF):** Some words are naturally more common than other words. TF-IDF quantifies the importance of words/terms in one document relative to other documents in a corpus.


TF-IDF
TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.
TF-IDF = TF(t, d) × IDF(t)
Term frequency
Number of times term t appears in a doc, d
Inverse document frequency
# of documents
log (1 + n) / (1 + df(d, t)) + 1
Document frequency of the term t

**Source:** Chris Albon

**Word cloud:** A method of visualizing word frequency where the frequency of a given word is encoded as the size of the word. Word clouds are overused and are tough to interpret. This 2016 election word cloud by Gallup is an exceptional word cloud.

## Collocation

**N-gram:** A consecutive sequence of $n$ words. Unigrams are words, bigrams are pairs of words, and trigrams are groups of three words.

**N-gram frequency:** A count or relative frequency of n-grams in a document or documents.

**TF-IDF of N-grams:** TF-IDF with n-grams instead of words.

**Bigram graph:** A directed or undirected graph with individual words as nodes and bigrams as edges. Example

## Topic Modeling

**Topic modeling:** Unsupervised and semi-supervised methods for grouping documents. Example

Popular methods include:

- Latent Dirichlet Allocation (LDA)
- Non-Negative Matrix Factorization
- CorEx (Correlation Explanation)

### Natural Language Processing

**Natural language processing:** The algorithmic analysis of text in a way that is rooted in linguistics. Stanford CoreNLP is an important tool for natural language processing.

**Sentiment analysis:** The process of labeling words or sentences as positive, neutral, or negative.

**Named entity recognition:** The process of identifying names, places, organizations, and more in text.

**Parts of speech tagging:** The process of identifying the part of speech of each word (noun, verb, etc.).

**Lemmatization:** The process of shortening words to a base grammar.

## Key Challenge

`R` and `Python` have really powerful tools for text analysis.

**Key challenge:** text contains a lot of useful information but its structure does not match how we have done data analysis up to this point.

## Vocabulary & Tidy Data

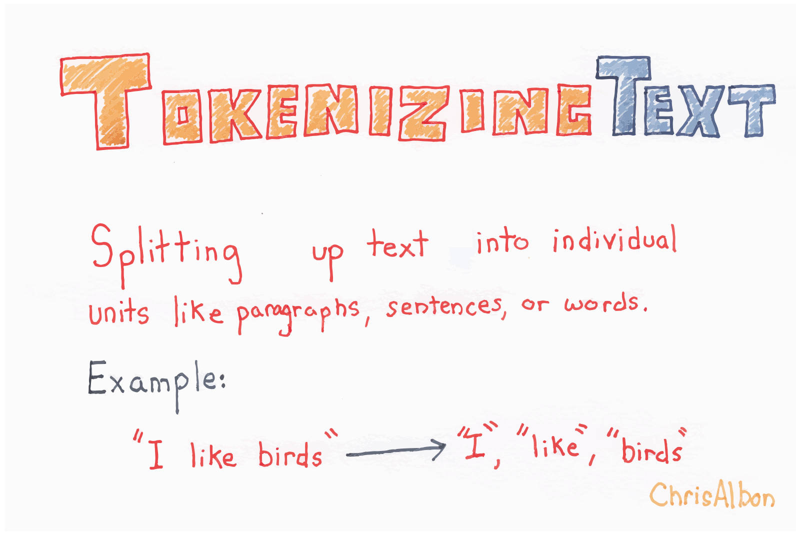
```
library(gutenbergr)

fed_papers <- gutenberg_download(gutenberg_id = 1404)

fed_papers
```

```
## # A tibble: 19,359 x 2
##    gutenberg_id text
##           <int> <chr>
##  1         1404 "THE FEDERALIST PAPERS"
##  2         1404 ""
##  3         1404 "By Alexander Hamilton, John Jay, and James Madison"
##  4         1404 ""
##  5         1404 ""
##  6         1404 ""
##  7         1404 ""
##  8         1404 "FEDERALIST No. 1"
##  9         1404 ""
## 10         1404 "General Introduction"
## # ... with 19,349 more rows
```

```
fed_paper1 <- fed_papers %>%
  filter(row_number() >= 8 & row_number() <= 165)
```

**Token:** A meaningful unit of text.

**Tokenization:** Process of splitting a larger unit of text into tokens.


TOKENIZING TEXT
Splitting up text into individual units like paragraphs, sentences, or words.
Example:
"I like birds" → "I", "like", "birds"
ChrisAlbon

