Computational Political Science

Session 6

David Broska
Zeppelin University
March 9, 2021

Outline for today

1. **Supervised scaling**
    - Wordscores model
    - How it relates to Naive Bayes
    - Wordscore example
2. **Unsupervised Learning**
    - Basics of unsupervised scaling methods
    - Parametric scaling models: Wordfish
3. **Coding examples**
    - Scaling UK manifestos with Wordscores
    - Scaling Irish budget debate 2010 using Wordfish

Course schedule

Session	Date	Topic	Assignment	Due date
1	Feb 02	Overview and key concepts	-	-
2	Feb 09	Preprocessing and descriptive statistics	Formative	Feb 22 23:59:59
3	Feb 16	Dictionary methods	-	-
4	Feb 23	Machine learning for texts: Classification I	Summative 1	Mar 08 23:59:59
5	Mar 02	Machine learning for texts: Classification II	-	-
6	Mar 09	*Supervised and unsupervised scaling*	Summative 2	Mar 22 23:59:59
7	Mar 16	Similarity and clustering	-	-
8	Mar 23	Topic models	Summative 3	Apr 12 23:59:59
-	-	*Break*	-	-
9	Apr 13	Retrieving data from the web	-	-
10	Apr 20	Published applications	-	-
11	Apr 27	Project Presentations	-	-

Wordscores

Supervised and unsupervised learning

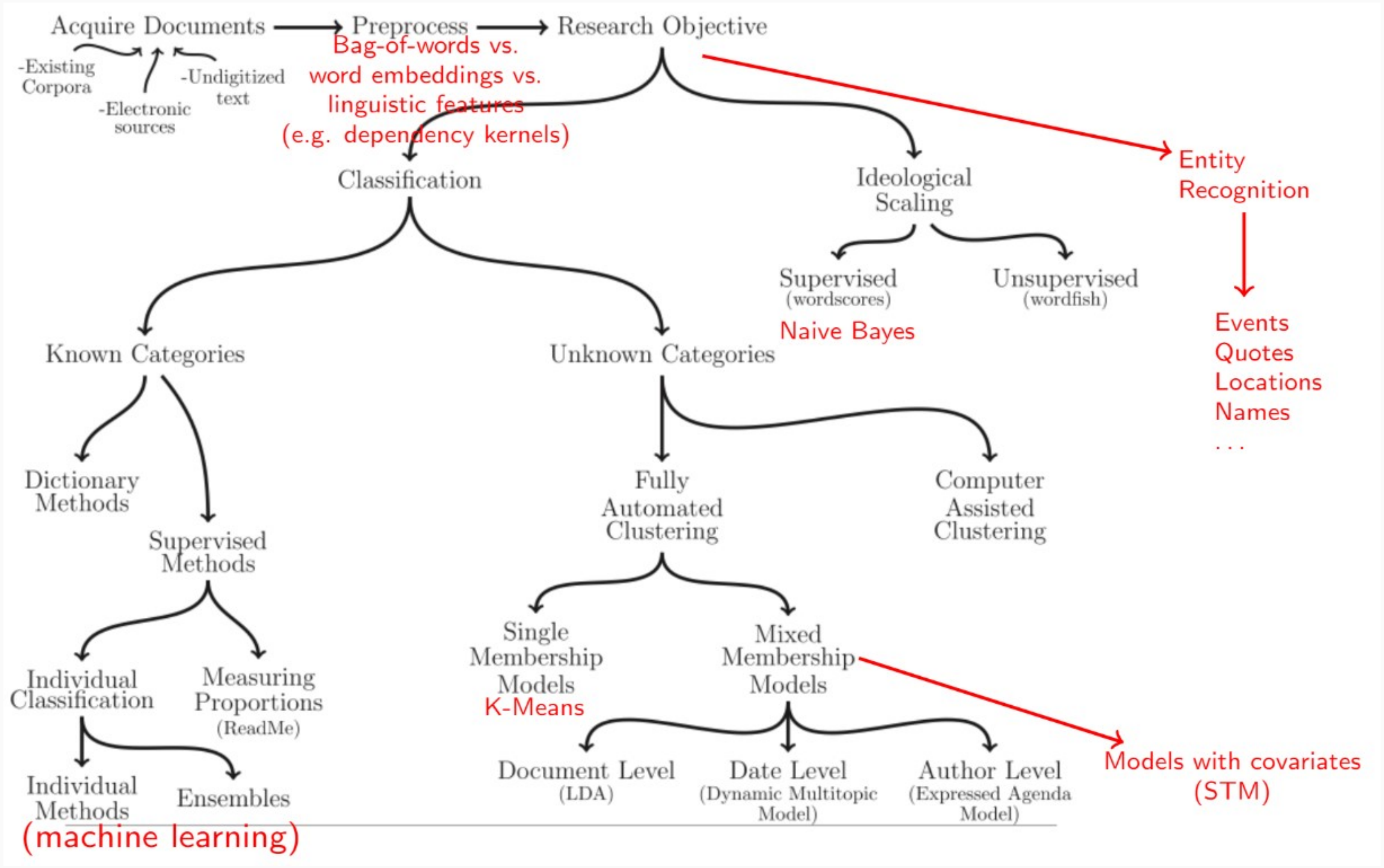


Fig. 1 in Grimmer and Stuart (2013)

From classification to scaling

Machine learning focuses on identifying classes (*classification*), while social science is typically interested in locating things on latent traits (*scaling*), for example:

- Policy positions on economic vs social dimension
- Inter- and intra-party differences
- Soft news vs hard news
- ...and any other continuous scale

But the two methods overlap and can be adapted - will demonstrate later using the Naive Bayes classifier

In fact, the class predictions for a collection of words from Naive Bayes can be adapted to scaling

Wordscores

Analogous to a "training set" and a "test set" in classification, the Wordscores method by Laver, Benoit, and Garry (2003) uses two sets of texts:

Reference texts

- texts about which we know something (a scalar dimensional score)

