Quantitative text analysis: Machine Learning for Text

Blake Miller

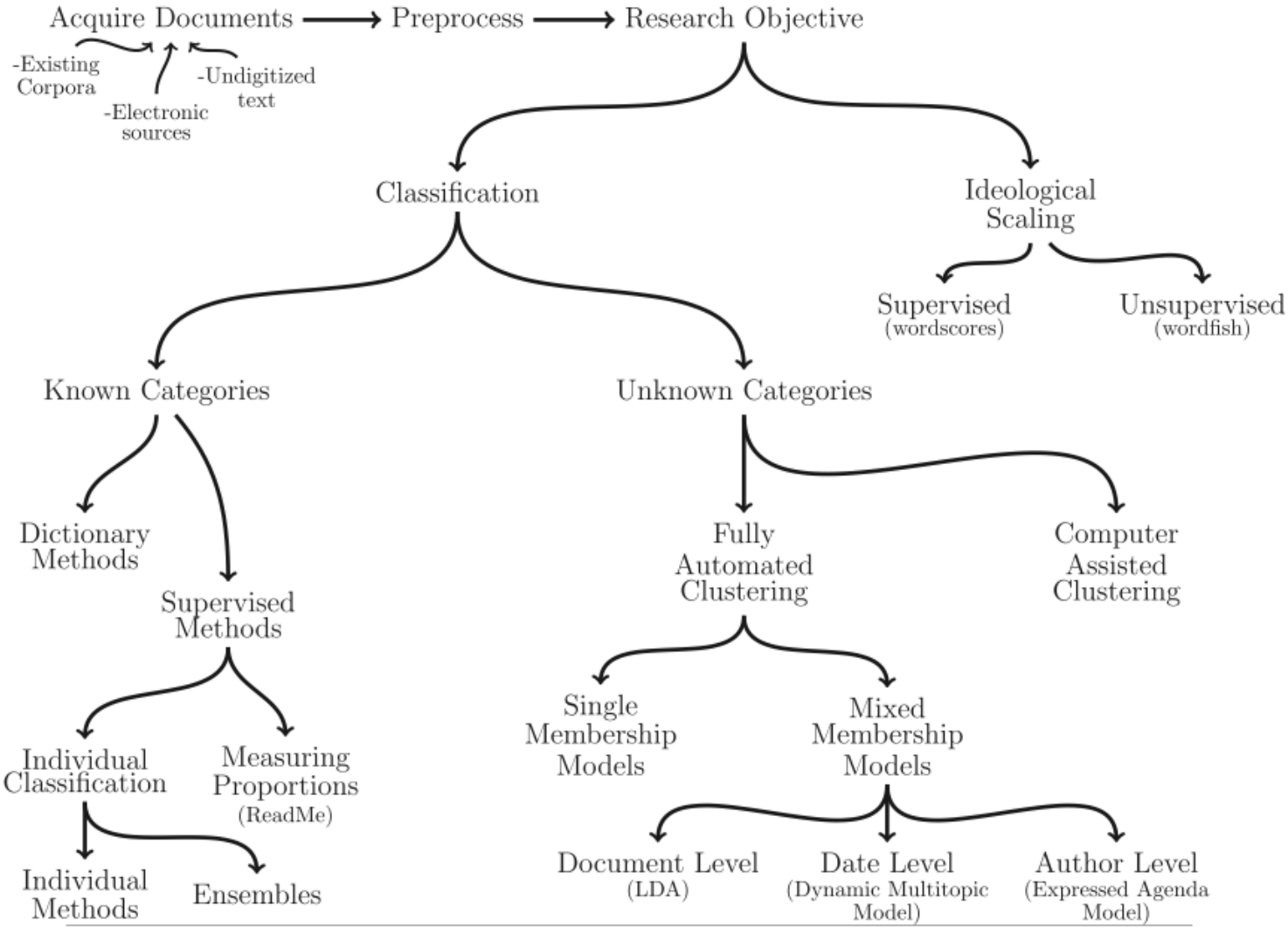
MY 459: Quantitative Text Analysis

February 17, 2019

Course website: lse-my459.github.io

1. Overview and Fundamentals
2. Descriptive Statistical Methods for Text Analysis
3. Automated Dictionary Methods
4. Machine Learning for Texts
5. Supervised Scaling Models for Texts
6. *Reading Week*
7. Unsupervised Models for Scaling Texts
8. Similarity and Clustering Methods
9. Topic models
10. Word embeddings
11. Working with Social Media

Overview of text as data methods



Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

Types of classifiers

General thoughts:

- Trade-off between accuracy and interpretability
- Hyperparameters need to be cross-validated

Frequently used classifiers:

- Naive Bayes
- Regularized regression
- Support Vector Machines (SVM)
- Deep learning: Convolutional Neural Networks (CNN), Long short-term memory networks (LSTM), etc.
- Ensemble methods

Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
  - Support vector machines
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

Supervised Learning Review

- X: data matrix, each column is called a feature, idependent variable, input variable (these can be words, n-grams, characters, etc.)
- Y: dependent variable/target
- Objective: Approximate the function of relationship between X and Y, make sure this function generalizes well to new data
- Regression: Y is a number
  - Linear regression is a very specialized case of regression
  - There are non-linear regression algorithms such as tree regression
- Classification: Y is a value in an unordered set (i.e. $y_i \in \{\text{ordinary}, \text{bot}, \text{troll}, \text{cyborg}\}$)
- Training data: Pairs $(x_1, y_1), \ldots, (x_N, y_N)$ used to "train" a model that will later predict y when given unseen test cases.
  - What are important/informative features?
  - How good are the model's predictions?/How useful is the model?

The Process of Supervised Learning For Text Data

1. Gather documents
2. Preprocess documents
3. Describe the documents
4. Transform the documents (to a dfm)
5. Specify a model
6. Train the model
7. Evaluate model performance

Multinomial Bayes model of Class given a Word

Consider J word types distributed across N documents, each assigned one of K classes.

At the word level, Bayes Theorem tells us that:

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

For two classes, this can be expressed as

$$= \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})} \tag{1}$$

Multinomial Bayes model of Class given a Word
Class-conditional word likelihoods

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- The word likelihood within class
- The maximum likelihood estimate is simply the proportion of times that word j occurs in class k, but it is more common to use Laplace smoothing by adding 1 to each observed count within class

Multinomial Bayes model of Class given a Word
Word probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

- This represents the word probability from the training corpus
- Usually uninteresting, since it is constant for the training data; is independent of the class k.
- This is called a scaling factor and is only needed to compute posteriors on a probability scale.

Multinomial Bayes model of Class given a Word
Class prior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the class prior probability
- Machine learning typically takes this as the document frequency in the training set

Multinomial Bayes model of Class given a Word
Class posterior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the posterior probability of membership in class k for word j
- Key for the classifier: in new documents, we only observe word distributions and want to predict class

Moving to the document level

- The "Naive" Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a "test" document, to produce:

$$P(c_k|d) = P(c_k) \prod_j^J \frac{P(w_j|c_k)}{P(w_j)}$$

$$P(c_k|d) \propto P(c_k) \prod_j^J P(w_j|c_k)$$

- This is why we call it "naive": because it (wrongly) assumes:
  - conditional independence of word counts given the class.
    - If the word "weather" appears in a document, it is more likely that we will see words like "forecast" and "report"
  - positional independence of word counts (bag of words assumption)
    - It is meaningful for words like "not" and "good" to be close together.
    - "Peace, no more war", "War, no more peace" (Source: Arthur Spirling)

Naive Bayes Classification Example

(From Manning, Raghavan and Schütze, *Introduction to Information Retrieval*)

▶ **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Naive Bayes Classification Example

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{Tokyo}|\overline{c}) = \hat{P}(\text{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Thus, the classifier assigns the test document to $c = \textit{China}$. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
  - Support vector machines
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

Regularized regression

Assume we have:

- $i = 1, 2, \ldots, N$ documents
- Each document i is in class $y_i = 0$ or $y_i = 1$
- $j = 1, 2, \ldots, J$ unique features
- And x_{ij} as the count of feature j in document i

We could build a linear regression model as a classifier, using the values of β_0, β_1, ..., β_J that minimize:

$$RSS = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2$$

But can we?

- If $J > N$, OLS does not have a unique solution
- Even with $N > J$, OLS has low bias/high variance (overfitting)

Regularized regression

What can we do? Add a penalty for model complexity, such that we now minimize:

