Political Analysis of Social Media Data

Text Analysis Basics

Instructor:	Gregory Eady
Office:	18.2.10
Office hours:	Fridays 13-15

Turning text into data

General approaches

- Keyword frequencies
  - "Going to be a **BAD** day for Crazy Bernie!"
  - Not as easy as you might think
- Bag of words
  - "Crazy a to Bernie day be ! for **BAD** Going"
  - More useful than you might think
- Words-in-context approaches
  - "<u>Going to be a</u> **BAD** <u>day for Crazy Bernie!</u>"
  - More advanced model (e.g. Large Language Models) do this

Keywords

- Simple to understand
- Simple to implement, although "regular expressions" can be tricky
- Simple to come up with?

How easy is keyword selection?

Computer-Assisted Keyword and Document Set Discovery from Unstructured Text

Gary King Harvard University
Patrick Lam Thresher
Margaret E. Roberts University of California, San Diego

Abstract: *The (unheralded) first step in many applications of automated text analysis involves selecting keywords to choose documents from a large text corpus for further study. Although all substantive results depend on this choice, researchers usually pick keywords in ad hoc ways that are far from optimal and usually biased. Most seem to think that keyword selection is easy, since they do Google searches every day, but we demonstrate that humans perform exceedingly poorly at this basic task. We offer a better approach, one that also can help with following conversations where participants rapidly innovate language to evade authorities, seek political advantage, or express creativity; generic web searching; eDiscovery; look-alike modeling; industry and intelligence analysis; and sentiment and topic analysis. We develop a computer-assisted (as opposed to fully automated or human-only) statistical approach that suggests keywords from available text without needing structured data as inputs. This framing poses the statistical problem in a new way, which leads to a widely applicable algorithm. Our specific approach is based on training classifiers, extracting information from (rather than correcting) their mistakes, and summarizing results with easy-to-understand Boolean search strings. We illustrate how the technique works with analyses of English texts about the Boston Marathon bombings, Chinese social media posts designed to evade censorship, and others.*

How easy is keyword selection?

Experiment

For our experiment, we asked 43 relatively sophisticated individuals (mostly undergraduate political science majors at a highly selective college) to recall keywords with this prompt:

> We have 10,000 twitter posts, each containing the word "healthcare," from the time period surrounding the Supreme Court decision on Obamacare. Please list any keywords which come to mind that will select posts in this set related to Obamacare and will not select posts unrelated to Obamacare.

We also gave our subjects access to a sample of the posts and asked them not to consult other sources. We repeated the experiment with an example about the Boston Marathon bombings.

Not very...

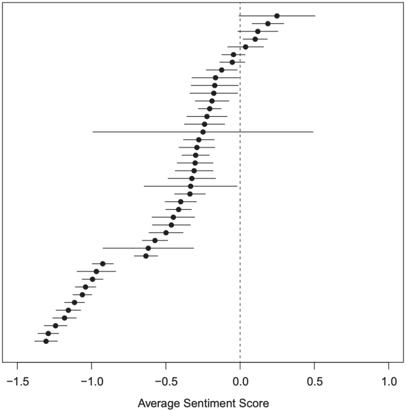
FIGURE 1 The Unreliability of Human Keyword Selection

A Obamacare   **B** Boston Bombings

Note: Word clouds of keywords were selected by human users; those selected by one and only one respondent are in red (or gray if printed in black and white). The position of each word within the cloud is arbitrary.

Does it matter? Unfortunately, yes. Sentiment of keyword lists from 43 people

FIGURE 2 Average Sentiment of 43 Document Sets



Note: Each document set was selected by a different keyword list, with point estimates (as dots) and 95% confidence intervals (horizontal lines) shown.

What to do?

- Keyword expansion technique (King et al. 2017)
  - Unfortunately, this technique is not yet available as an R library
  - You need to code it from scratch, but you need to know other techniques first
  - The technique is a lot simpler than the notation in the paper suggests
- But for now: be aware that keyword approaches are deceptively simple
- Be cautious, but these approaches can be very useful
- We will implement keyword/dictionary-based approaches shortly

Bag of words

- Many approaches assume the context that words find themselves in doesn't matter
- "But context matters!" Yes, of course.
- George Box: "all models are wrong, but some are useful"
- Knowing the relative frequencies of words used by politicians & social media users is actually very informative
  - Topics
  - Ideology
  - Incivility
  - ... Arbitrary classifications of posts or users