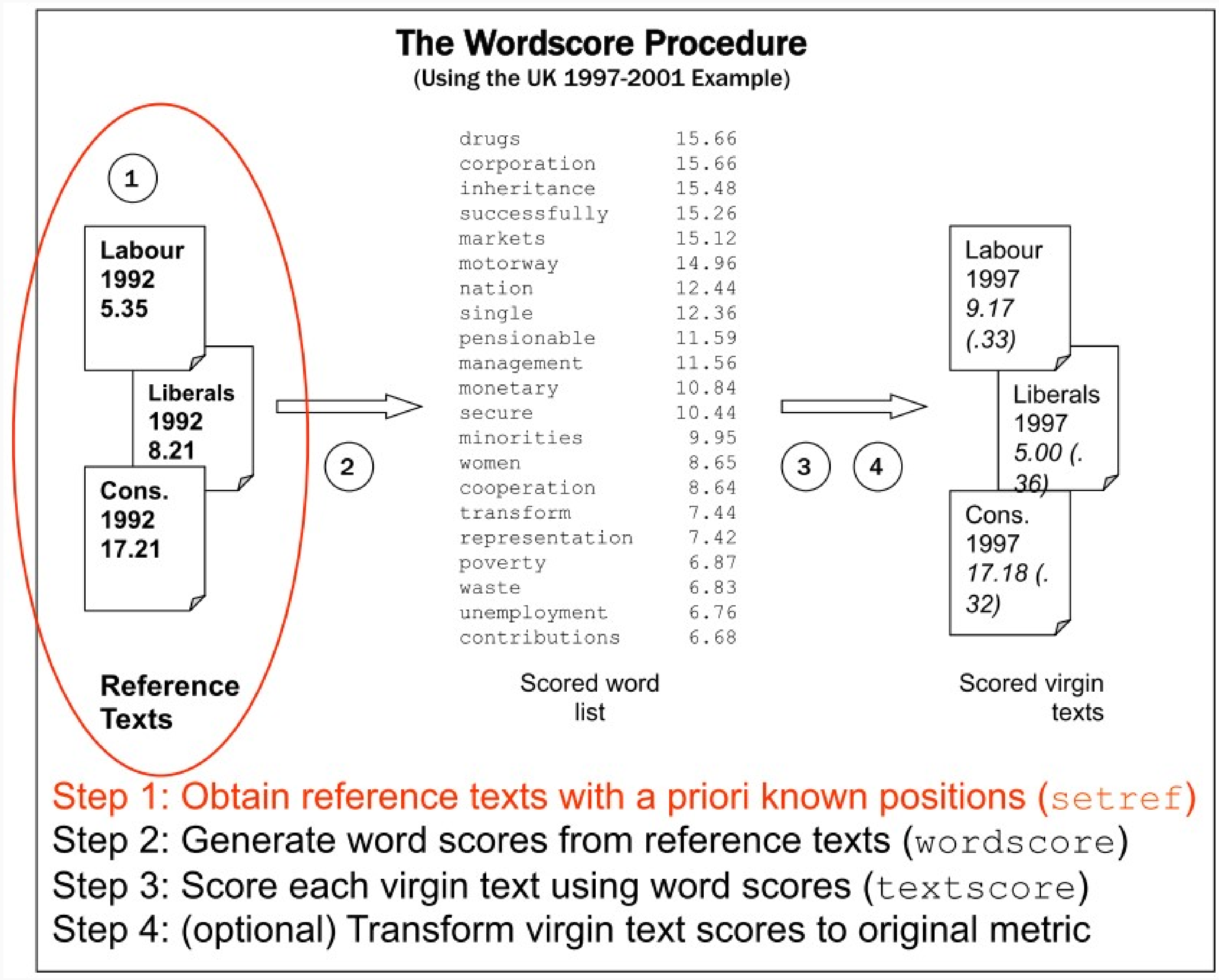
Virgin texts

- texts about which we know nothing (but whose dimensional score we'd like to know)

Basic procedure

1. Analyze reference texts to obtain a single "score" for every word
2. Use word scores to score virgin texts

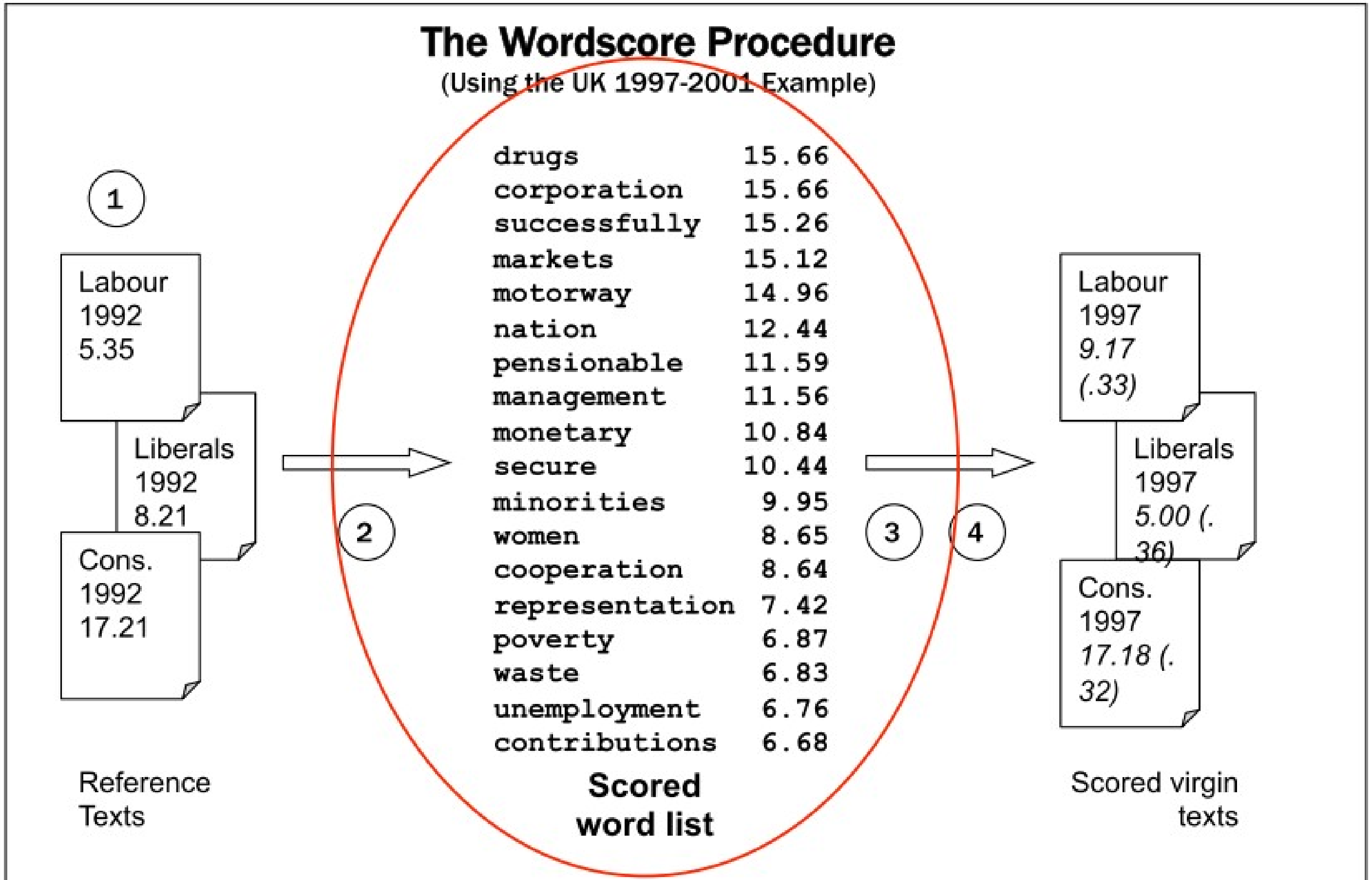
Wordscores procedure (I)



drugs	15.66
corporation	15.66
inheritance	15.48
successfully	15.26
markets	15.12
motorway	14.96
nation	12.44
single	12.36
pensionable	11.59
management	11.56
monetary	10.84
secure	10.44
minorities	9.95
women	8.65
cooperation	8.64
transform	7.44
representation	7.42
poverty	6.87
waste	6.83
unemployment	6.76
contributions	6.68

Step 1: Obtain reference texts with a priori known positions (`setref`)
Step 2: Generate word scores from reference texts (`wordscore`)
Step 3: Score each virgin text using word scores (`textscore`)
Step 4: (optional) Transform virgin text scores to original metric