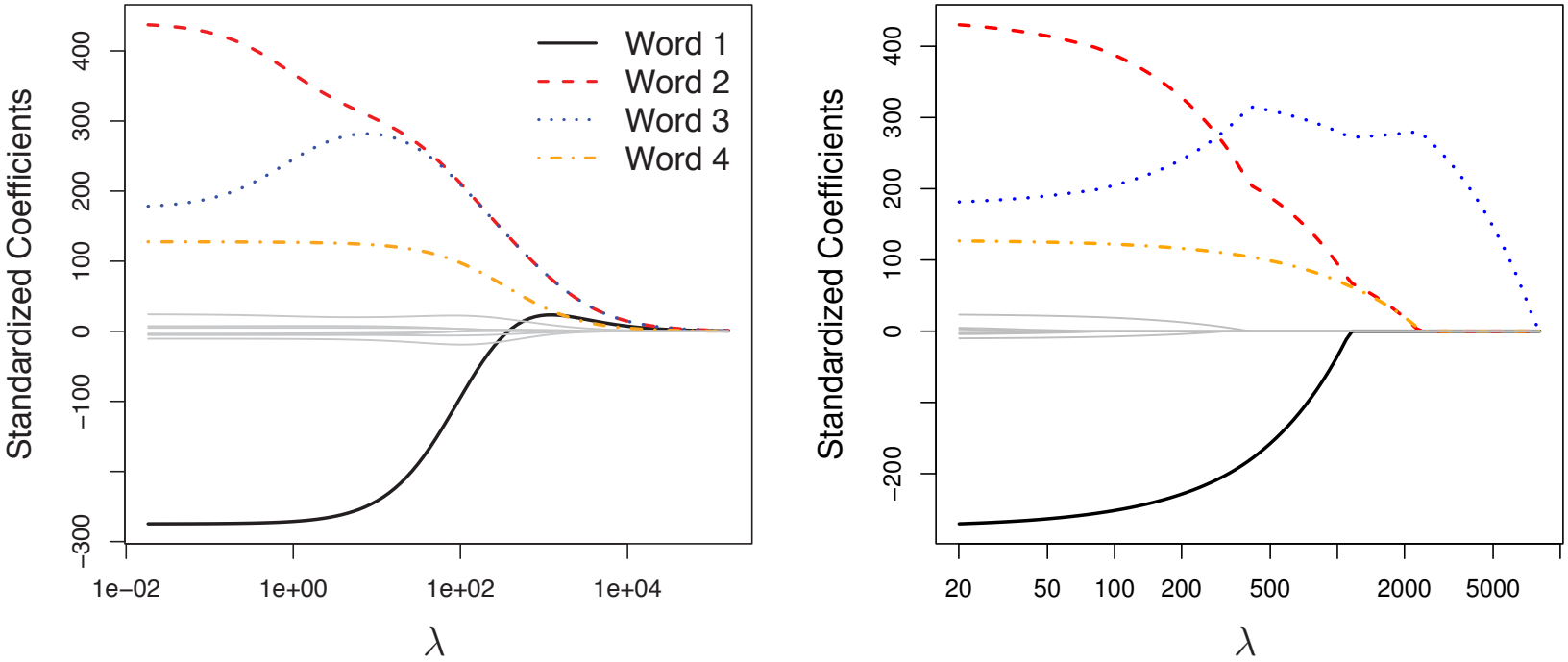
$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{J}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{J}\beta_j^2 \rightarrow \text{ridge regression}$$

or

$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{J}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{J}|\beta_j| \rightarrow \text{lasso regression}$$

where λ is the **penalty parameter** (to be estimated)

Visualizing Lasso vs. Ridge Regression



- We penalize large coefficients more as the size of λ increases
- Ridge regression shrinks some parameters close to zero but never quite gets there.
- Lasso shrinks some parameters to exactly zero.

Lasso and Ridge as Constrained Optimization

- Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?
- One can show that the lasso and ridge regression coefficient estimates solve the problems below (λ may now look familiar if you recall Lagrange multipliers from calculus)