From natural language to a bag of words... a Document-Feature Matrix

- Just a matrix of the frequency of "tokens" in a document
- Can think of conceptually as, for example:
  - Document-term matrix or term-document matrix
  - A Tweet-feature matrix
  - A User-feature matrix
  - i.e. However you want to aggregate your data
- Basic idea:
  - Matrix **rows**: a single document, a single tweet, a single user's tweets, etc.
  - Matrix **columns**: the frequency of tokens (i.e. the "words")

Simple example of a document-feature matrix

- Rows are the policy documents of political parties
- Columns are the token ("word") frequencies

```
## Document-feature matrix of: 5 documents, 10 features (0.0% sparse).
## 5 x 10 sparse Matrix of class "dfm"
##        features
## docs     people budget government public minister tax economy pay jobs billion
##   FF         23     44         47     65       11  60      37  41   41      32
##   FG         78     71         61     47       62  11      20  29   17      21
##   Green      15     26         19      4        4  11      16   4   15       3
##   LAB        69     66         36     32       54  47      37  24   20      34
##   SF         81     53         73     31       39  34      50  24   27      29
```

BUT, you have choices of how to construct this matrix

1. **Punctuation**: Spaces & special characters (e.g. $, %, &)
2. **Numbers**: Sometimes informative (e.g. Section 423 of the U.S. Code); other times not
3. **Lowercasing**: Sometimes informative (e.g. "Trump" the president, versus "trump" the verb)
4. **Stopwords**: Common function words, e.g. "the," "and", "it," and "she," or domain-specific ones "congress"

English stop words (are various such lists)

a
about
above
after
again
against
all
am
an
and
any
are
aren't
as
at
be
because
been
before
being
below
between
both
but
by
can't
cannot
could
couldn't
did

ourselves
out
over
own
same
shan't
she
she'd
she'll
she's
should
shouldn't
so
some
such
than
that
that's
the
their
theirs
them
themselves
then
there
there's
these
they
they'd
they'll

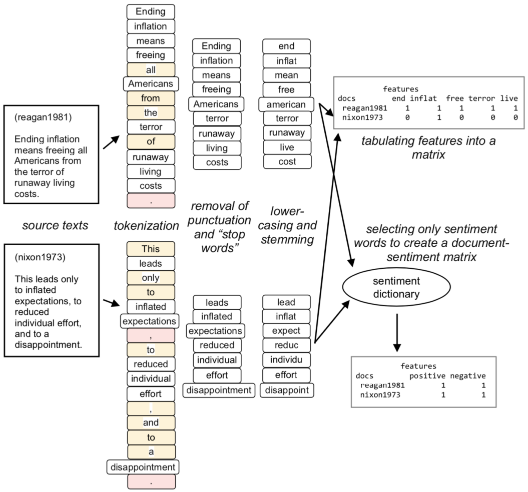
Danish stop words (https://www.ranks.nl/stopwords)

af
alle
andet
andre
at
begge
da
de
den
denne
der
deres
det
dette
dig
din
dog
du
ej
eller
en
end
ene
eneste
enhver
et
fem
fire
flere
fleste
for
fordi
forrige
fra
få
før
god
han
hans
har
hendes
her
hun
hvad
hvem
hver
hvilken
hvis
hvor
hvordan
hvorfor
hvornår
i
ikke
ind
ingen
intet
jeg
jeres
kan
kom
kommer
lav
lidt
lille
man
mand
mange
med
meget
men
mens
mere
mig
ned
ni
nogen
noget
ny
nyt
nær
næste
næsten
og
op
otte
over
på
se
seks
ses
som
stor
store
syv
ti
til
to
tre
ud
var

You have choices of how to construct this matrix

5. **Stemming**: Reducing a word to its root form
    - e.g. "party," "partying," and "parties" all share a common stem "parti"
6. **n-Grams**: treat multiple words as single "tokens". As bi-grams (2) or tri-grams (3), or more
    - e.g. "national" means something much different when combined with "debt" or "defense", ("national defense" versus "national debt")
7. **Infrequently used terms**: Remove very infrequent terms (e.g. remove words that occur in fewer than 0.5-1% of documents)
    - Often don't contribute much information
    - Can substantially reduce the vocabulary size

Example of this process:



These choices matter

- $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 128$ possible combinations
- Many of these choices can only be made relatively arbitrarily
- The results of your work will frequently be sensitive to these choices (e.g. ideology, numbers of topics)

Pre-processing varies a lot in practice:

Table 1. Preprocessing steps taken/suggested in recent notable papers that deal with unsupervised learning methods. The cite total is taken from Google Scholar at the time of writing. In the case of Slapin and Proksch (2008), we consulted their Wordfish manual (version 1.3). In the case of Roberts *et al.* (2014), the authors suggest further steps might be appropriate for a given application.