**Source:** Chris Albon

```
# tidytext can tokenize text with unnest_tokens()
tidy_fed_paper1 <- fed_paper1 %>%
  unnest_tokens(output = word, input = text)

tidy_fed_paper1
```

```
## # A tibble: 1,612 x 2
##    gutenberg_id word
##           <int> <chr>
##  1         1404 federalist
##  2         1404 no
##  3         1404 1
##  4         1404 general
##  5         1404 introduction
##  6         1404 for
##  7         1404 the
##  8         1404 independent
##  9         1404 journal
## 10         1404 saturday
## # ... with 1,602 more rows
```

**Stemming:** A method of removing the end, and keeping only the root, of a word. Stemming is unaware of the context or use of the word. Example

```
# SnowballC has a stemmer that works well with tidytext
library(SnowballC)

tidy_fed_paper1 %>%
```

```r
  mutate(stem = wordStem(word)) %>%
  filter(word != stem)
```

```
## # A tibble: 536 x 3
##    gutenberg_id word         stem
##           <int> <chr>        <chr>
##  1         1404 general      gener
##  2         1404 introduction introduct
##  3         1404 independent  independ
##  4         1404 saturday     saturdai
##  5         1404 october      octob
##  6         1404 people       peopl
##  7         1404 unequivocal  unequivoc
##  8         1404 experience   experi
##  9         1404 inefficacy   inefficaci
## 10         1404 subsisting   subsist
## # ... with 526 more rows
```

**Lemmatizing:** A method of returning the base of a word. Lemmatization considers the context of a word. Example Lemmatizing requires natural language processing, so this requires Stanford CoreNLP or the Python package spaCy.

**Stop words:** Extremely common words that are often not useful for text analysis. `library(tidytext)` contains stop words from the `onix`, `SMART`, and `snowball` lexicons.

```r
stop_words
```

```
## # A tibble: 1,149 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 a           SMART
##  2 a's         SMART
##  3 able        SMART
##  4 about       SMART
##  5 above       SMART
##  6 according   SMART
##  7 accordingly SMART
##  8 across      SMART
##  9 actually    SMART
## 10 after       SMART
## # ... with 1,139 more rows
```

**Tidy text:** "A table with one token per row." ~ Text Mining with R

Text can also be stored as a string, a corpus, and a document-term matrix.

```r
# tidy text format
tidy_fed_paper1
```

```
## # A tibble: 1,612 x 2
##    gutenberg_id word
##           <int> <chr>
##  1         1404 federalist
##  2         1404 no
##  3         1404 1
##  4         1404 general
##  5         1404 introduction
##  6         1404 for
##  7         1404 the
##  8         1404 independent
##  9         1404 journal
## 10         1404 saturday
## # ... with 1,602 more rows
```

## Example 1

```
# load necessary packages
library(tidyverse)
library(tidytext)
library(gutenbergr)
library(SnowballC)

# download the Federalist Papers from Project Gutenberg
fed_papers <- gutenberg_download(gutenberg_id = 1404)

# this data cleaning comes from Emil Hvitfeldt
# https://www.hvitfeldt.me/blog/predicting-authorship-in-the-federalist-papers-with-tidytext/
hamilton <- c(1, 6:9, 11:13, 15:17, 21:36, 59:61, 65:85)
madison <- c(10, 14, 18:20, 37:48)
jay <- c(2:5, 64)
unknown <- c(49:58, 62:63)

fed_papers <- pull(fed_papers, text) %>%
  str_c(collapse = " ") %>%
  str_split(pattern = "\\.|\\?|\\!") %>%
  unlist() %>%
  tibble(text = .) %>%
  mutate(sentence = row_number())

tidy_fed_papers <- fed_papers %>%
  mutate(paper_number = cumsum(str_detect(text, regex("FEDERALIST No",
                                          ignore_case = TRUE)))) %>%
  unnest_tokens(word, text) %>%
  mutate(author = case_when(
    paper_number %in% hamilton ~ "hamilton",
    paper_number %in% madison ~ "madison",
    paper_number %in% jay ~ "jay",
    paper_number %in% unknown ~ "unknown"
  ))
```

### Approach 1

Here we'll calculate term frequency without stop words and with stemming.

```r
# filter to main authors
tidy_fed_papers1 <- tidy_fed_papers %>%
  filter(author %in% c("hamilton", "madison"))

# remove stop words and stem the words
tidy_fed_papers1 <- tidy_fed_papers1 %>%
  anti_join(stop_words, by = "word") %>%
  mutate(word = wordStem(word))

# count Hamilton's most-frequent words
tidy_fed_papers1 %>%
  filter(author == "hamilton") %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 4,073 x 2
##    word          n
##    <chr>     <int>
##  1 power       545
##  2 govern      519
##  3 constitut   372
##  4 nation      370
##  5 peopl       272
##  6 author      251
##  7 union       251
##  8 law         236
##  9 object      223
## 10 court       215
## # ... with 4,063 more rows
```