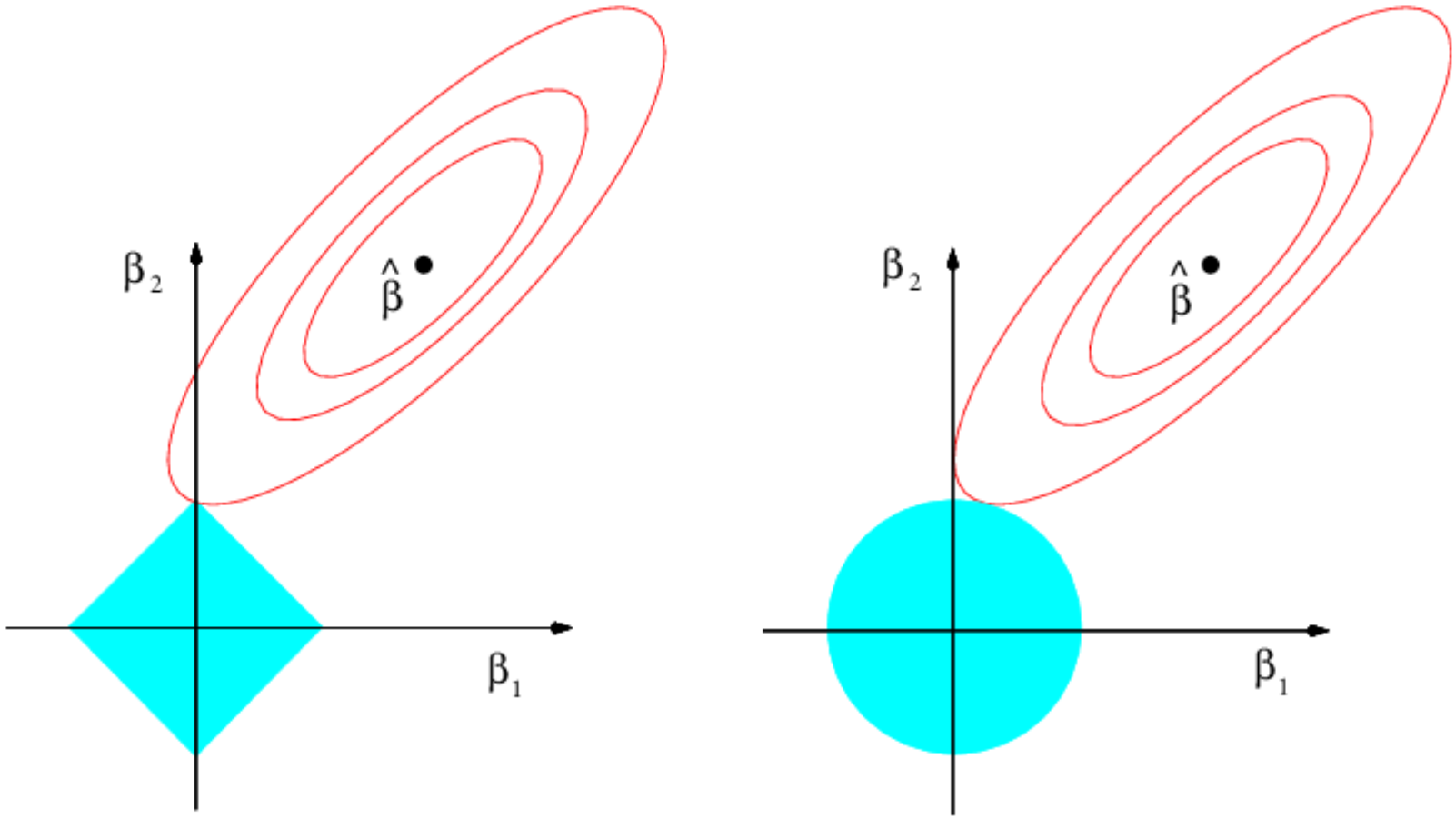
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| < s$$

and

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 < s$$

respectively.

Visualizing Lasso vs. Ridge Regression



Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

Regularized regression

Why the penalty (shrinkage)?

- Reduces the variance
- Identifies the model if J $>$ N
- Some coefficients become zero (feature selection)

The penalty can take different forms:

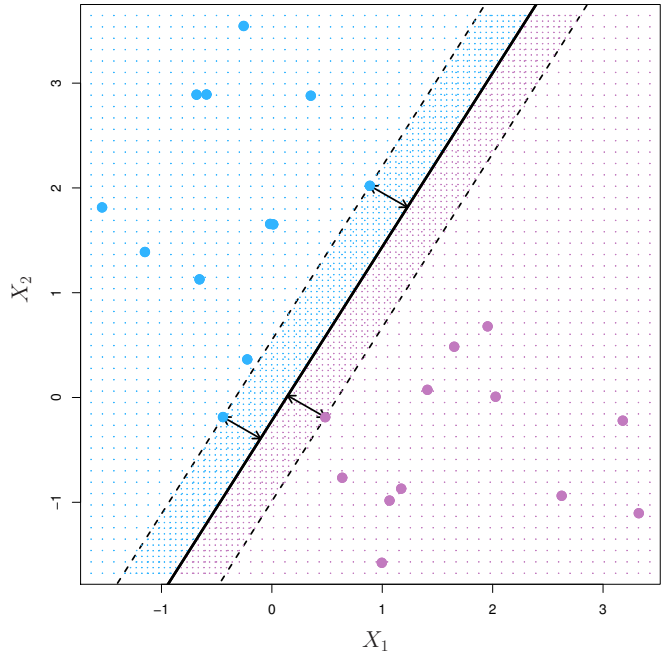
- Ridge regression: $\lambda \sum_{j=1}^{J} \beta_j^2$ with $\lambda > 0$; and when $\lambda = 0$ becomes OLS
- Lasso $\lambda \sum_{j=1}^{J} |\beta_j|$ where some coefficients become zero.
- Elastic Net: $\lambda_1 \sum_{j=1}^{J} \beta_j^2 + \lambda_2 \sum_{j=1}^{J} |\beta_j|$ (best of both worlds?)

How to find best value of λ? Cross-validation.
Evaluation: regularized regression is easy to interpret, but often outperformed by more complex methods.

