Week 3: Exploiting Word Meanings

LSE MY459: Quantitative Text Analysis
https://lse-my459.github.io/

Ryan Hübert

Slides last updated: 3 February 2025

Outline for today

- Automated dictionary methods
- Discriminating words
- Probabilistic language models
- Fightin' words
- Extra material on probabilistic models

Automated dictionary methods

Word meanings

An obvious point: words have meanings!

So, word use in a document can give us a sense of the overall "meaning" of that document

What kind of by "meanings"? Some examples:

- Sentiment: positive or negative, etc.
- Emotions: sad, happy, angry, anxious, etc.
- Cognitive processes: insight, causation, discrepancy, tentative, etc.
- Hate speech: sexism, homophobia, xenophobia, racism, etc.

Word meanings

Automated dictionary methods (ADMs) exploit word meanings to learn about the meaning of documents

Two important steps:

1. Construct list of words and associated "meaning(s)" called a **dictionary**

2. Quantify overall "meaning(s)" of documents based on word counts and mapping between dictionary words and meanings

Important: dictionary has to be appropriate for your task, and requires **validation** (more later)

This is another kind of descriptive statistics for texts

Structure of a dictionary

A **dictionary** is a pre-defined list of features (e.g., words) associated with specific meanings

- ➜ It creates equivalence classes of features
- ➜ Frequently involves stemming/lemmatisation: transformation of all word forms to their "dictionary look-up form"

Two important components of dictionaries:

1. **Keys**: the labels for the equivalence class for the concept or canonical term
2. **Values**: (multiple) terms or patterns that are declared equivalent occurrences of the key class

An example

Example from `quanteda` reference documents, in R list form

```
list(christmas = c("Christmas", "Santa", "holiday"),
     opposition = c("Opposition", "reject", "notincorpus"),
     taxglob = "tax*",
     taxregex = "tax.+$",
     country = c("United_States", "Sweden"))
```

Several keys matched to vectors of words or glob/regex patterns

Demonstrates both conceptual flexibility and "look up" flexibility

- ➜ Looking for Christmas-related and tax-related words
- ➜ Looking for exact matches of specific words like "holiday" and also regex patterns like `tax.+$`

(Might want to stem this dictionary, depending on task!)

A "real-world" example

The **LexiCoder Sentiment Dictionary** (Young and Soroka 2012): a dictionary for sentiment analysis in political texts, validated with human-coded news content

Included in `quanteda` as `data_dictionary_LSD2015`:

```
Dictionary object with 4 key entries.
- [negative]:
  - a lie, abandon*, abas* [ ... and 2,855 more ]
- [positive]:
  - ability*, abound*, absolv* [ ... and 1,706 more ]
- [neg_positive]:
  - best not, better not, no damag* [ ... and 1,718 more ]
- [neg_negative]:
  - not a lie, not abandon*, not abas* [ ... and 2,857 more ]
```

Source: https://www.snsoroka.com/data-lexicoder/

Dictionary versus thesaurus

The word "dictionary" is a little bit of a misnomer

- ➜ Like "dictionary" in the python programming sense

Substantively, an ADM dictionary is more similar to a **thesaurus** where a canonical term or concept (a "key") is associated with a list of equivalent synonyms

But unlike thesauruses, ADM dictionaries:

- ➜ Tend to be "exclusive" (each value associated with *one* key)
- ➜ Aren't strictly meant to identify synonyms

But, you could create a non-exclusive ADM dictionary, such as:

- ➜ **marriage** = `engage, ring, wedding, spouse, husband, wife`
- ➜ **interest** = `engage, appeal, excite, attract, entertain`

Famous dictionaries

General Inquirer: an early "all purpose" dictionary

➜ https://inquirer.sites.fas.harvard.edu/spreadsheet_guide.htm

Regressive Imagery Dictionary: designed to measure primordial vs. conceptual thinking

➜ http://www.kovcomp.co.uk/wordstat/RID.html

Linguistic Inquiry and Word Count or **LIWC** (pronounced "Luke"): large dictionary + (paid) software to generate analyses

➜ https://liwcsoftware.onfastspring.com

Famous dictionaries

Valence Aware Dictionary and sEntiment Reasoner or (**VADER**): open source alternative to LIWC, tuned for modern social media text

➜ https://github.com/cjhutto/vaderSentiment

LexiCoder (mentioned above)

SentiStrength: For social media text

➜ https://mi-linux.wlv.ac.uk/~cm1993/sentistrength/

Moral Foundations Dictionary: meant to capture "moral foundations" concepts

➜ https://moralfoundations.org/other-materials/

Always: read the docs, use correctly and for intended use cases!

Automated dictionary methods

Suppose you already have a dictionary in hand

Two common goals of automated dictionary methods:

1. Quantify documents on a scale, such as *how positive or negative is a document?*

2. Classify documents into discrete categories based on meaning of words used, such as *is this a positive tone document or a negative tone document?*

Calculating quantities of interest: example

Example dictionary: every feature j labeled with a sentiment, positive ($s_j = 1$), negative ($s_j = -1$) or neither ($s_j = 0$)

For each document i, you can calculate its "sentiment" S_i:

$$S_i = \sum_j s_j W_{ij}$$

You can normalise by document length:

$$\frac{S_i}{M_i}, \quad \text{where } M_i = \sum_j W_{ij}$$

Or you can classify using a decision rule:

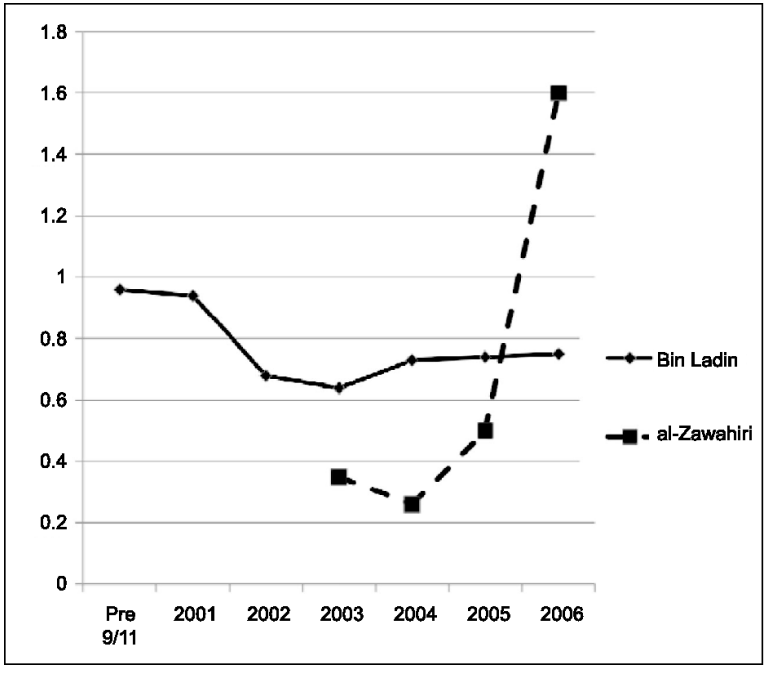
$$D_i = \begin{cases} 1 & \text{if } S_i \geq 0 \\ -1 & \text{if } S_i < 0 \end{cases}$$

Example: Terrorist speech

Pennebaker and Chung (2008) and Hancock et al. (2010)

- ➜ Use ADMs to analyse Al Qaeda discourse in videotapes, interviews and letters

From Hancock et al. (2010):



- ➜ Key Finding: Zawahiri was feeling threatened, indicating a rift with bin Laden
- ➜ Use of first-person pronouns (I, me, my, mine): bin Laden's use remained constant, Zawahri's use increased

Source: https://doi.org/10.1080/19434471003597415

Emotional contagion on Facebook

Kramer et al. (2014): study of 689,003 Facebook users

Experimentally manipulated content shown on news feeds to test **emotional contagion hypothesis**

- ➜ Treatment 1: Positive content more visible on news feed
- ➜ Treatment 2: Negative content more visible on news feed
- ➜ Control: No news feed intervention

Use ADM to measure sentiment of users' Facebook posts post-treatment, showing:

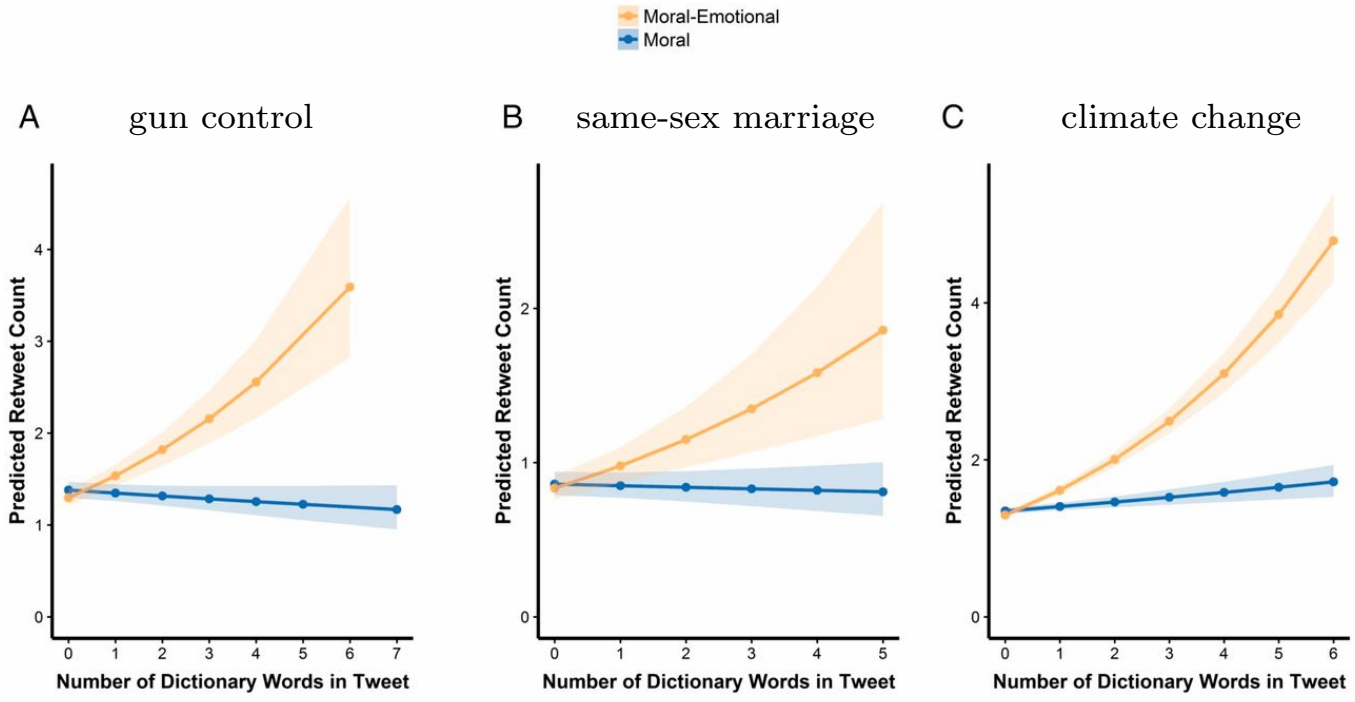
- ➜ positive content boosts positive posting, reduces negative
- ➜ negative content boosts negative posting, reduces positive

Huge concerns about ethics of the study; very controversial

Source: https://www.pnas.org/doi/10.1073/pnas.1320040111

Moral contagion on Twitter

Brady et. al. (2017) used dictionary of moral-emotional words to measure whether moral-emotional language use predicts the greatest number of retweets, which they call "moral contagion"



Source: https://doi.org/10.1073/pnas.1618923114

Validating a dictionary

Automated dictionary methods should always be validated!

Qualitative evaluations:

- ➜ When creating or choosing a dictionary: careful identification of concepts, associated keys/categories, and textual features associated with each key/category
- ➜ Read some documents to get a gut check about whether the dictionary is capturing meanings correctly
- ➜ Check sensitivity of results to exclusion of specific words

Validating a dictionary

Automated dictionary methods should always be validated!

Quantitative evaluations:

- Create a human-coded **validation set** and compare your dictionary method, treating human-coding as the truth
  - **Accuracy**: what % of documents is correctly coded by your ADM?
  - **Precision** of category: what % of documents coded in a category by your ADM were coded correctly?
  - **Recall**: what % of documents coded in a category by human-coders were coded correctly by your ADM?

Weakness: context specificity

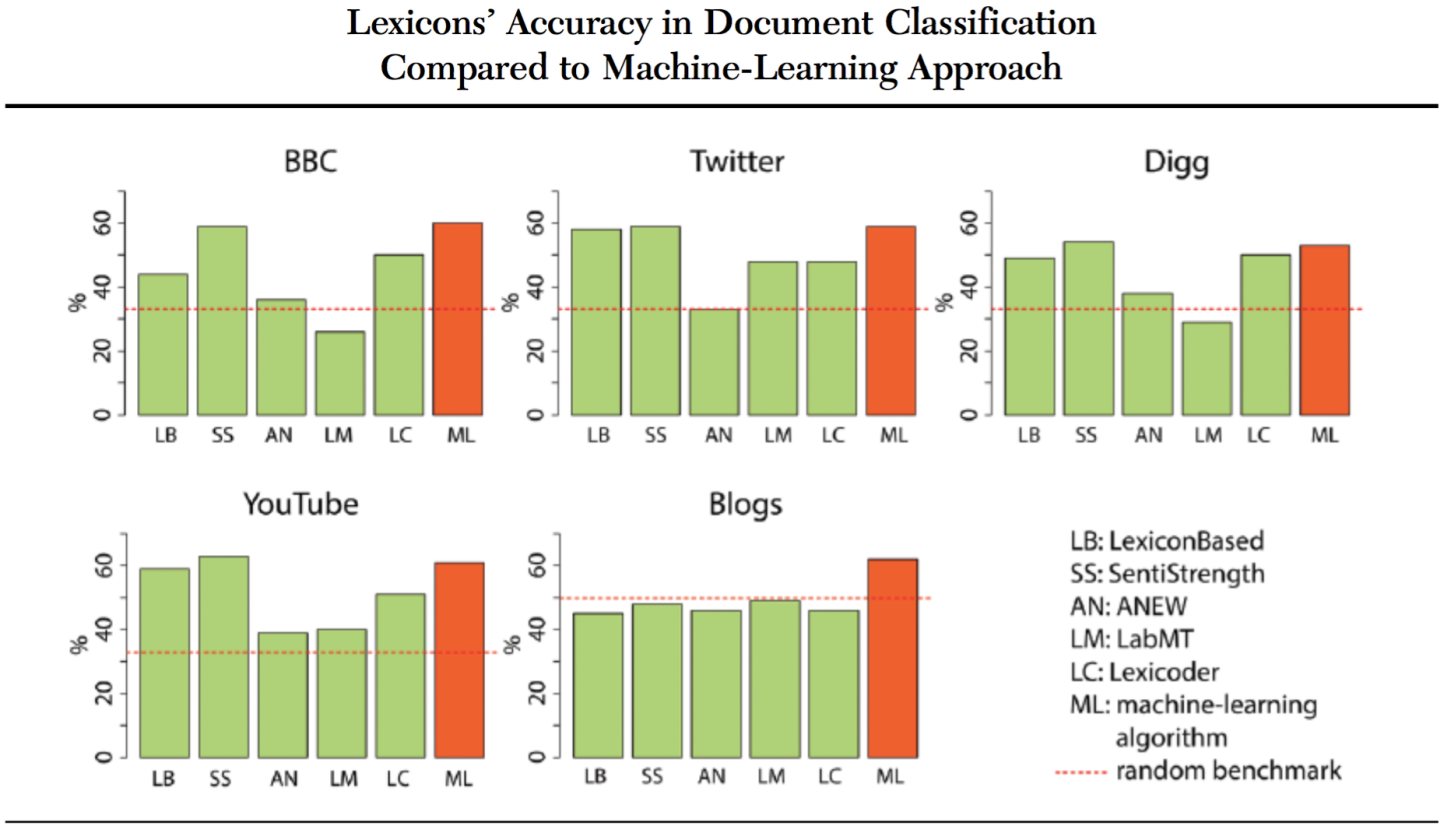
Loughran and McDonald (2011): used the Harvard-IV-4 TagNeg (H4N) file to classify sentiment for a corpus of 50,115 firm-year 10-K filings from 1994–2008