Citation	Steps	Cites
Slapin and Proksch (2008)	P-S-L-N-W	427
Grimmer (2010)	L-P-S-I-W	258
Quinn *et al.* (2010)	P-L-S-I	275
Grimmer and King (2011)	L-P-S-I	109
Roberts *et al.* (2014)	P-L-S-W	117

P = punctuation; N = numbers; L = lowercasing; S = stemming; W = stopwords; 3 = n-grams; I = infrequent terms

It affects measurement tasks (ideology):

Figure 1. Wordfish results for the 128 different preprocessing possibilities. Each row of the plot represents a different specification. A white bar implies that the manifesto for that year is in the correct place as regards our priors. A black bar implies it was misplaced.

It affects measurement tasks (topics):

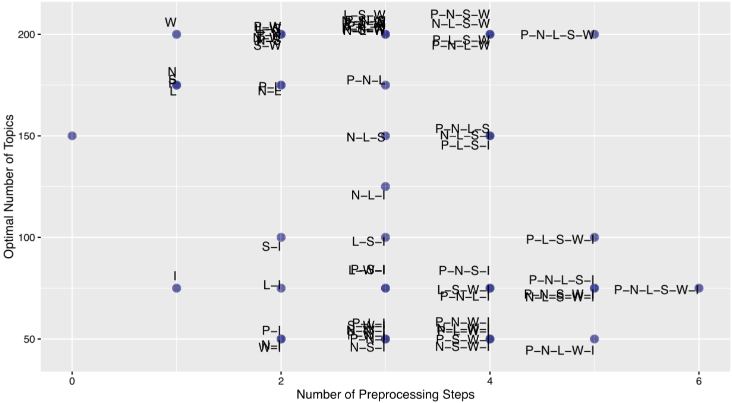


Figure 2. Plot depicting the optimal number of topics (as selected via perplexity) for each of 64 preprocessing specifications not including trigrams. On the x-axis is the number of preprocessing steps, and the y-axis is the number of topics. Each point is labeled according to its specification.

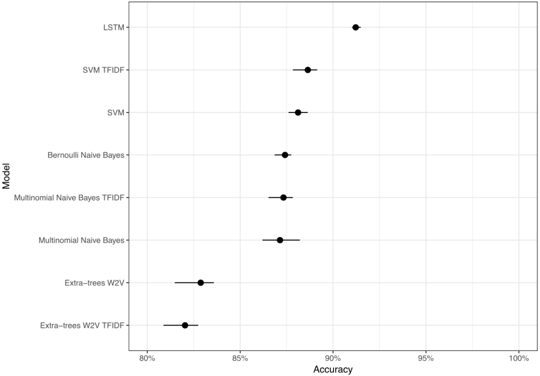
What to do?

- At the moment: Keep in mind that this is a problem
- We may look at robustness checks once you learn how the models work in practice

Words-in-context approaches

- We "know a word by the company it keeps" (Firth, 1957)
- Much more realistic assumptions
- Also much more complex to model (technically & computationally)
- Introduced relatively recently (mid-2010s)
- e.g. Word embeddings (type of neural network) models (Spirling & Rodriguez, 2020) (look up Word2Vec and GloVe)
- Large Language Models

LSTM performance: Predicting whether a Weibo post is about politics



The bars show the minimum and maximum scores for 5 draws of train and test sets from the data while the points show the mean score. The full dataset is 10,691 posts. Each model uses a balanced training dataset of 8,552 posts or 80% of the full dataset. The remaining 2,139 are used in the test set. In the case of LSTM, one-tenth of the training set is split off to use as a validation set. The hyper-parameters for each model are tuned independently.

Figure 1. Classifying political vs. nonpolitical Weibo posts.

LSTM performance: Predicting whether a Weibo post is about politics

Table 2. Precision and recall.

Model	Precision	Recall
LSTM	0.914 (0.011)	0.913 (0.014)
SVM TFIDF	0.887 (0.006)	0.887 (0.006)
SVM	0.882 (0.005)	0.881 (0.005)
Bernoulli naive Bayes	0.874 (0.003)	0.874 (0.003)
Multinomial naive Bayes TFIDF	0.874 (0.005)	0.873 (0.006)
Multinomial naive Bayes	0.873 (0.007)	0.871 (0.007)
Extra-trees W2V	0.829 (0.008)	0.829 (0.008)
Extra-trees W2V TFIDF	0.821 (0.007)	0.820 (0.007)

Note: The mean scores from the five train and test set draws are shown with standard errors in parentheses. Precision measures whether the models identify nonpolitical posts correctly, and recall measures whether the model identifies political posts correctly.