Outline

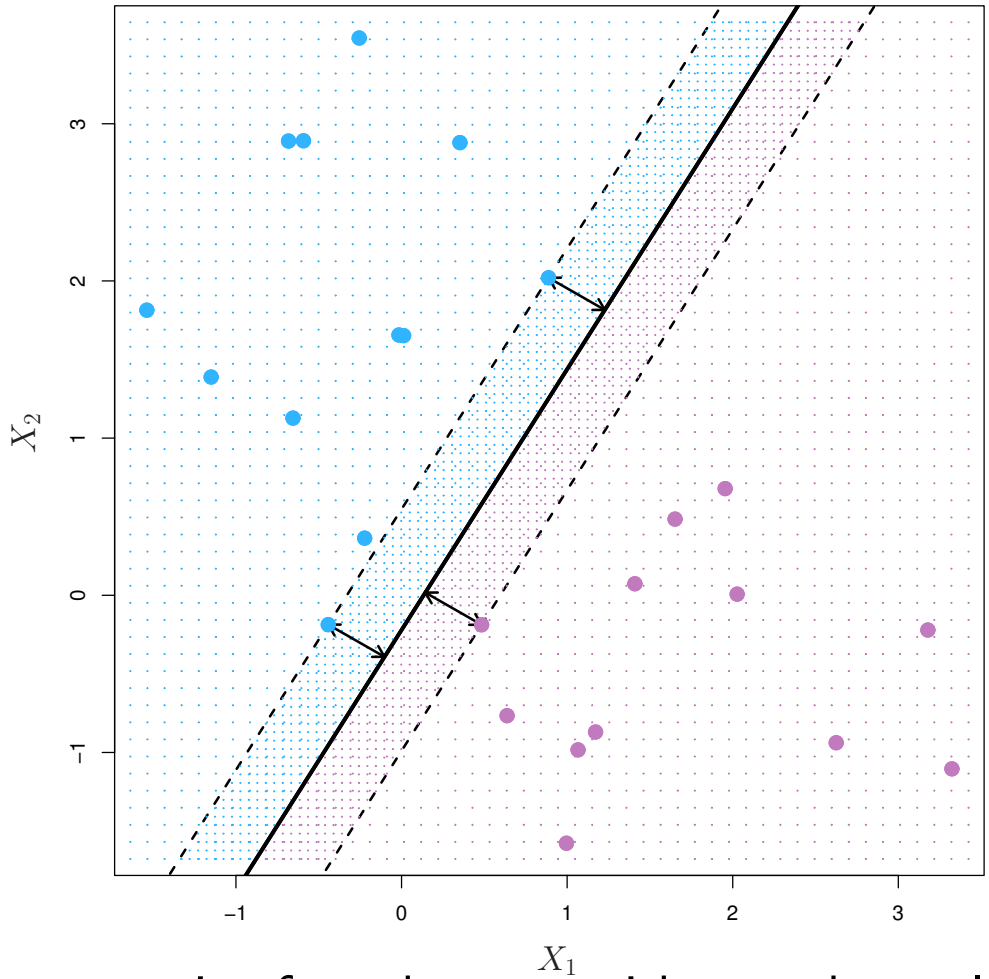
- Types of classifiers:
  - Naive Bayes
  - Regularized regression
  - Support vector machines
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

A Very Brief Intro to Support Vector Machines



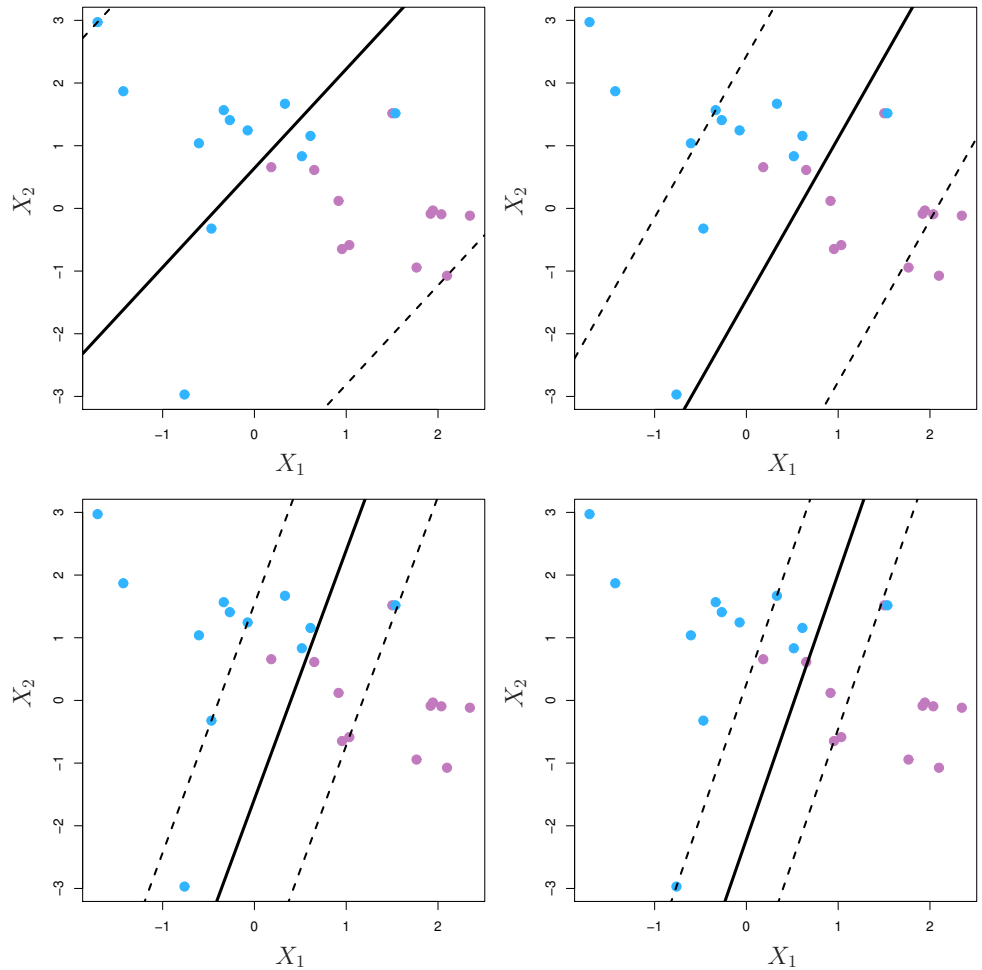
- Consider the simplest case of two linearly separable classes (separable by a line/plane/hyperplane)
- Devise a method for finding a "good" separating hyperplane
- We will not go into details in this class, but if you are interested in learning more, check out An Introduction to Statistical Learning by Gareth et. al.: `http://faculty.marshall.usc.edu/gareth-james/ISL/`