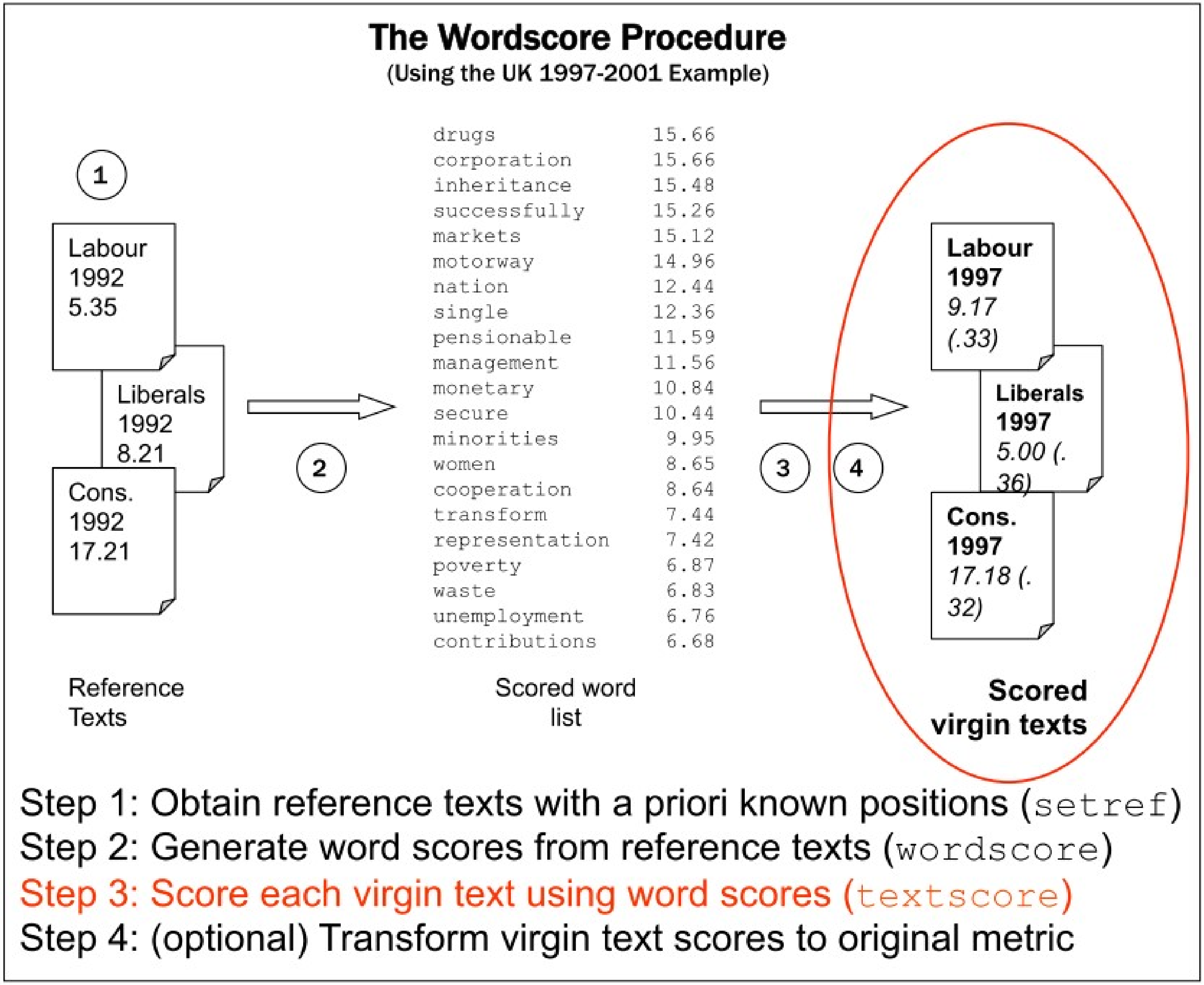
Wordscores procedure (II)



drugs	15.66
corporation	15.66
successfully	15.26
markets	15.12
motorway	14.96
nation	12.44
pensionable	11.59
management	11.56
monetary	10.84
secure	10.44
minorities	9.95
women	8.65
cooperation	8.64
representation	7.42
poverty	6.87
waste	6.83
unemployment	6.76
contributions	6.68

Step 1: Obtain reference texts with a priori known positions (`setref`)
Step 2: Generate word scores from reference texts (`wordscore`)
Step 3: Score each virgin text using word scores (`textscore`)
Step 4: (optional) Transform virgin text scores to original metric

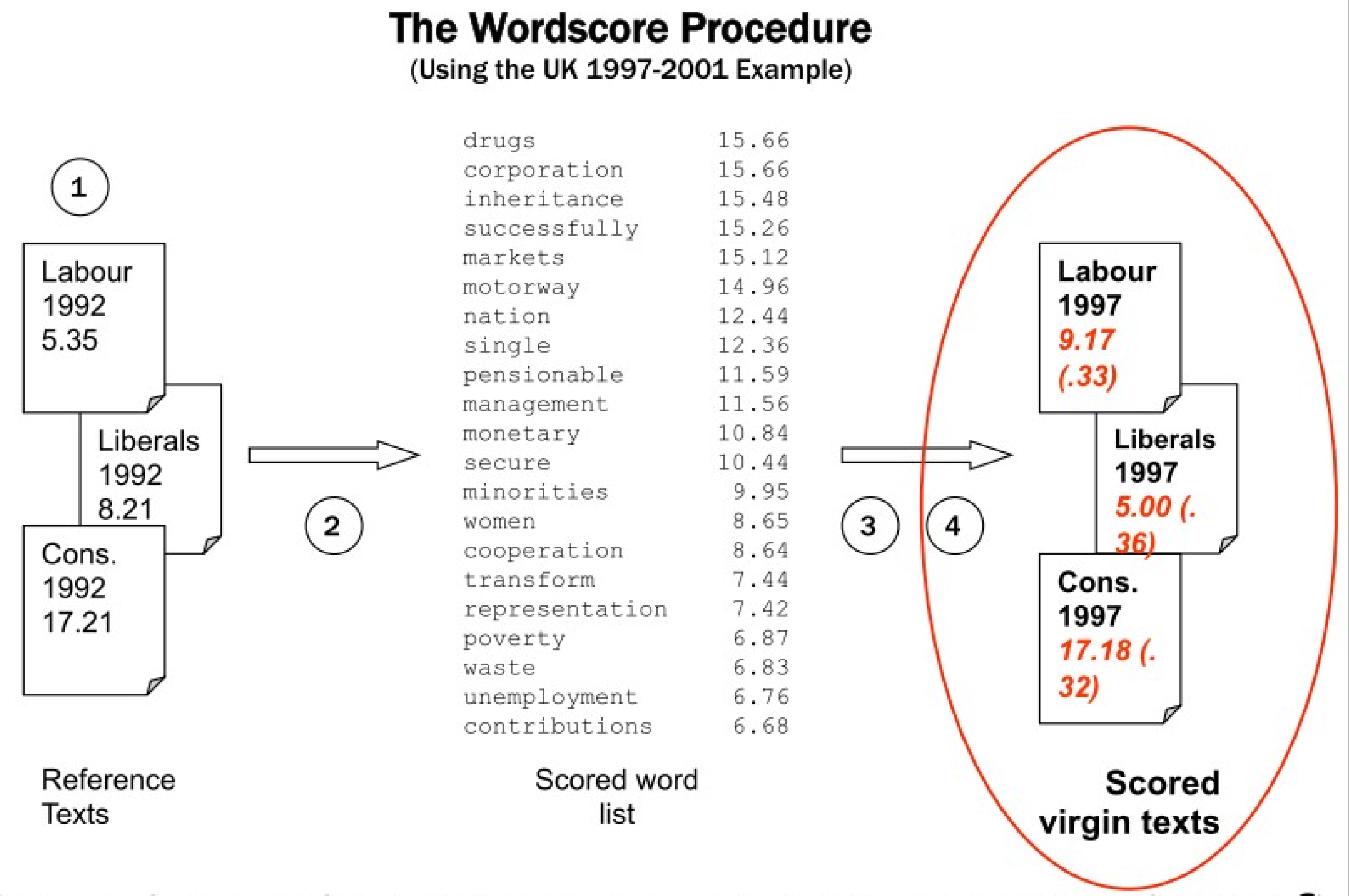
Wordscores procedure (III)