In summary

1. Keyword frequencies
   - Simple in principle, but can be challenging in practice
2. Bag of words
   - Obviously false assumptions, but useful in practice
   - Many meaningful choices in pre-processing can affect results
3. Words-in-context
   - Cutting edge
   - Technically and computationally expensive
   - May perform "best", but the benefits may be marginal
   - Don't fixate on technical performance without good reason

- When people talk about “strings” in R, they are referring to character vectors
  - Example: `c("Donald Trump", "Boris Johnson", "Justin Trudeau")`
- We often want to:
  1. Manipulate strings
  2. Parse/search strings

- Using the `stringr` library
  - stringr library website: `https://stringr.tidyverse.org`
  - List of stringr functions + regular expression cheat sheet: `https://github.com/rstudio/cheatsheets/blob/master/strings.pdf`

Basic operations with text in R

```
# Get the length of a string
# str_length(string)
str_length(c("G. W. Bush", "Obama", "Trump"))
[1] 10  5  5
```

```
# Replace a pattern in a string
# str_replace(string, pattern, replacement)
str_replace("Donald Trump", "Trump", "Drumpf")
[1] "Donald Drumpf"
```

```
# Make a string all upper case
# toupper(x)
toupper(c("G. W. Bush", "Obama", "Trump"))
[1] "G. W. BUSH" "OBAMA"      "TRUMP"
```

```
# Make a string all lower case
# tolower(x)
tolower(c("G. W. Bush", "Obama", "Trump"))
[1] "g. w. bush" "obama"      "trump"
```

Basic regular expressions R

```
# Match basic text
# str_detect(string, pattern, negate = FALSE)
str_detect("Donald Trump", "Trump")
[1] TRUE
```

```
# Match basic text that _starts with_ a string
# str_detect(string, pattern, negate = FALSE)
str_detect("Donald Trump", "^Trump")
[1] FALSE
```

```
# Match basic text that _starts with_ a string
# str_detect(string, pattern, negate = FALSE)
str_detect("Trump, Donald", "^Trump")
[1] TRUE
```

```
# Match basic text that _starts with_ a string
# str_detect(string, pattern, negate = FALSE)
str_detect("Donald Trump", "^Donald Trump Jr.")
[1] FALSE
```

Basic regular expressions R

```
# Match basic text that _ends with_ a string
# str_detect(string, pattern, negate = FALSE)
str_detect("Donald Trump", "Trump$")
[1] TRUE
```

```
# Match basic text that _ends with_ a string
str_detect("Donald Trump", "trump$")
[1] FALSE
```

```
# Match basic text that _ends with_ a string
str_detect("Donald Trump Jr", "Trump$")
[1] FALSE
```

```
# Match basic text that _ends with_ a string
str_detect("Donald Trump", "^Trump$")
[1] FALSE
```

Basic regular expressions R

```
# Match text anywhere
str_detect("Vote now! #Obama #GetOutTheVote", "#Obama")
[1] TRUE
```

```
# Match text anywhere
str_detect("#TeaPartyPatriots #ObamaHatesAmerica", "#Obama")
[1] TRUE
```

```
# Match text with a word boundary
str_detect("#TeaPartyPatriots #ObamaHatesAmerica", "#Obama\\b")
[1] FALSE
```

```
# Match text with a word boundary
str_detect("Vote now! #Obama #BlackLivesMatter", "#Obama\\b")
[1] TRUE
```

Basic regular expressions R

```
# Match multiple possible matches i.e. "Romney" OR "Obama"
str_detect("Vote now! #Obama #GetOutTheVote", "Romney|Obama")
[1] TRUE
```

```
# Match multiple possible matches i.e. "Romney" OR "Obama"
str_detect("Everyone vote for Mitt Romney #tcot", "Romney|Obama")
[1] TRUE
```

```
# Match text regardless of case
str_detect("#TeaPartyPatriots", "party")
[1] FALSE
```

```
# Match text regardless of case
str_detect("#TeaPartyPatriots", regex("party", ignore_case = TRUE))
[1] TRUE
```

Basic regular expressions R

```
# Count possible matches
str_count("Good! This is so so good #goodtimes",
          regex("good", ignore_case = TRUE))
[1] 3
```

```
# Count multiple possible matches
# E.g. for positive sentiment
search_query <- "good|great|amazing|fantastic|best"
str_count("This is the best, greatest thing. So so good! Wow, just amazing!!",
          regex(search_query, ignore_case = TRUE))
[1] 4
```

R Video + exercise

- Online video tutorial using Regex_Example.R in RStudio
- Regex exercise