Finding a good hyperplane to separable classes



- In some cases, as is often the case with text data, classes are linearly separable.
- A "good" hyperplane maximizes a margin between the two classes.
- When data are not linearly separable, we need to allow for some error (training observations on the wrong side of the hyperplane)

Allowing for some points to be on the wrong side



- Hyperparameter C, chosen using cross-validation, is a total budget for errors.
- Changing the value of C affects the fit of the model. In the figure above, the value of C decreases from top left to bottom right.

Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

From Classification to Scaling

- Machine learning focuses on identifying classes (classification), while social science is typically interested in locating things on latent traits (scaling), e.g.:
  - Policy positions on economic vs social dimension
  - Inter- and intra-party differences
  - Soft news vs hard news
  - Petitioner vs respondent in legal briefs
  - ...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 NB can be adapted to scaling

Outline

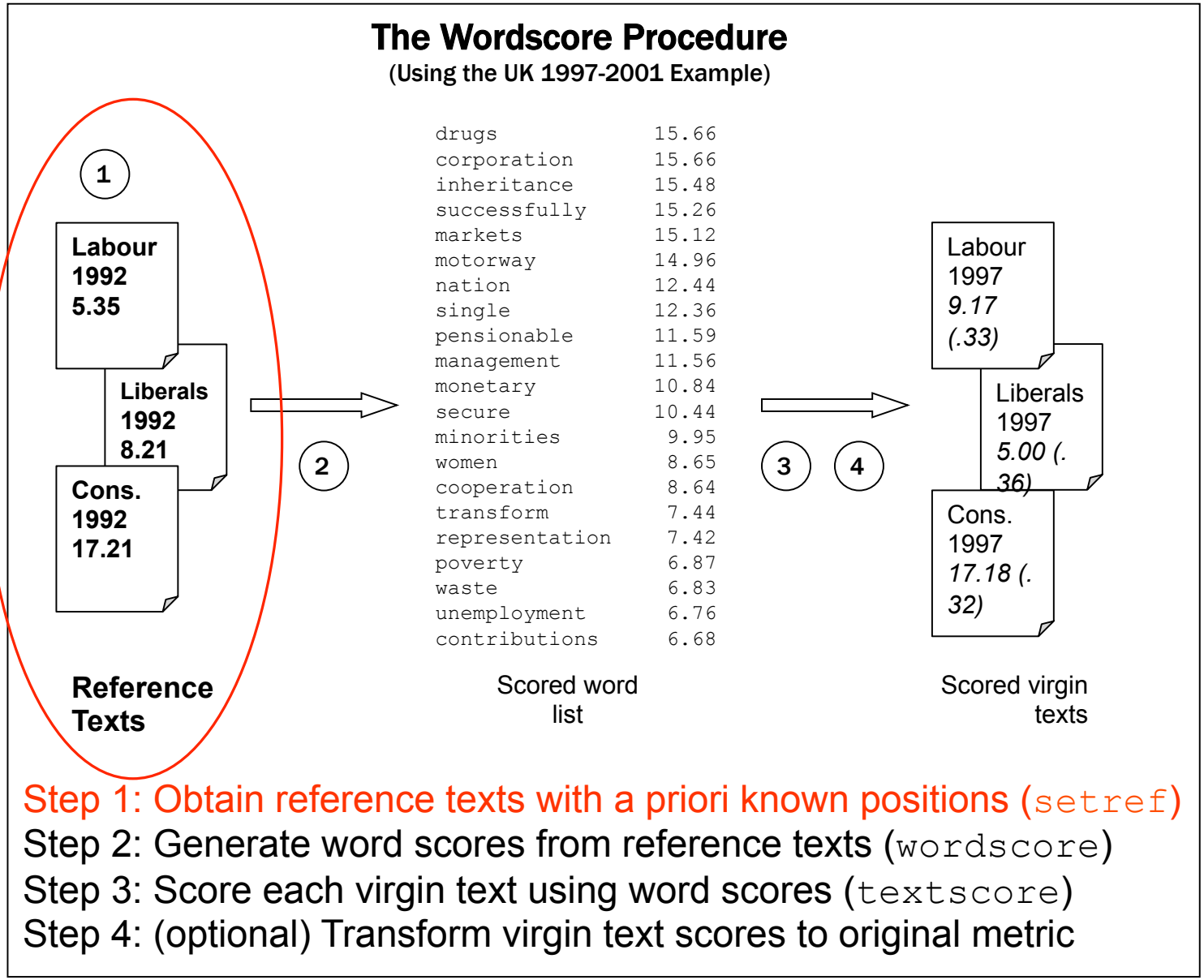
- Types of classifiers:
  - Naive Bayes
  - Regularized regression
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