- ➜ Almost three-fourths of "negative" words in dictionary were not typically negative in financial context, e.g. `tax`, `cost`, `crude`, `cancer`
- ➜ Problem is **polysemes**—words that have multiple meanings
- ➜ Another problem: dictionary lacked important negative financial words, such as `felony`, `litigation`, `restated`, `misstatement`, and `unanticipated`

Source: https://doi.org/10.1111/j.1540-6261.2010.01625.x

Weakness: context specificity

González-Bailón and Paltoglou (2015) shows how different dictionaries perform differently across different contexts:



Source: https://doi.org/10.1177/0002716215569192

Weakness: worse than ML classification

Barberá et al. (2021) codes a set of >4,000 articles, showing ML generally does better than ADMs coding sentiment

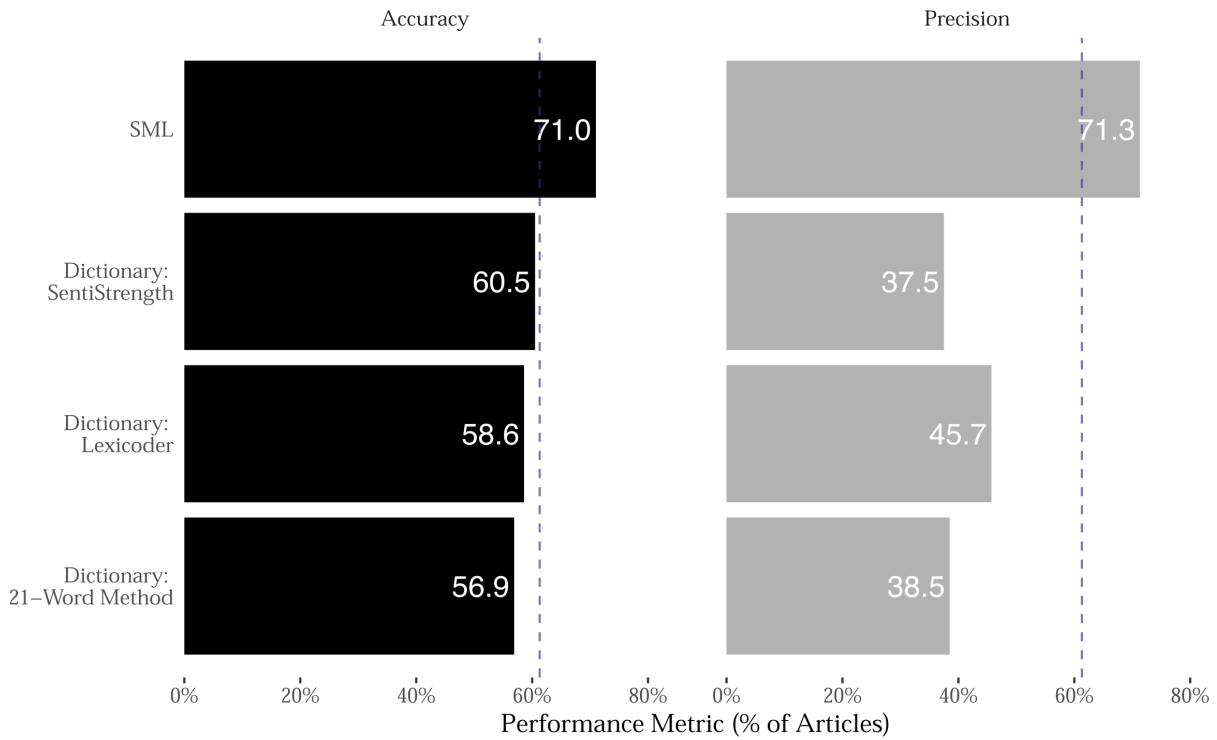


Figure 3. Performance of SML and Dictionary Classifiers—Accuracy and Precision.
Note: Accuracy (percent correctly classified) and precision (percent of positive predictions that are correct) for the ground truth dataset coded by ten CrowdFlower coders. The dashed vertical lines indicate the baseline level of accuracy or precision (on any category) if the modal category is always predicted. The corpus used in the analysis is based on the keyword search of *The New York Times* 1980–2011 (see the text for details).

Source: http://doi.org/10.1017/pan.2020.8

Weakness: sensitive to frequent words

Pury (2011) showing problem with ADM in Back et al. (2010):

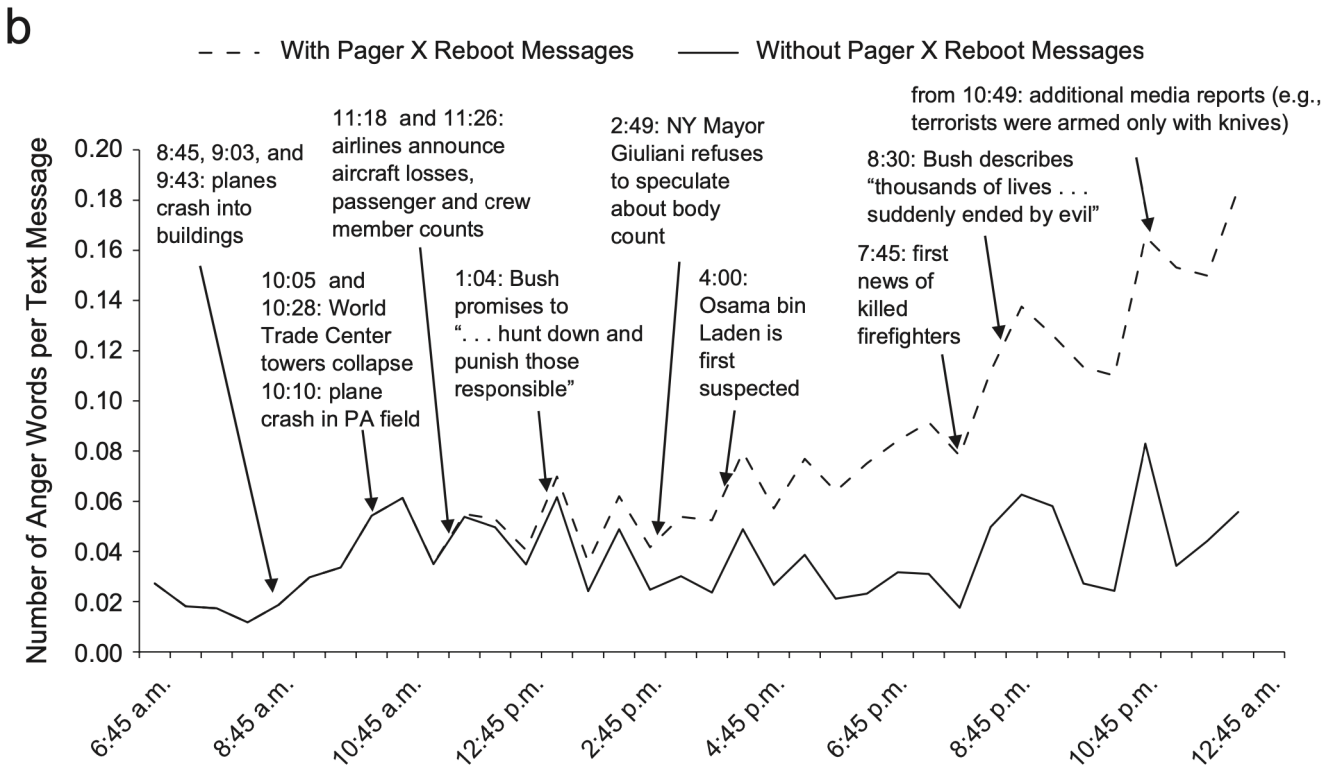


Fig. 1. Proportion of automatically generated text messages and timeline of anger as expressed in text messages on September 11, 2001. The graph in (a) shows the proportion of the analyzed pager messages that were nearly identical technical "reboot" messages sent to a single pager, Pager X, as a function of time block. The graph in (b) shows the mean number of Linguistic Inquiry and Word Count anger words per alphanumeric pager message as a function of time block; means were calculated both with and without the Pager X messages included in the population of messages. In both graphs, the 30-min time blocks are indicated by their starting time. The timeline in (b) is based on Figure 1 in Back, Küfner, and Egloff (2010).