drugs	15.66
corporation	15.66
inheritance	15.48
successfully	15.26
markets	15.12
motorway	14.96
nation	12.44
single	12.36
pensionable	11.59
management	11.56
monetary	10.84
secure	10.44
minorities	9.95
women	8.65
cooperation	8.64
transform	7.44
representation	7.42
poverty	6.87
waste	6.83
unemployment	6.76
contributions	6.68

Step 1: Obtain reference texts with a priori known positions (`setref`)
Step 2: Generate word scores from reference texts (`wordscore`)
Step 3: Score each virgin text using word scores (`textscore`)
Step 4: (optional) Transform virgin text scores to original metric

Wordscores procedure (IV)



drugs	15.66
corporation	15.66
inheritance	15.48
successfully	15.26
markets	15.12
motorway	14.96
nation	12.44
single	12.36
pensionable	11.59
management	11.56
monetary	10.84
secure	10.44
minorities	9.95
women	8.65
cooperation	8.64
transform	7.44
representation	7.42
poverty	6.87
waste	6.83
unemployment	6.76
contributions	6.68

Step 1: Obtain reference texts with a priori known positions (`setref`)
Step 2: Generate word scores from reference texts (`wordscore`)
Step 3: Score each virgin text using word scores (`textscore`)
Step 4: (optional) Transform virgin text scores to original metric

Wordscore implementation

```r
# 4 texts with known and 3 texts with unknown category
txt <- c(k1 = "$ Win $",
         k2 = "$ Prize $",
         k3 = "Earn $ Easily",
         k4 = "Paypal 100 $",
         u1 = "$",
         u2 = "$ $",
         u3 = "Paypal 100 $ $")
x <- dfm(txt)
y <- c(1, 1, 1, -1, NA, NA, NA)
```

training dfm from references texts

	$	win	prize	earn	easily	paypal	100
k1	2	1	0	0	0	0	0
k2	2	0	1	0	0	0	0
k3	1	0	0	1	1	0	0
k4	1	0	0	0	0	1	1

training vector with known positions

y
1
1
1
-1

Wordscores

Compute probability of a reading document given a word

Start with a set of D reference texts, represented by an $D \times W$ document-feature matrix C_{dw} , where d indexes the document and w indexes the W total word types.

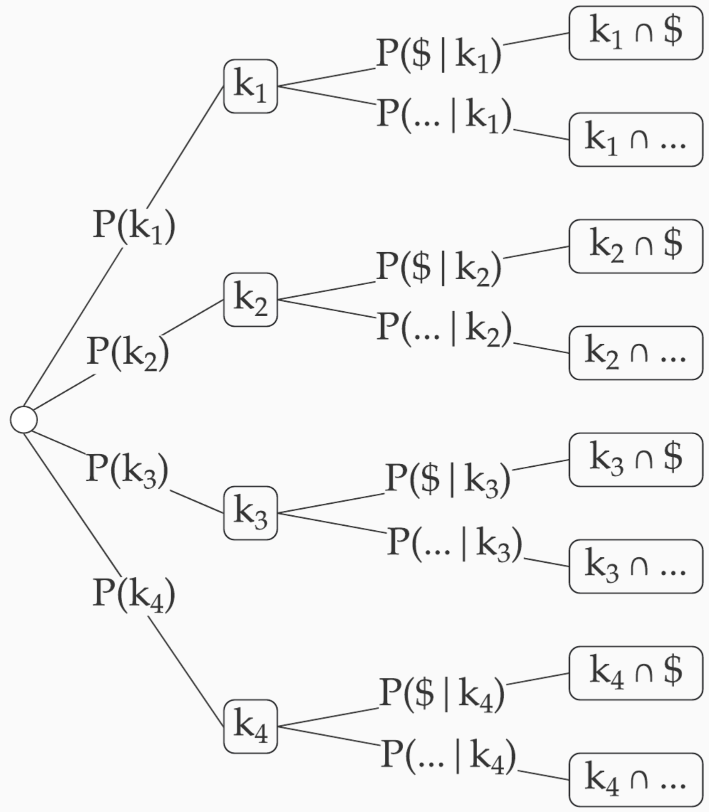
We normalize the document-feature matrix within each document by converting C_{ij} into a relative document-feature matrix (within document), by dividing C_{ij} by its word total marginals

Probability of word given the document

```r
( PwGd <- dfm_weight(x[1:4,],scheme="prop") )
```

```
##      features
## docs    $  win prize earn easily paypal  100
##   k1 0.67 0.33  0.00 0.00   0.00   0.00 0.00
##   k2 0.67 0.00  0.33 0.00   0.00   0.00 0.00
##   k3 0.33 0.00  0.00 0.33   0.33   0.00 0.00
##   k4 0.33 0.00  0.00 0.00   0.00   0.33 0.33
```

P(k_1 | $)



Uniform priors: P(k_1)=...=P(k_4)= ¼

If we only read "$" the probability of reading the document k_1 is ⅓.

Probability of word given the document:

	$	**win**	**prize**	**earn**	**easily**	**paypal**	**100**
k1	0.67	0.33	0.00	0.00	0.00	0.00	0.00
k2	0.67	0.00	0.33	0.00	0.00	0.00	0.00
k3	0.33	0.00	0.00	0.33	0.33	0.00	0.00
k4	0.33	0.00	0.00	0.00	0.00	0.33	0.33

$$P(k_1|\$)$$
$$= \frac{P(k_1)P(\$|k_1)}{P(k_1)P(\$|k_1) + ... + P(k_2)P(\$|k_4)}$$
$$= \frac{P(\$|k_1)}{P(\$|k_1) + ... + P(\$|k_4)}$$
$$= \frac{\frac{2}{3}}{\frac{2}{3} + \frac{2}{3} + \frac{1}{3} + \frac{1}{3}} = \frac{1}{3}$$

P(document | word)