Supervised scaling methods

Wordscores method (Laver, Benoit & Garry, 2003):

- 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)
- These are analogous to a "training set" and a "test set" in classification
- Basic procedure:
  1. Analyze reference texts to obtain word scores
  2. Use word scores to score virgin texts

Wordscores Procedure



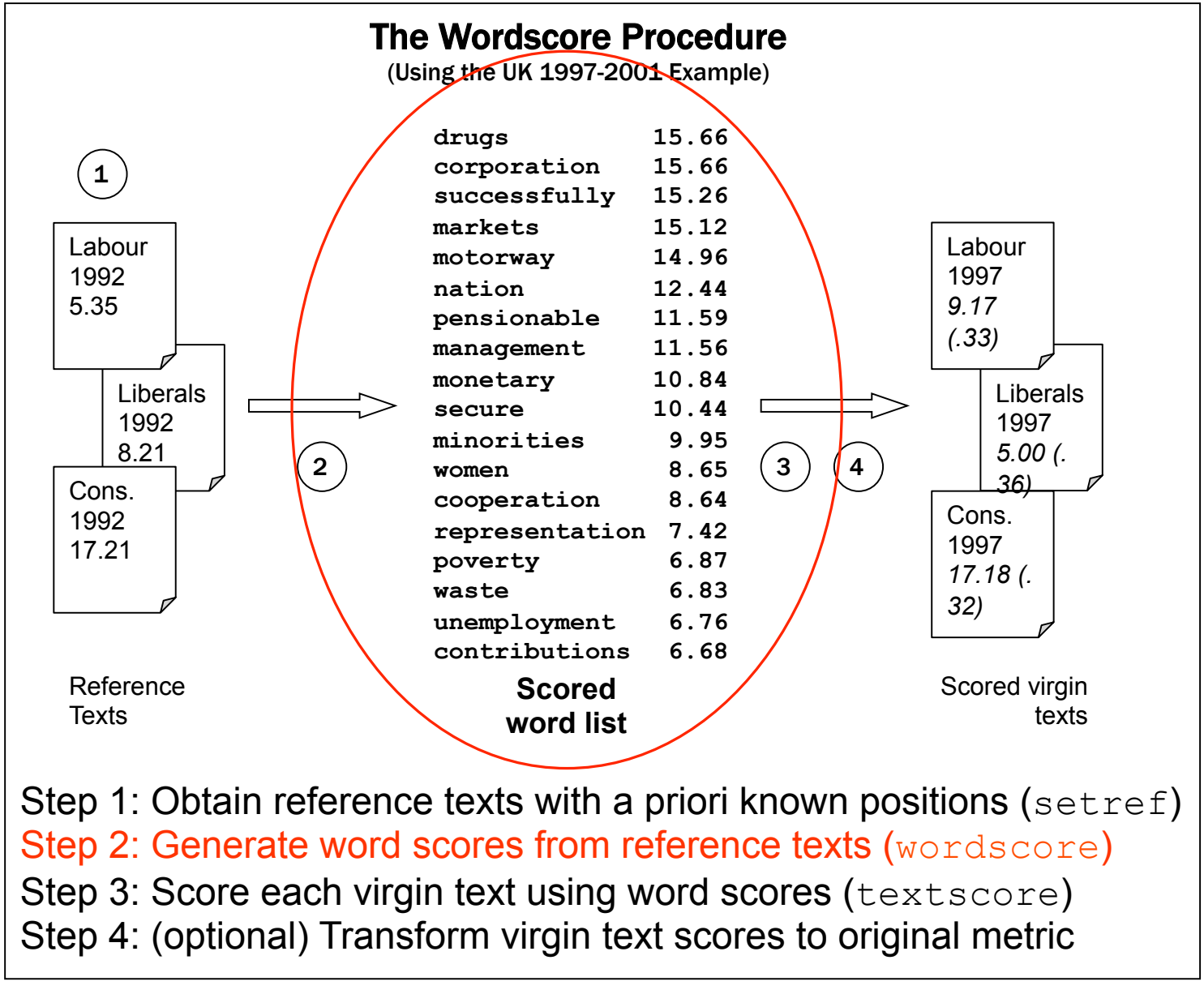
Step 1: Obtain reference texts with a priori known positions (`setref`)