Sources: original paper, https://doi.org/10.1177/0956797610382124; critique, https://doi.org/10.1177/0956797611408735; reanalysis, https://doi.org/10.1177/0956797611409592

Discriminating words

Discriminating words

Automated dictionary approaches presume the existence of a pre-made dictionary constructed for the particular context

But what if you want to *figure out* which words are most illustrative of some pre-defined categories?

Why might you want to do this? E.g.,

- ➜ understand how rhetoric differs between Democrats and Republicans (Monroe, Colaresi and Quinn 2008)
- ➜ understand how rhetoric differs between first-wave and second-wave feminist movements in the US (Nelson 2020)
- ➜ build a new dictionary based on example texts

This is known as identifying "**discriminating words**" or "key words", and `quanteda` calls it "measuring keyness"

Detecting discriminating words

Suppose you have N documents and each document i fits into one of K known categories, e.g.,

- ➜ partisanship of author, e.g. Democrat v. Republican
- ➜ time period of document, e.g. first wave v. second wave feminist movements

We'll use k as a generic placeholder for a specific category

The core question: *for each feature (e.g. "word") j in a corpus's vocabulary, how much does the presence of that feature in a document inform our guess about the document's category?*

- ➜ Process: go feature-by-feature and calculate how much that feature discriminates

We'll consider simple examples with two categories

Contingency table

A useful starting point is a **contingency table**

This is a general concept from statistics, but in our case, it will look like a "consolidated" DFM

Specifically:

- ➜ consolidate all features that are not j into a feature "not j"
- ➜ consolidate all documents into their categories

Here: "consolidate" means add together word counts

Running example: which words distinguish Trump's speech, relative to all other post-war presidents in the inaugural address corpus? (Corpus available in `quanteda` as `data_corpus_inaugural`)

Contingency table

Let's measure how discriminating the word "America" is between Trump's address and all other postwar presidents

The contingency table looks like this:

```
Document-feature matrix of: 2 documents, 2
  features (0.00% sparse) and 1 docvar.
           features
docs        america not.america
  Not.Trump     167        17490
  Trump          19          694
```

Some useful quantities from the table:

- Total tokens: $W = 167 + 19 + 17490 + 694 = 18370$
- Total instances of "america": $W_{\mathtt{a}} = 167 + 19 = 186$
- Total tokens by Trump: $W_{\mathcal{T}} = 19 + 694 = 713$
- Total instances of "america" by Trump: $W_{\mathcal{T}\mathtt{a}} = 19$

Two measurement approaches

The contingency table gives us the basic word counts we need to do our calculations

Once you have it, then, what's next?

Two approaches for measuring feature discrimination among known categories:

1. Statistical association measures
2. Model-based approaches ("fightin' words")

Statistical association measures

We will use the logic of hypothesis testing from statistics

1. Set up an expectation based on an **independence assumption** that there is no correlation between features and categories

   ➜ In math, assume that $\Pr(j, k) = \Pr(j) \Pr(k)$ for every j and k

2. Estimate $\widehat{\Pr}(j, k)$ using data, such as:

$$\widehat{\Pr}(\texttt{america}, \textsf{Trump}) = \widehat{\Pr}(\texttt{america})\widehat{\Pr}(\textsf{Trump}) = \frac{186}{18370} \times \frac{713}{18370} = 0.0003930$$

3. Calculate expected word counts under independence assumption, given corpus size, such as:

$$E_{\mathcal{T}_\mathsf{a}} = 18370 \times 0.0003930 \approx 7.21941$$

Hypothetical contingency table

We can repeat for each combination of feature and category, and create a *hypothetical* contingency table assuming independence

In a 18,370-token corpus, here's what we would expect to see if Trump was no more or less likely to use the token "america"

	features	
docs	america	not.america
Not.Trump	178.8	17478.2
Trump	7.2	705.8

Keep in mind: we will notate word counts in the real contingency table using W, and word counts in the hypothetical contingency table using E

With both tables, we can calculate three measures

Point-wise mutual information (PMI)

For a category k and a feature j, the **point-wise mutual information** score is:

$$\text{pmi}_{kj} = \log\left(\frac{W_{kj}}{E_{kj}}\right)$$

If you want to skip making the hypothetical contingency table, you can use this formula, where W_j is number of times feature j is used and k is number of tokens in category k documents

$$\text{pmi}_{kj} = \log\left(\frac{W_{kj} \times W}{W_j \times W_k}\right)$$

Note: GSR discusses **mutual information** which is related to PMI

Point-wise mutual information (PMI)

Recall our two tables

```
     doc_id america not.america
1 Not.Trump     167       17490
2     Trump      19         694
```

```
                features
docs         america not.america
  Not.Trump    178.8     17478.2
  Trump          7.2       705.8
```

We can calculate PMI as (using base e):

$$\text{pmi}_{\mathcal{T}\text{a}} = \log\left(\frac{19}{7.2}\right) = 0.97$$

Note: we use base e because `quanteda` does; but base 2 is common — it doesn't matter as long as you are consistent

Positive numbers: feature is more prevalent in category (relative to other categories); negative: feature less prevalent

Statistical association measures

Two other measures use all the info in the contingency table:

The **likelihood ratio statistic** (G^2):

$$G^2_{kj} = 2 \times \sum_k \sum_j \left(W_{kj} \times \log \left(\frac{W_{kj}}{E_{kj}} \right) \right)$$

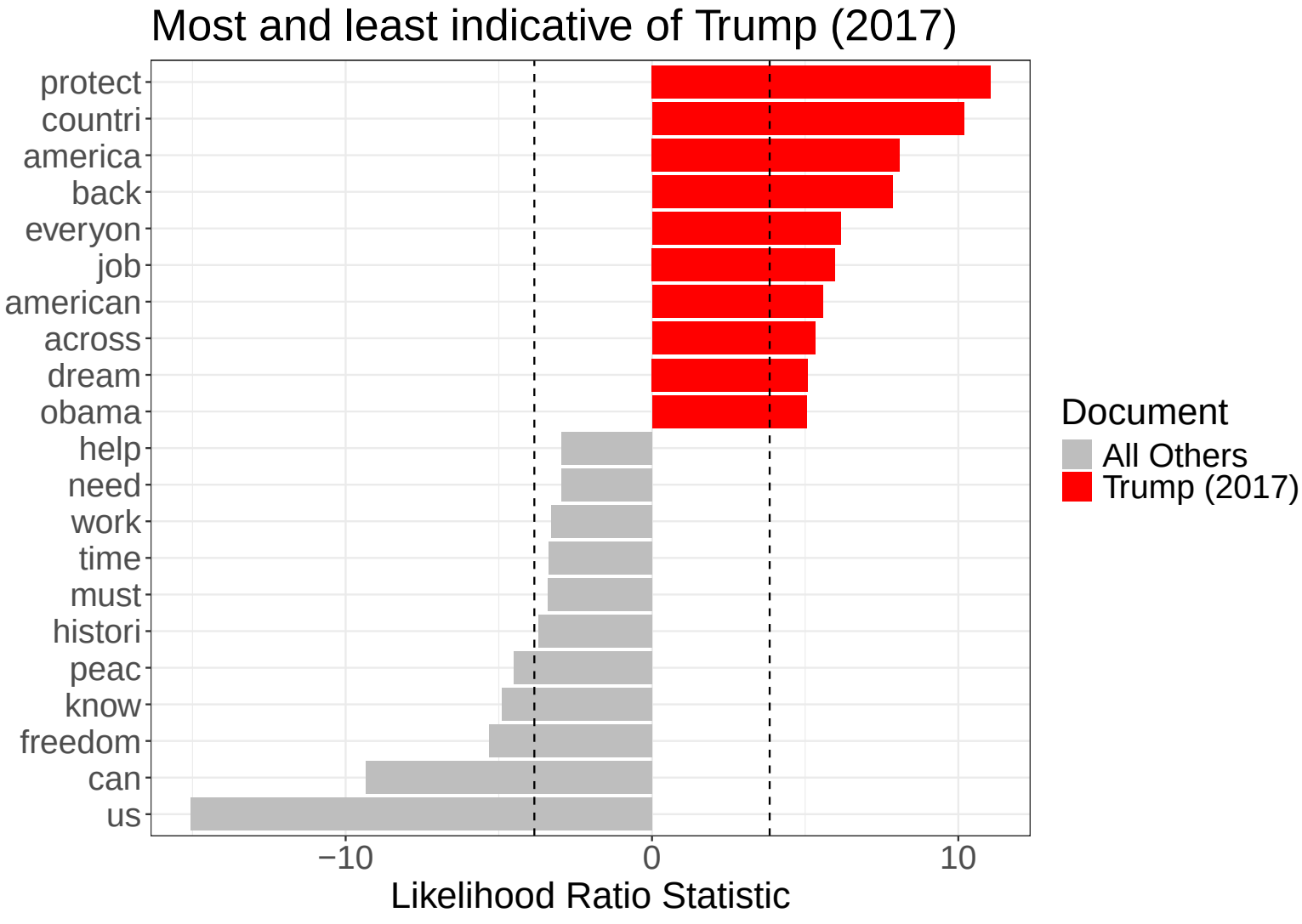
The **Pearson's χ^2 statistic**:

$$\chi^2_{kj} = \sum_k \sum_j \left(\frac{(W_{kj} - E_{kj})^2}{E_{kj}} \right)$$

Example: Trump's 2017 inaugural address

	feature	G2	p	n_target	n_reference
1	protect	11.050824	0.0008864781	8	33
2	countri	10.180940	0.0014189980	13	85
3	america	8.085458	0.0044621847	20	185
4	back	7.852023	0.0050763866	7	35
5	everyon	6.162821	0.0130462956	5	23
6	job	5.985209	0.0144263330	6	33
7	american	5.574184	0.0182271526	16	157
8	across	5.325173	0.0210195268	6	36
9	dream	5.081332	0.0241847898	7	48
10	obama	5.048276	0.0246504586	4	18

Example: Trump's 2017 inaugural address



Probabilistic language models

Probabilistic models

Basic idea: documents are "draws" from a probability distribution

So, a dfm is a sample with all the associated sampling properties

Think back to your stats courses:

- ➜ A **data generating process** is a probability distribution that (we assume) captures way real-life data occurs
- ➜ Real-life datasets are conceptualised as **samples** from a DGP
- ➜ In some sense, the DGP is what we care about since it tells us "how things work"
- ➜ We use real-life data to try to learn about the DGP

Language is often modeled using probabilistic models

Language models

makes total sense for the context of bag of words

we are assuming words are independent of each other (independence assumption)

One of the simplest language models: assume each document $\mathbf{W}_i$ is a "draw" from a **multinominal distribution** with two parameters:

- $\boldsymbol{\mu} \rightarrow$ probability each feature in the J-feature vocabulary is drawn → these are effectively vectors with probabilities for each word that appear in the text
- $M_i \rightarrow$ the document length (number of tokens)

In math notation, $\mathbf{W}_i \sim \text{Multinomial}(M_i, \boldsymbol{\mu})$

Diagram for 3-document model

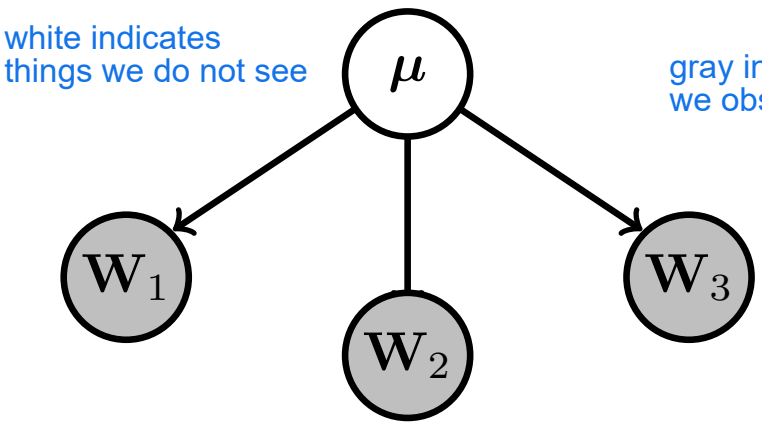
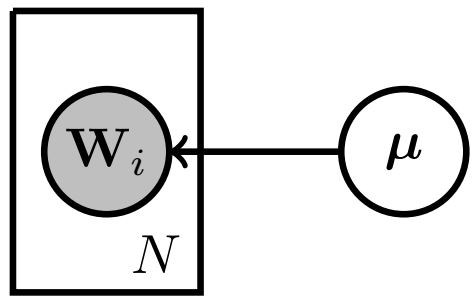


Diagram for N-document model
(plate diagram)



The multinomial distribution

Multinomial draws a "document" as just a vector of word counts, ignoring word order

This makes sense with bag of words

Also the multinomial distribution is really easy to work with, as you'll see shortly

Very simple example

Suppose a vocabulary with $J = 3$ features: `cat`, `dog`, `rat`

Further suppose that `cat` is twice as likely as `dog` or `rat` (which are equally likely):

$$\boldsymbol{\mu} = (\mu_{\text{cat}} = 0.50, \mu_{\text{dog}} = 0.25, \mu_{\text{rat}} = 0.25)$$

Notice we've made an *assumption* about the way documents are made

Very simple example

Suppose our assumption is correct, and then *simulate* drawing ("creating") four documents of different lengths in R:

```
v <- c("cat","dog","rat") # the features
mu <- c(0.5,0.25,0.25) # the assumed feature probabilities
M <- c(2,3,2,6) # draw a sample of 4 docs with varying lengths
set.seed(2025) # set seed for reproducibility
for(i in 1:length(M)){
  w <- rmultinom(1,M[i],mu)
  wi <- c(rep(v[1],w[1,1]), rep(v[2],w[2,1]), rep(v[3],w[3,1]))
  print(paste0("Document ",i,": ",
               paste0(sample(wi), collapse = " ")))
}
```

```
[1] "Document 1: rat cat"
[1] "Document 2: cat cat dog"
[1] "Document 3: rat cat"
[1] "Document 4: dog rat rat rat cat dog"
```

Very simple example

These documents could be put into a DFM:

	cat	dog	rat	
Document 1	1	0	1	<- "rat cat"
Document 2	2	1	0	<- "cat cat dog"
Document 3	1	0	1	<- "rat cat"
Document 4	1	2	3	<- "dog rat rat rat cat dog"

Each document can be represented mathematically in vector form

For example, Documents 1 and 4:

$$\mathbf{W}_1 = (1, 0, 1) \qquad \mathbf{W}_4 = (1, 2, 3)$$

Calculating important stuff

It is common to use the multinomial distribution to model language because it's easy to work with

- ➜ Again: keep in mind we're assuming bag of words!

We can look up various important formulas for multinomial distributions

E.g., calculate the probability of getting a particular document $\mathbf{W}_i$, given $\boldsymbol{\mu}$:

$$p(\mathbf{W}_i|\boldsymbol{\mu}) = \frac{M_i!}{\prod_{j=1}^{J}(W_{ij}!)} \prod_{j=1}^{J} \left(\mu_j^{W_{ij}}\right)$$

- ➜ Again: keep in mind we're assuming bag of words!