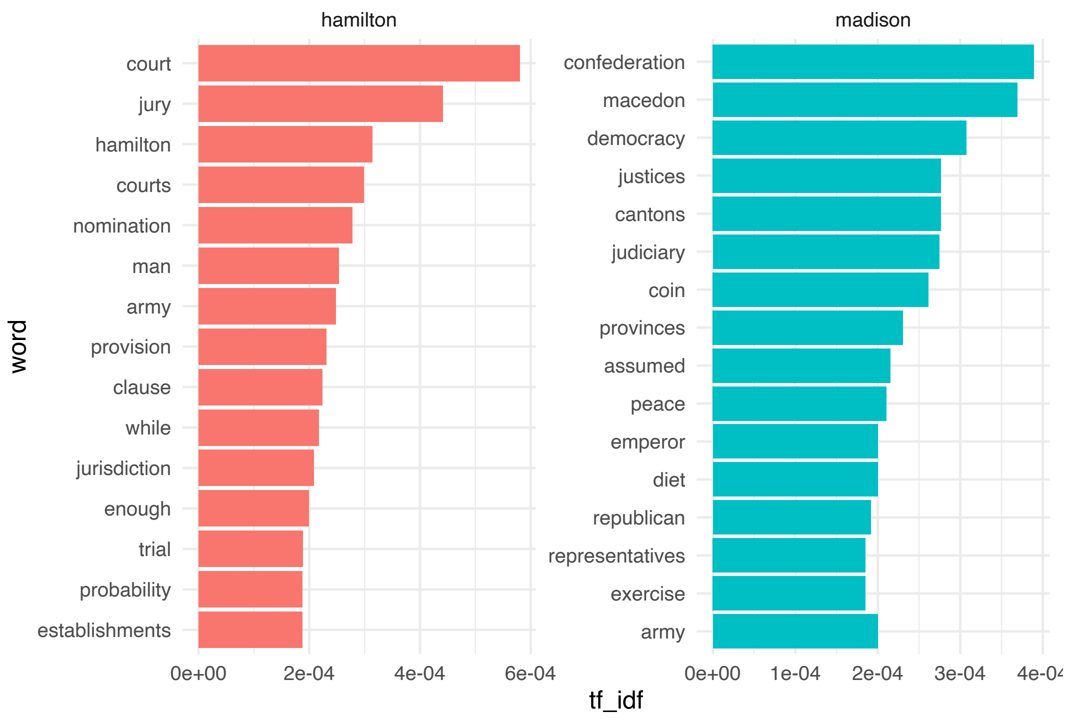
```r
# count Madison's most-frequent words
tidy_fed_papers1 %>%
  filter(author == "madison") %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 2,863 x 2
##    word          n
##    <chr>     <int>
##  1 govern      318
##  2 power       311
##  3 constitut   213
##  4 peopl       144
##  5 author      125
##  6 execut      120
##  7 feder       118
##  8 form         99
##  9 depart       98
## 10 union        97
## # ... with 2,853 more rows
```

## Approach 2

Here we'll perform TF-IDF.

```
# calculate tf-idf
tf_idf <- tidy_fed_papers %>%
  count(author, word, sort = TRUE) %>%
  bind_tf_idf(term = word, document = author, n = n)

# plot
tf_idf %>%
  filter(author %in% c("hamilton", "madison")) %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(tf_idf, word, fill = author)) +
  geom_col() +
  facet_wrap(~author, scales = "free") +
  theme_minimal() +
  guides(fill = "none")
```