Now let's compute all probabilities of reading a document given a word

```
PwGd # recall our matrix containing all P(word | document)
```

```
##     features
## docs    $  win prize earn easily paypal  100
##   k1 0.67 0.33  0.00 0.00   0.00   0.00 0.00
##   k2 0.67 0.00  0.33 0.00   0.00   0.00 0.00
##   k3 0.33 0.00  0.00 0.33   0.33   0.00 0.00
##   k4 0.33 0.00  0.00 0.00   0.00   0.33 0.33
```

```
# transpose PwGd matrix
( tPwGd <- t(PwGd) )
```

```
##          docs
## features   k1   k2   k3   k4
##   $      0.67 0.67 0.33 0.33
##   win    0.33 0.00 0.00 0.00
##   prize  0.00 0.33 0.00 0.00
##   earn   0.00 0.00 0.33 0.00
##   easily 0.00 0.00 0.33 0.00
##   paypal 0.00 0.00 0.00 0.33
```

```
# P(document | word)
( PdGw <- tPwGd / rowSums(tPwGd) )
```

```
##          docs
## features   k1   k2   k3   k4
##   $      0.33 0.33 0.17 0.17
##   win    1.00 0.00 0.00 0.00
##   prize  0.00 1.00 0.00 0.00
##   earn   0.00 0.00 1.00 0.00
##   easily 0.00 0.00 1.00 0.00
##   paypal 0.00 0.00 0.00 1.00
```

Scoring words

Compute a J-length "score" vector S for each word j as the average of each document i's scores a_i, weighted by each word's P_{ij} so that $S_j = \sum_i^I a_i P_{ij}$

```r
y[1:4] # the "a" vector with the positions of the document
```

```
## [1]  1  1  1 -1
```

```r
t(PdGw) * y[1:4] # transpose matrix so we can multiply PdGw with the doc positions
```

```
##      features
## docs      $ win prize earn easily paypal 100
##    k1  0.33   1     0    0      0      0   0
##    k2  0.33   0     1    0      0      0   0
##    k3  0.17   0     0    1      1      0   0
##    k4 -0.17   0     0    0      0     -1  -1
```

```r
( ws <- colSums( t(PdGw) * y[1:4] )) # then, sum up the result column-wise
```

```
##      $    win  prize   earn easily paypal    100
##   0.67   1.00   1.00   1.00   1.00  -1.00  -1.00
```

Scoring words

We obtain the scored words *also* by using matrix multiplication. In matrix algebra,

$$\underset{1\times J}{S} = \underset{1\times I}{a} \cdot \underset{I\times J}{P}$$

```
PdGw # P(document | word)
```

```
##          docs
## features   k1   k2   k3   k4
##    $      0.67 0.67 0.33 0.33
##    win    0.33 0.00 0.00 0.00
##    prize  0.00 0.33 0.00 0.00
##    earn   0.00 0.00 0.33 0.00
##    easily 0.00 0.00 0.33 0.00
##    paypal 0.00 0.00 0.00 0.33
##    100    0.00 0.00 0.00 0.33
```

```
y[1:4] # documents scale
```

```
## [1]  1  1  1 -1
```

```
 # matrix multiplication with P(document|words) and scores
( ws <- PdGw %*% y[1:4] )
```

```
##       $    win  prize   earn easily paypal    100
##    0.67   1.00   1.00   1.00   1.00  -1.00  -1.00
```

Scoring texts

The goal is to obtain a single score for any new text, relative to the reference texts

We do this by taking the mean of the scores of its words, weighted by their term frequency

```r
PwGd <- dfm_weight(x, scheme="prop")
t(PwGd) # transpose matrix
```

```
##          docs
## features    k1   k2   k3   k4 u1 u2   u3
##   $       0.67 0.67 0.33 0.33  1  1 0.50
##   win     0.33 0.00 0.00 0.00  0  0 0.00
##   prize   0.00 0.33 0.00 0.00  0  0 0.00
##   earn    0.00 0.00 0.33 0.00  0  0 0.00
##   easily  0.00 0.00 0.33 0.00  0  0 0.00
##   paypal  0.00 0.00 0.00 0.33  0  0 0.25
##   100     0.00 0.00 0.00 0.33  0  0 0.25
```

```r
# row-wise PwGd * score
t(PwGd) * ws[,1]
```

```
##          docs
## features    k1   k2   k3    k4   u1   u2
##   $       0.44 0.44 0.22  0.22 0.67 0.67
##   win     0.33 0.00 0.00  0.00 0.00 0.00
##   prize   0.00 0.33 0.00  0.00 0.00 0.00
##   earn    0.00 0.00 0.33  0.00 0.00 0.00
##   easily  0.00 0.00 0.33  0.00 0.00 0.00
##   paypal  0.00 0.00 0.00 -0.33 0.00 0.00
##   100     0.00 0.00 0.00 -0.33 0.00 0.00
```

```r
colSums( t(PwGd) * ws[,1] )
```

```
##    k1    k2    k3    k4    u1    u2    u3
##  0.78  0.78  0.89 -0.44  0.67  0.67 -0.17
```

Scoring texts

We obtain the scored words *also* by using matrix multiplication.

```
# matrix multiplication with P(word | document) and obtained wordscores
PwGd %*% ws
```

```
##    k1    k2    k3    k4    u1    u2    u3
##  0.78  0.78  0.89 -0.44  0.67  0.67 -0.17
```

Does this result make sense in the context of the spam example?

k1 (s)	k2 (s)	k3 (s)	k4 (¬s)	u1	u2	u3
$ Win $	$ Prize $	Earn $ Easily	Paypal 100 $	$	$ $	Paypal 100 $ $

Final remarks

- Note that new words outside of the set J may appear in the K virgin documents – these are simply ignored (because we have no information on their scores)
- Note also that nothing prohibits reference documents from also being scored as virgin documents

Using textmodel_wordscores()

For convenience we can use the quanteda function to obtain the above results

```
ws_mod <- textmodel_wordscores(x,y)
```

Wordscores

```
summary(ws_mod)
```

```
## textmodel_wordscores.dfm(x = x, y = y)

## (showing first 7 elements)
##      $    win  prize   earn easily paypal    100
##   0.67   1.00   1.00   1.00   1.00  -1.00  -1.00
```

Scaled documents

```
predict(ws_mod)
```

```
##    k1    k2    k3    k4    u1    u2    u3
##  0.78  0.78  0.89 -0.44  0.67  0.67 -0.17
```

Unsupervised scaling

Unsupervised scaling methods

Text gets converted into a quantitative matrix of features

- words, typically
- could be dictionary entries, or parts of speech

Documents are scaled based on similarity or distance in feature use but the fundamental problem is distance on which scale?