Calculating important stuff

Back to our simple example with three word vocabulary:

$$\boldsymbol{\mu} = (\mu_{\text{cat}} = 0.50, \mu_{\text{dog}} = 0.25, \mu_{\text{rat}} = 0.25)$$

What is probability of getting a document i that looks like this?
`cat rat dog cat`

- In vector form, this is $\mathbf{W}_i = (2, 1, 1)$

$$p(\mathbf{W}_i|\boldsymbol{\mu}) = \frac{(2+1+1)!}{2! \times 1! \times 1!} \left[0.5^2 \times 0.25^1 \times 0.25^1\right] = \frac{12}{64} = 0.1875$$

Very simple example

We can calculate in R as well

```
v <- c("cat","dog","rat") # the features
m <- c(0.5,0.25,0.25) # the assumed feature probabilities
dmultinom(c(2,1,1),prob=m)
```

```
[1] 0.1875
```

we do that process backwards in real life to figure out the real (true) μ

--> this is inference

Estimating models with data

In real-world situations, we don't usually know the DGP

➜ We don't actually know $\boldsymbol{\mu}$ and we need to estimate it from data

How? It's simple: for every feature j, we can calculate the proportion of all documents that use word j:

$$\hat{\mu}_j = \frac{W_{ij}}{M_i} = \frac{W_{ij}}{\sum_j W_{ij}}$$

Why does this matter?

➜ We want to estimate the parameters of the DGP so we have a rough idea how documents show up in our data

➜ This then enables us to do some cool things to "fill in" gaps in our knowledge and learn about texts (example next!)

Example: the Federalist papers

When the US was established, three people wrote a set of essays called The Federalist Papers in support of the new US Constitution

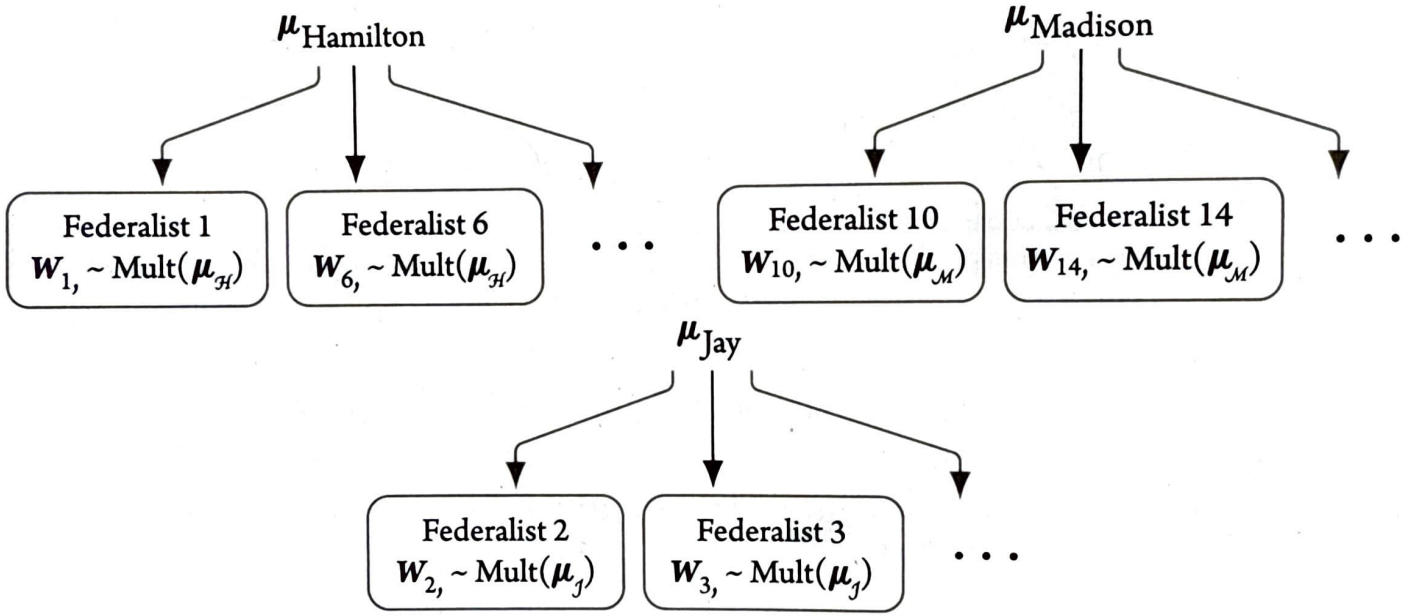
- The people: Alexander Hamilton, John Jay, James Madison
- There were 85, none of them had bylines (all "Publius")
- Historical records tell us who wrote 73 of them; remainder are contested and thus remain "unlabeled"

Mosteller and Wallace (1963) use a probability model to try to figure out who wrote the unlabeled Federalist papers

Example: the Federalist papers

We'll examine a simplified version of this task where we set up probabilistic models that assume each writer has distinctive style

Mathematically, each writer's μ is different: $\mu_{\mathcal{H}}$, $\mu_{\mathcal{J}}$, $\mu_{\mathcal{M}}$



Source: Figure 6.3 of GRS (p. 64)

Example: the Federalist papers

The basic process:

Step 1: estimate $\boldsymbol{\mu}$ for each writer ("figure out their writing style") using real-world data, generating estimates: $\hat{\boldsymbol{\mu}}_{\mathcal{H}}$, $\hat{\boldsymbol{\mu}}_{\mathcal{J}}$, $\hat{\boldsymbol{\mu}}_{\mathcal{M}}$

Step 2: see which model best explains the unlabeled documents by calculating probability each person was the author of the unlabeled documents

necessary assumption, otherwise cannot use that method

If we assume each author's style is constant across documents, things get real simple (due to properties of multinomial distribution):

➜ Just add together all the tokens — essentially assuming that each author writes one big document

Example: the Federalist papers

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

What is the probability that Hamilton wrote the unlabeled ones?

Example: the Federalist papers

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

What is the probability that Hamilton wrote the unlabeled ones?

Step 1: estimate Hamilton's "language model" $\hat{\boldsymbol{\mu}}_{\mathcal{H}}$

$$\hat{\boldsymbol{\mu}}_{\mathcal{H}} = \left(\frac{859}{859+102+374}, \frac{102}{859+102+374}, \frac{374}{859+102+374} \right) \approx (0.64, 0.08, 0.28)$$

Example: the Federalist papers

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

What is the probability that Hamilton wrote the unlabeled ones?

Step 1: estimate Hamilton's "language model" $\hat{\boldsymbol{\mu}}_{\mathcal{H}}$

$$\hat{\boldsymbol{\mu}}_{\mathcal{H}} = \left(\frac{859}{859+102+374}, \frac{102}{859+102+374}, \frac{374}{859+102+374} \right) \approx (0.64, 0.08, 0.28)$$

Step 2: calculate probability Hamilton wrote unlabeled documents

$$\Pr(\mathbf{W}_{\text{Unlabeled}} | \hat{\boldsymbol{\mu}}_{\mathcal{H}}) = \frac{(15+2+0)!}{15! \times 2! \times 0!} \times 0.64^{15} \times 0.08^{2} \times 0.28^{0} = 0.001$$

Conclude: approx. 0.1% chance Hamilton wrote disputed documents

Example: the Federalist papers

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

Can repeat for Madison:

$$\hat{\boldsymbol{\mu}}_{\mathcal{M}} = \left(\frac{474}{474+17+7}, \frac{17}{859+17+7}, \frac{7}{859+17+7}\right) \approx (0.95, 0.035, 0.015)$$

$$\Pr(\mathbf{W}_{\text{Unlabeled}}|\hat{\boldsymbol{\mu}}_{\mathcal{M}}) = \frac{(15+2+0)!}{15! \times 2! \times 0!} \times 0.95^{15} \times 0.035^{2} \times 0.015^{0} = 0.077$$

Conclude: approx. 8% chance Madison wrote disputed documents

Example: the Federalist papers

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

And for Jay:

$$\hat{\boldsymbol{\mu}}_{\mathcal{J}} = \left(\frac{82}{82+0+1}, \frac{0}{82+0+1}, \frac{1}{82+0+1} \right) \approx (0.99, 0, 0.01)$$

$$\Pr(\mathbf{W}_{\text{Unlabeled}} | \hat{\boldsymbol{\mu}}_{\mathcal{J}}) = \frac{(15+2+0)!}{15! \times 2! \times 0!} \times 0.99^{15} \times 0^{2} \times 0.01^{0} = 0$$

Conclude: 0% chance Jay wrote disputed documents

Overfitting due to low word counts

We conclude Madison is most likely to be author of unlabeled texts

But, there's a weird issue: we predicted a 0% chance Jay wrote the documents

- ➜ This is an example of **overfitting**: lessons drawn from data (a sample) are "too" closely tied to the patterns in the data

Problem: Jay never used the word "man" in labeled documents

- ➜ Estimated model: any document with "man" can never be written by Jay
- ➜ Our model of Jay's word use is overfitting the sample data
- ➜ No use of "man" is likely just a sampling fluke—surely Jay knew and used the word "man"!