## Approach 3

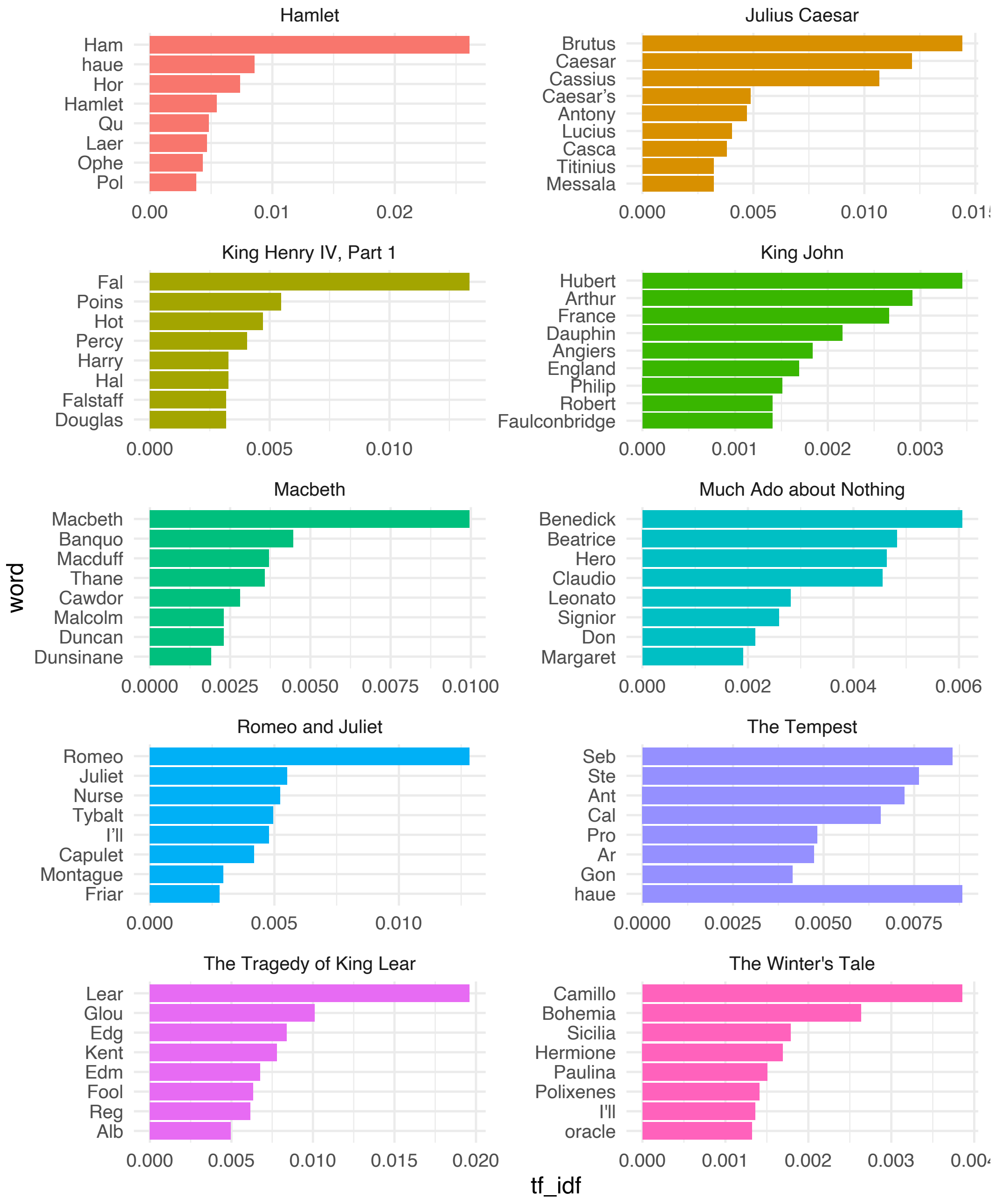
```
tidy_fed_papers %>%
  count(author, word, sort = TRUE) %>%
  filter(word == "upon")
```

```
## # A tibble: 4 x 3
##   author   word      n
##   <chr>    <chr> <int>
## 1 hamilton upon    374
## 2 madison  upon      9
## 3 unknown  upon      3
## 4 jay      upon      1
```

## Example 2

Let's consider ten of Shakespeare's plays.

```r
ids <- c(
  2265, # Hamlet
  1795, # Macbeth
  1522, # Julius Caesar
  2235, # The Tempest
  1780, # 1 Henry IV
  1532, # King Lear
  1513, # Romeo and Juliet
  1110, # King John
  1519, # Much Ado About Nothing
  1539  # The Winter's Tale
)

# download corpus
shakespeare <- gutenberg_download(
  gutenberg_id = ids,
  meta_fields = "title"
)

# create tokens and drop character cues
shakespeare_clean <- shakespeare %>%
  unnest_tokens(word, text, to_lower = FALSE) %>%
  filter(word != str_to_upper(word))

# calculate TF-IDF
shakespeare_tf_idf <- shakespeare_clean %>%
  count(title, word, sort = TRUE) %>%
  bind_tf_idf(term = word, document = title, n = n)

# plot
shakespeare_tf_idf %>%
  group_by(title) %>%
  top_n(8, tf_idf) %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(tf_idf, word, fill = title)) +
  geom_col() +
  facet_wrap(~title, scales = "free", ncol = 2) +
  theme_minimal() +
  guides(fill = "none")
```



## Resources

- Corporate Reporting in the Era of Artificial Intelligence
- Text Mining with R by Julia Silge and David Robinson

- Supervised Machine Learning for Text Analysis in R by Emil Hvitfledt and Julia Silge
- Corpus