- Ideally, something we care about, e.g. policy positions, ideology, preferences, sentiment
- But often other dimensions (language, rhetoric style, authorship) are more predictive

First dimension in unsupervised scaling will capture main source of variation, whatever it is

Unlike supervised models, validation comes after estimating the model

Unsupervised scaling methods

Two main approaches

Parametric methods model feature occurrence according to some stochastic distribution, typically in the form of a measurement model

- for instance, model words as a multi-level Bernoulli distribution, or a Poisson distribution
- word effects and “positional” effects are unobserved parameters to be estimated
- e.g. Wordfish (Slapin and Proksch 2008) and Wordshoal (Lauderdale and Herzog 2016)

Non-parametric methods typically based on the Singular Value Decomposition of a matrix

- correspondence analysis
- factor analysis
- other (multi)dimensional scaling methods

Parametric scaling models

Wordfish (Slapin and Proksch 2008)

The goal is unsupervised scaling of ideological positions

The frequency with which politician $\boldsymbol{i}$ uses word $\boldsymbol{k}$ is drawn from a Poisson distribution:

$$w_{ik} \sim \text{Poisson}(\lambda_{ik})$$

$$\lambda_{ik} = \exp(\alpha_i + \psi_k + \beta_k \times \boldsymbol{\theta}_i)$$

with latent parameters:

$\boldsymbol{\alpha}_i$ is “loquaciousness” of politician $\boldsymbol{i}$ (document fixed effect, hence it’s associated with the party or politician)

ψ_k is frequency of word $\boldsymbol{k}$ (word fixed effect)

β_k is discrimination parameter of word $\boldsymbol{k}$

$\boldsymbol{\theta}_i$ is the politician's ideological position

Key intuition: controlling for document length and word frequency, words with negative $\boldsymbol{k}$ will tend to be used more often by politicians with negative $\boldsymbol{\theta}_i$ (and vice versa)

Wordfish (Slapin and Proksch 2008)

Why Poisson?

- Poisson distributed variables are bounded between $(0, \infty)$ and take on only discrete values 0, 1, 2, ..., ∞
- Exponential transformation: word counts are function of log document length and word frequency

$$
\begin{aligned}
w_{ik} &\sim \text{Poisson}(\lambda_{ik}) \\
\lambda_{ik} &= \exp(\alpha_i + \psi_k + \beta_k \times \theta_i) \\
\log(\lambda_{ik}) &= \alpha_i + \psi_k \times \theta_i
\end{aligned}
$$

How to estimate this model

Conditional maximum likelihood estimation:

- If we knew ψ and β (the word parameters) then we have a Poisson regression model
- If we knew α and β (the party/politician/document parameters) then we have a Poisson regression model too!
- So we alternate them and hope to converge to reasonable estimates for both
- Implemented in the quanteda package `textmodel_wordfish()`

An alternative is MCMC with a Bayesian formulation or variational inference using an Expectation-Maximization algorithm (Imai et al 2016)

Conditional max. likelihood for wordfish

Start by guessing the parameters (some guesses are better than other, e.g. SVD)

Algorithm:

1. Assume the current *legislator parameters* are correct and fit as a Poisson regression model
2. Assume the current *word parameters* are correct and fit as a Poisson regression model
3. Normalize θs to mean 0 and variance 1

Iterate until convergence (change in value below a certain threshold)

Identification

The *scale* and *direction* of θ is undetermined - like most models with latent variables

To identify the model in Wordfish

- Fix one α to zero to specify the left-right direction (Wordfish option 1)
- Fix one $\hat{\theta}$s to mean zero and variance 1 to specify the scale (Wordfish option 2)
- Fix two $\hat{\theta}$s to specify the direction and scale (Wordfish option 3 and Wordscores)

Note: Fixing two reference scores does not specify the policy domain, it just identifies the model

"Features" of parametric scaling

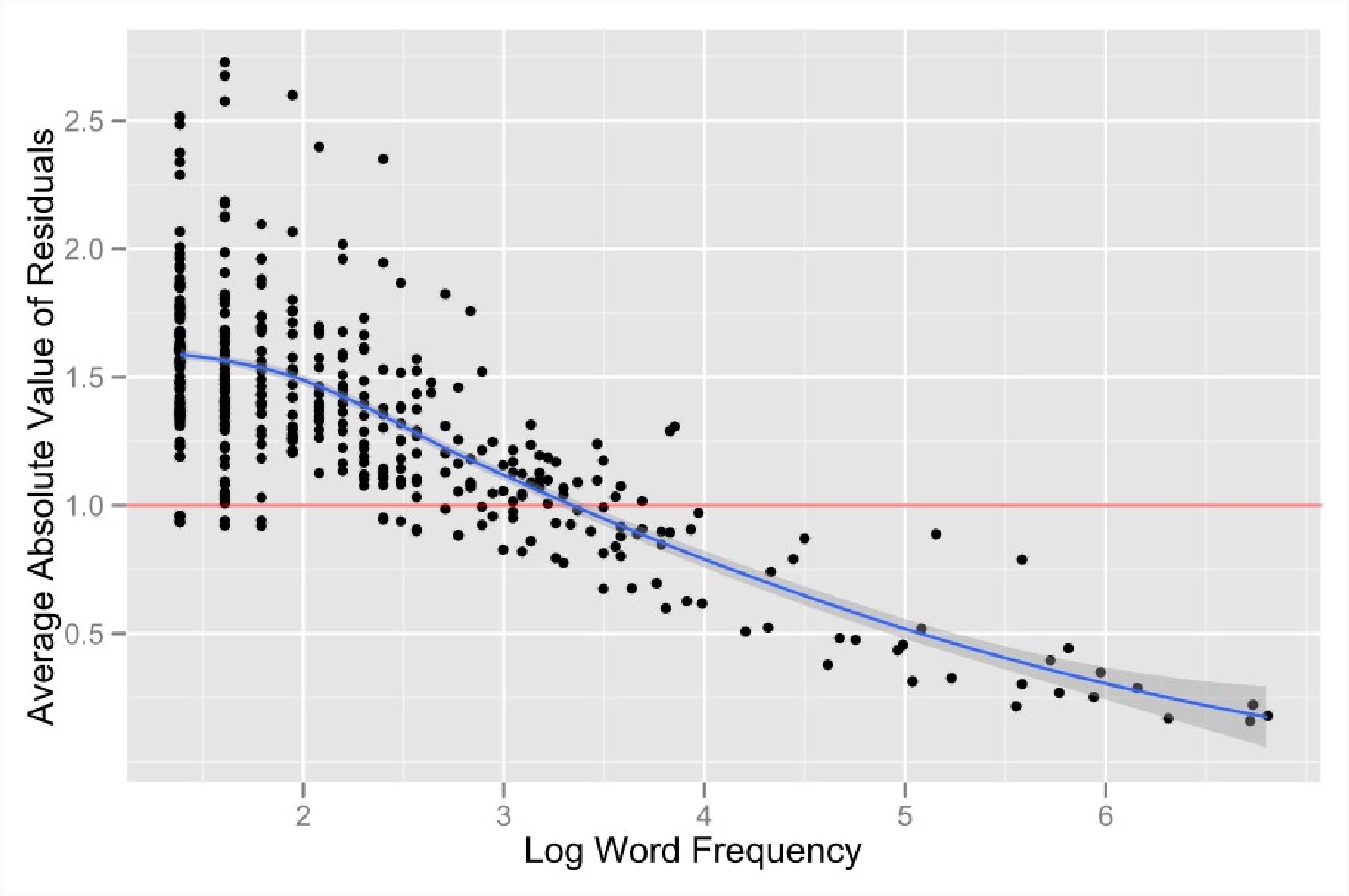
- Standard (statistical) inference about parameters
- Uncertainty accounting for parameters
- Distributional assumptions are made explicit (as part of the data generating process motivating the choice of stochastic distribution)
    - conditional independence
    - stochastic process (e.g. $\mathbf{E}(Y_{ij} = \mathbf{Var}(Y_{ij}) = \lambda_{ij}$)
- Permits hierarchical reparameterization (to add covariates)
- Generative model: given the estimated parameters, we could generate a document for any specified length