General issue with sparse DFMs

Regularisation and smoothing

We solve this with **regularisation**: "a non-data piece of information or a constraint that is added to the model to draw estimates toward a particular value." (GSR, p. 66)

Many approaches to regularisation, but a very easy one in this context is **Laplace smoothing**

➜ Add a little bit of fake data (a constant α) to ensure no zeroes

For example we can add $\alpha = 1$ to our Federalist papers data

	by	man	upon
Hamilton	859	102	374
Jay	82	0	1
Madison	474	17	7
Unlabeled	15	2	0

	by	man	upon
Hamilton	860	103	375
Jay	83	1	2
Madison	475	18	8
Unlabeled	16	3	1

Regularisation and smoothing

We solve this with **regularisation**: "a non-data piece of information or a constraint that is added to the model to draw estimates toward a particular value." (GSR, p. 66)

Many approaches to regularisation, but a very easy one in this context is **Laplace smoothing**

➜ Add a little bit of fake data (a constant α) to ensure no zeroes

For example we can add $\alpha = 1$ to our Federalist papers data

This changes our calculations slightly:

$$\hat{\mu}_j = \frac{W_{ij} + \alpha}{M_i + \sum_j \alpha_j}$$

Regularisation and smoothing

Let's recalculate Federalist probabilities with this regularised data

	by	man	upon
Hamilton	860	103	375
Jay	83	1	2
Madison	475	18	8
Unlabeled	16	3	1

For example, probability the unlabeled texts are written by Jay

$$\hat{\boldsymbol{\mu}}_{\mathcal{J}} = \left(\frac{83}{83+1+2}, \frac{1}{83+1+2}, \frac{2}{83+1+2}\right) \approx (0.97, 0.01, 0.02)$$

$$\Pr(\mathbf{W}_{\text{Unlabeled}} | \hat{\boldsymbol{\mu}}_{\mathcal{J}}) = \frac{(16+3+1)!}{16! \times 3! \times 1!} \times 0.97^{16} \times 0.01^{3} \times 0.02^{1} = 0.0000048$$

Fightin' words

Discriminating words, continued

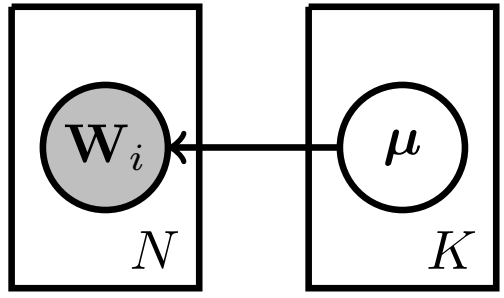
Back to our example of the Trump inaugural addresses

Recall: statistical association measures posit an independence model and then calculate the likelihood that the observed text is consistent with it

Now that we know how to construct rich probabilistic models of language, why not just model Trump's language directly and then see how word use differs?

Language model

Let's assume a language model like the following:



Specifically:

- $K = 2$, so $\boldsymbol{\mu} = [\boldsymbol{\mu}_{\mathcal{T}}, \boldsymbol{\mu}_{\mathcal{O}}]$ ($\mathcal{O}$ = other postwar presidents)
- $J = 2$, so each of $\boldsymbol{\mu}_{\mathcal{T}}$ and $\boldsymbol{\mu}_{\mathcal{O}}$ has a length of 2

Again, because DFMs are sparse, we probably want to regularise

- And we will use Laplace smoothing with $\alpha = 1$

(Note: original paper regularises differently)

Estimating $\boldsymbol{\mu}$ with data

We can estimate using this formula

$$\widehat{\mu}_{kj} = \frac{W_{kj} + \alpha_j}{W_k + \sum_j \alpha_j}$$

For each token j,

$$\widehat{\boldsymbol{\mu}}_{\mathcal{T}} = [\widehat{\mu}_{\mathcal{T}j}, 1 - \widehat{\mu}_{\mathcal{T}j}] \qquad \widehat{\boldsymbol{\mu}}_{\mathcal{O}} = [\widehat{\mu}_{\mathcal{O}j}, 1 - \widehat{\mu}_{\mathcal{O}j}]$$

Estimating μ with data

Let's do this for "america" — recall the contingency table

```
Document-feature matrix of: 2 documents, 2
  features (0.00% sparse) and 1 docvar.
                features
docs          america not.america
  Not.Trump       167        17490
  Trump            19          694
```

So:

$$\widehat{\mu}_{\mathcal{T}\mathtt{a}} = \frac{19+1}{19+694+1+1} = 0.0280 \qquad \widehat{\mu}_{\mathcal{O}\mathtt{a}} = \frac{167+1}{167+17490+1+1} = 0.0095$$

We can see right away: our estimates tell us Trump has a higher probability of using "america" than others

➜ How to use them to calculate a standard "score" for the word?

Measuring word discrimination

Monroe, Colaresi and Quinn (2008) recommend log-odds difference for quantifying how discriminating a word j is between group k and group k':

$$\hat{\delta}_j^{(k-k')} = \log\left(\frac{\widehat{\mu}_{kj}}{1-\widehat{\mu}_{kj}}\right) - \log\left(\frac{\widehat{\mu}_{k'j}}{1-\widehat{\mu}_{k'j}}\right)$$

This puts too much weight on infrequent words, so we can normalise to a z-score as follows:

$$\hat{z}_j^{(k-k')} = \frac{\hat{\delta}_j^{(k-k')}}{\sqrt{\mathrm{Var}\left(\hat{\delta}_j^{(k-k')}\right)}}$$

Where:

$$\mathrm{Var}\left(\hat{\delta}_j^{(k-k')}\right) \approx \frac{1}{W_{kj}+\alpha_j} + \frac{1}{W_{k'j}+\alpha_j}$$

Measuring word discrimination

Back to our example, we can use $\widehat{\mu}_{\mathcal{T}\mathtt{a}}, \widehat{\mu}_{\mathcal{O}\mathtt{a}}$ to calculate:

$$\hat{\delta}_{\mathtt{a}}^{(\mathcal{T}-\mathcal{O})} = \log\left(\frac{0.0280}{0.972}\right) - \log\left(\frac{0.0095}{0.9905}\right) \approx 1.10$$

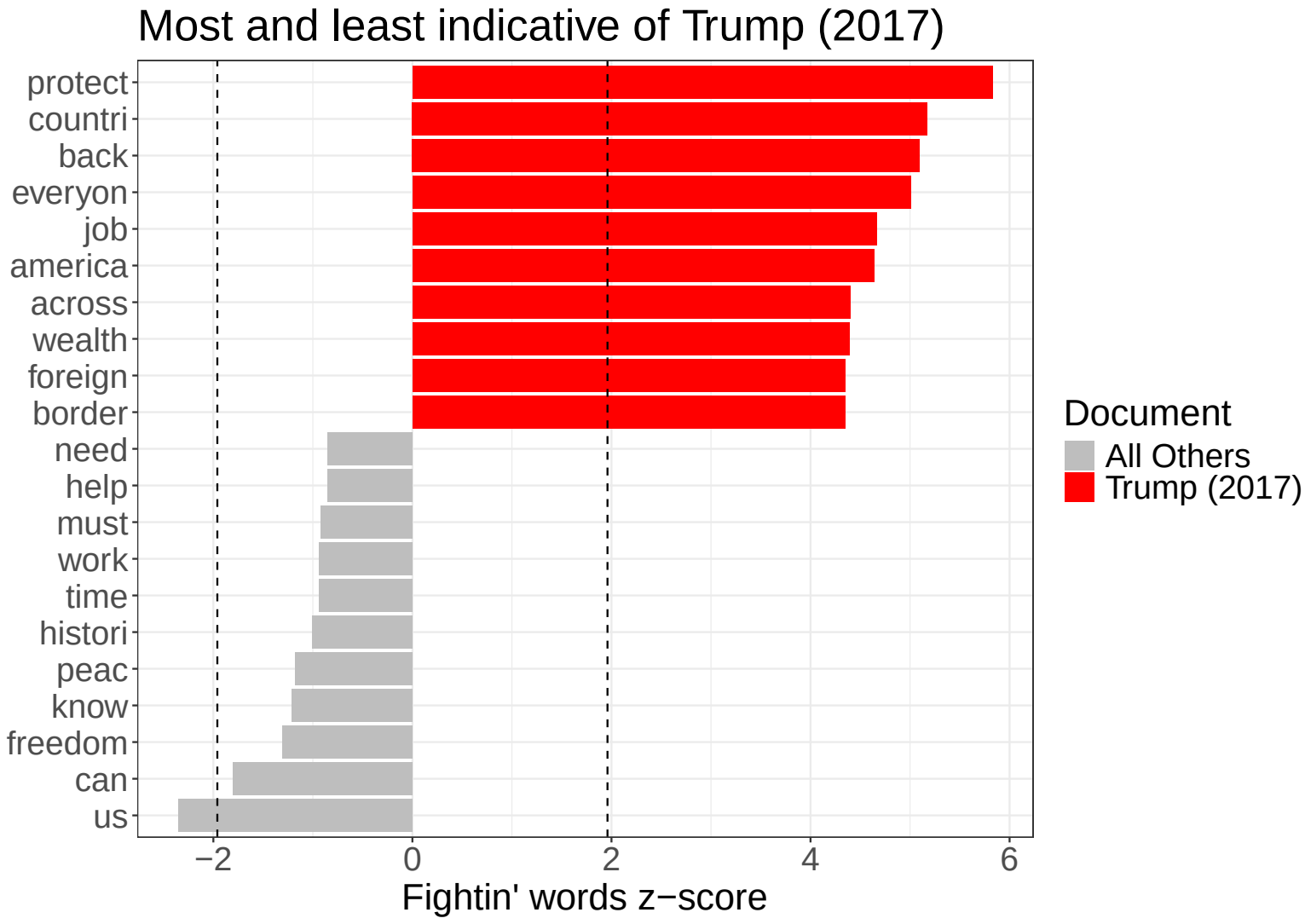
and

$$\mathrm{Var}\left(\hat{\delta}_{\mathtt{a}}^{(\mathcal{T}-\mathcal{O})}\right) = \frac{1}{20} + \frac{1}{168} \approx 0.0560$$

This yields a standardised z-score of

$$\hat{z}_j^{(\mathcal{T}-\mathcal{O})} \approx \frac{1.10}{\sqrt{0.0560}} = 4.65$$

Measuring word discrimination



Extra material on probabilistic models

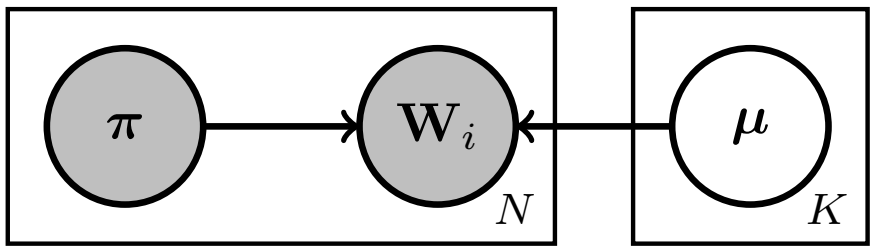
Mixture modeling

In this example, we conceptualised three separate probabilistic language models above: one for each author

We can unify them into a "meta" model, called a **mixture model**

Simple math trick: add a parameter $\boldsymbol{\pi}$ indicating which author wrote which document

- E.g., $\boldsymbol{\pi} = (1, 0, 0)$ could indicate Hamilton, $\boldsymbol{\pi} = (0, 1, 0)$ could indicate Jay and $\boldsymbol{\pi} = (0, 0, 1)$ could indicate Madison



Then, our "task" is to figure out: $\boldsymbol{\pi}_{\text{Unlabeled}}$

Where does μ come from?

In our probabilistic language model, μ is a *parameter*

➜ Parameters are parts of models that are just given/assumed

For our task, we used our data to try to estimate it

But, we can push the logic up one level and ask: in the model, where does μ come from?

➜ I.e., what is the data generating process for μ?

Why do we want to ask this? Helps us deal with some problems

One example is such as regularisation:

➜ Laplace smoothing outside the model (just add some fake data)

➜ Let's do smoothing *inside* the model

Where does μ come from?

We will add a **prior** to our model:

- ➜ Assume μ is sampled from a **Dirichlet distribution** with parameter α
- ➜ Parameter α is same length as μ
- ➜ So for our Federalist paper example, we need a prior α with three elements in it (one for each author)

Why this distribution?

- ➜ Makes the math really simple, but we'll mostly ignore technical details

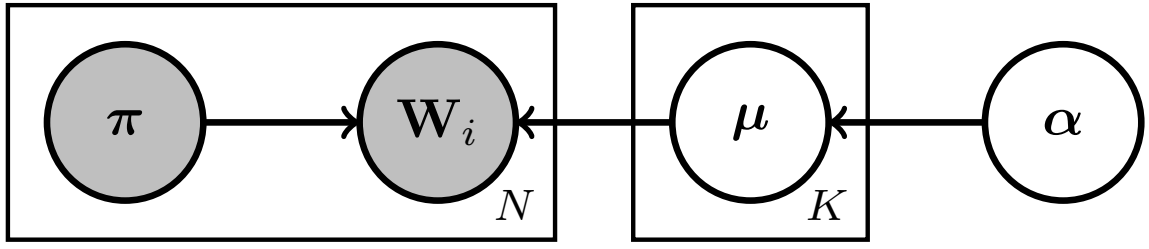
Model of language with a prior

New model of language:

1. Sample each author k's "writing style", $\boldsymbol{\mu}_k \sim \text{Dirichlet}(\boldsymbol{\alpha})$
2. Create a matrix $\boldsymbol{\mu}$ by staking each $\boldsymbol{\mu}_k$
3. Using these sampled $\boldsymbol{\mu}$ plus authorship indicator $\boldsymbol{\pi}$, draw documents:

$$\mathbf{W}_i | \boldsymbol{\mu}, \boldsymbol{\pi}_k \sim \text{Multinomial}(M_i, \boldsymbol{\mu}\boldsymbol{\pi}_k)$$

Graphically:



Estimating the model with the prior included

Our goal is still to figure out each writer k's $\boldsymbol{\mu}_k$: $\boldsymbol{\mu}_{\mathcal{H}}$, $\boldsymbol{\mu}_{\mathcal{J}}$, $\boldsymbol{\mu}_{\mathcal{M}}$

We can estimate it with our data, but using a modified formula that has the prior incorporated

- ➜ We *choose* the prior, i.e., choosing $\boldsymbol{\alpha}$
- ➜ Common to chose an uninformative one, where every element is the same (small) number, like 1

Then, under this model, the μ we estimate for a particular author k and feature j is

$$\hat{\mu}_{kj} = \frac{\alpha_k + W_{kj}}{\sum_k \alpha_k + \sum_j W_{kj}}$$

The prior has the mathematical effect of drawing any given estimate $\hat{\mu}_{kj}$ toward $\alpha_k / \sum_k \alpha_k$ — it regularises them!

Estimating the model with the prior included

For example, if we use a prior $\boldsymbol{\alpha} = (1, 1, 1)$, then our estimate for the probability Hamilton would use the word "by" is:

$$\hat{\mu}_{\mathcal{H},\text{by}} = \frac{1 + 859}{(1 + 1 + 1) + (859 + 102 + 374)} \approx 0.64$$

Turns out this is the same as Laplace smoothing

➜ But, depending on your application, it may not be!

We could go further and create a prior for $\boldsymbol{\alpha}$

Stacking "layers" on a probabilistic model is known as **hierarchical modeling**

➜ This has advantages uses in more complex settings, but we'll discuss when we need to

➜ For now, keep in mind that stacking a prior on $\boldsymbol{\mu}$ allowed us to regularise