Some reasons why this model is wrong

- Violations of conditional independence:
    - Words occur in sequence (serial correlation)
    - Words occur in combinations (e.g. as collocations) “carbon tax” / “income tax” / “inheritance tax” / “capital gains tax” /”bank tax”
    - Legislative speech uses rhetoric that contains frequent synonyms and repetition for emphasis (e.g. “Yes we can!”)
- Heteroskedastic errors (variance not constant and equal to mean):
    - overdispersion when “informative” words tend to cluster together
    - underdispersion could (possibly) occur when words of high frequency are uninformative and have relatively low between-text variation (once length is considered)

Overdispersion

Overdispersion in German manifesto data (data taken from Slapin and Proksch 2008)



One solution to model overdispersion

Negative binomial model (Lo, Proksch, and Slapin 2014):

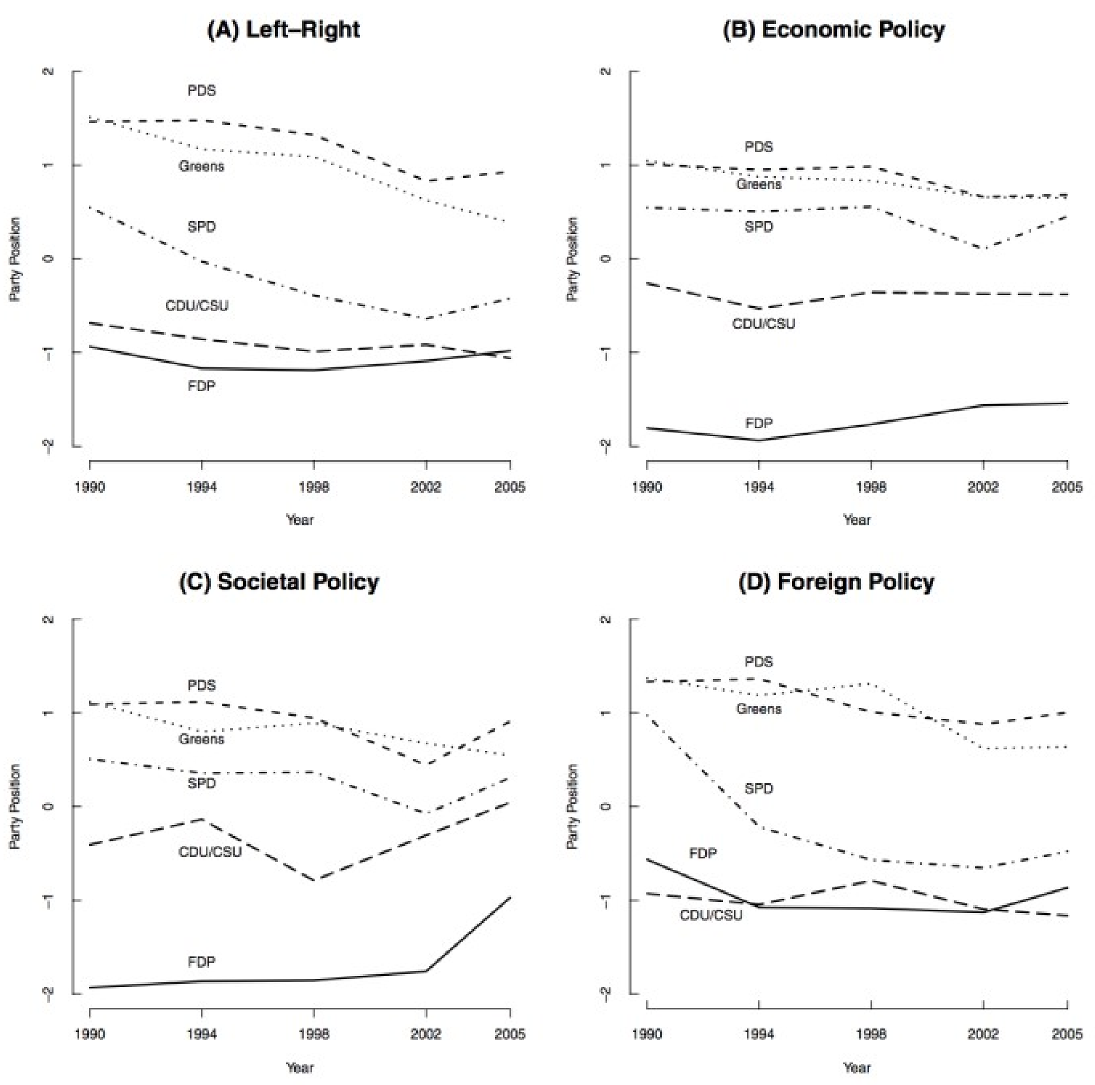
$$w_{ik} \sim \mathrm{NB}(r, \frac{\lambda_{ik}}{\lambda_{ik} + r_i})$$
$$\lambda_{ik} = \exp(\alpha_i + \psi_k + \beta_k \times \theta_i)$$

where ri is a variance inflation parameter that varies across documents.

It can have a substantive interpretation (ideological ambiguity), e.g. when a party emphasizes an issue but fails to mention key words associated with it that a party with similar ideology mentions.

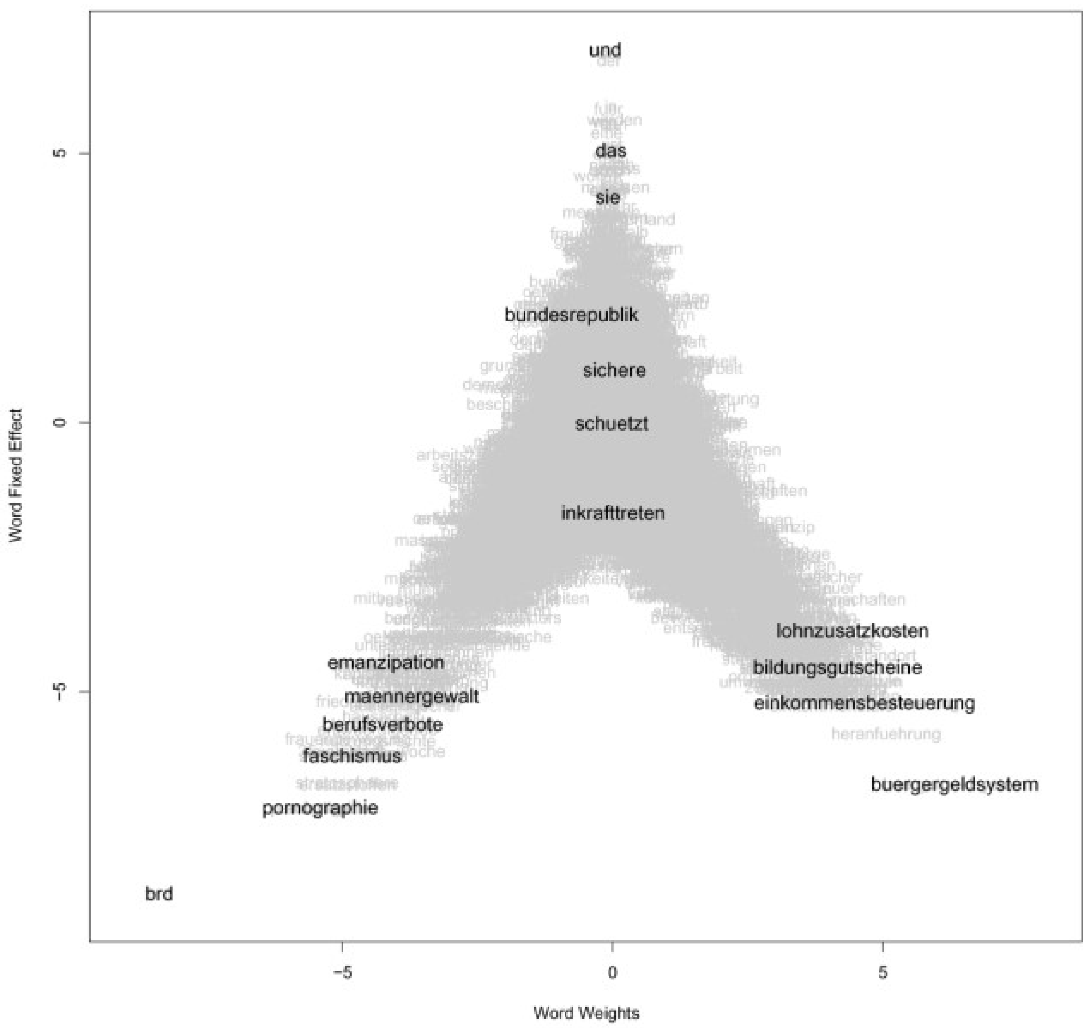
Example from Slapin and Proksch 2008

FIGURE 1 Estimated Party Positions in Germany, 1990–2005



Example from Slapin and Proksch 2008

FIGURE 2 **Word Weights vs. Word Fixed Effects. Left-Right Dimension, Germany 1990–2005 (Translations given in text)**



Example from Slapin and Proksch 2008

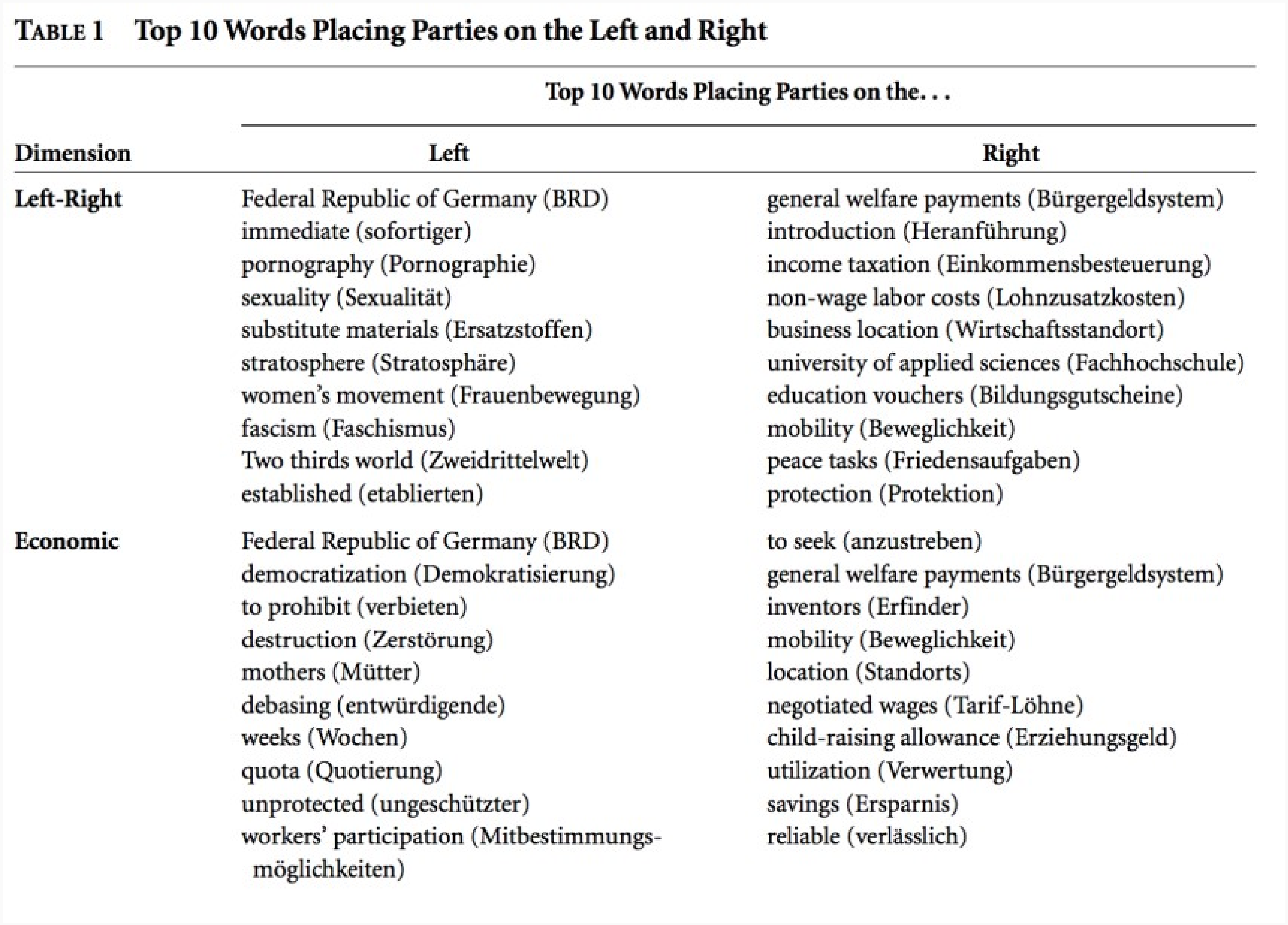


TABLE 1 Top 10 Words Placing Parties on the Left and Right

	Top 10 Words Placing Parties on the...	
Dimension	**Left**	**Right**
Left-Right	Federal Republic of Germany (BRD)	general welfare payments (Bürgergeldsystem)
	immediate (sofortiger)	introduction (Heranführung)
	pornography (Pornographie)	income taxation (Einkommensbesteuerung)
	sexuality (Sexualität)	non-wage labor costs (Lohnzusatzkosten)
	substitute materials (Ersatzstoffen)	business location (Wirtschaftsstandort)
	stratosphere (Stratosphäre)	university of applied sciences (Fachhochschule)
	women's movement (Frauenbewegung)	education vouchers (Bildungsgutscheine)
	fascism (Faschismus)	mobility (Beweglichkeit)
	Two thirds world (Zweidrittelwelt)	peace tasks (Friedensaufgaben)
	established (etablierten)	protection (Protektion)
Economic	Federal Republic of Germany (BRD)	to seek (anzustreben)
	democratization (Demokratisierung)	general welfare payments (Bürgergeldsystem)
	to prohibit (verbieten)	inventors (Erfinder)
	destruction (Zerstörung)	mobility (Beweglichkeit)
	mothers (Mütter)	location (Standorts)
	debasing (entwürdigende)	negotiated wages (Tarif-Löhne)
	weeks (Wochen)	child-raising allowance (Erziehungsgeld)
	quota (Quotierung)	utilization (Verwertung)
	unprotected (ungeschützter)	savings (Ersparnis)
	workers' participation (Mitbestimmungs-möglichkeiten)	reliable (verlässlich)

Coding exercise