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



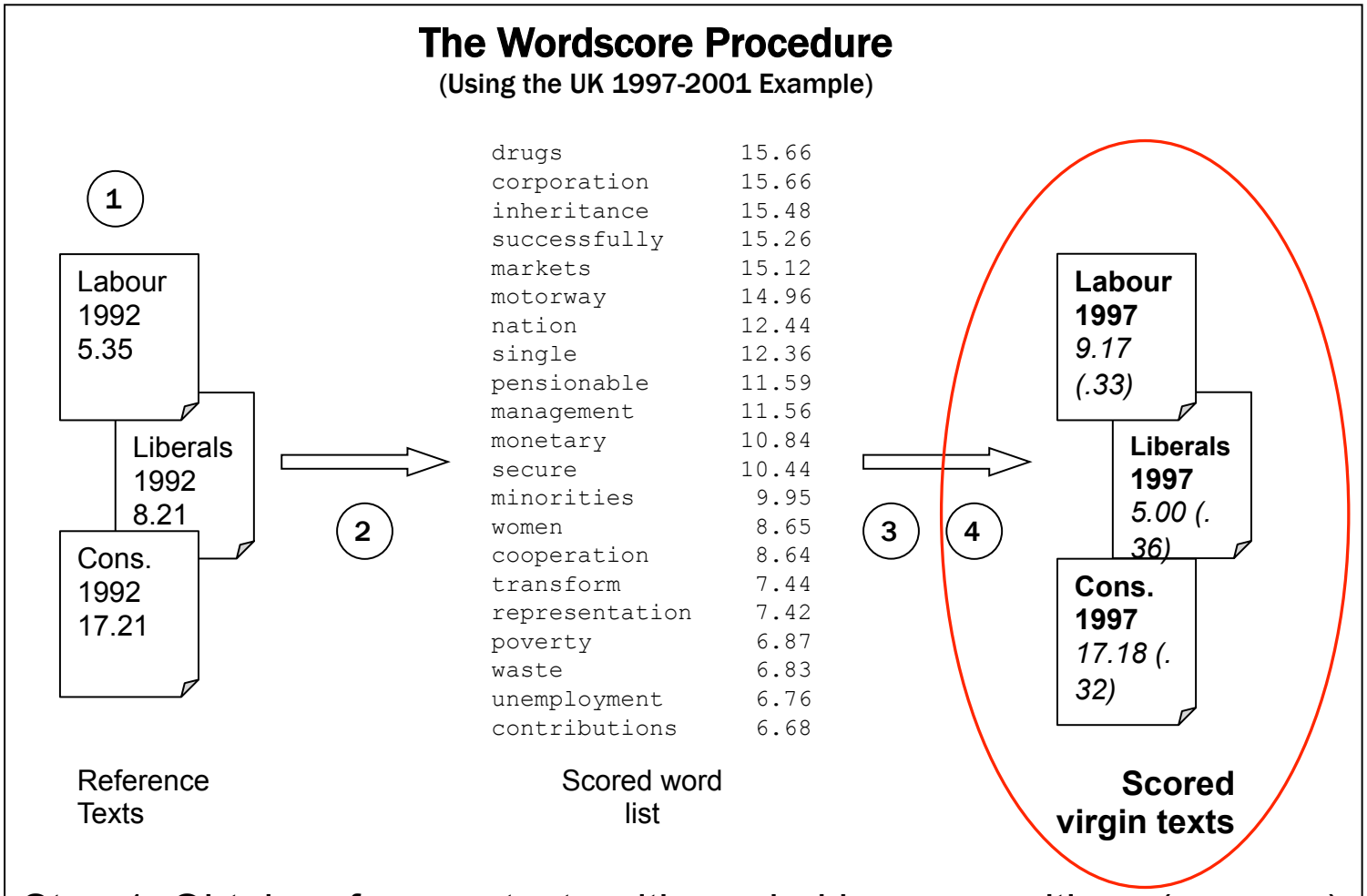
Step 1: Obtain reference texts with a priori known positions (`setref`)

Step 2: Generate word scores from reference texts (`wordscore`)

Step 3: Score each virgin text using word scores (`textscore`)

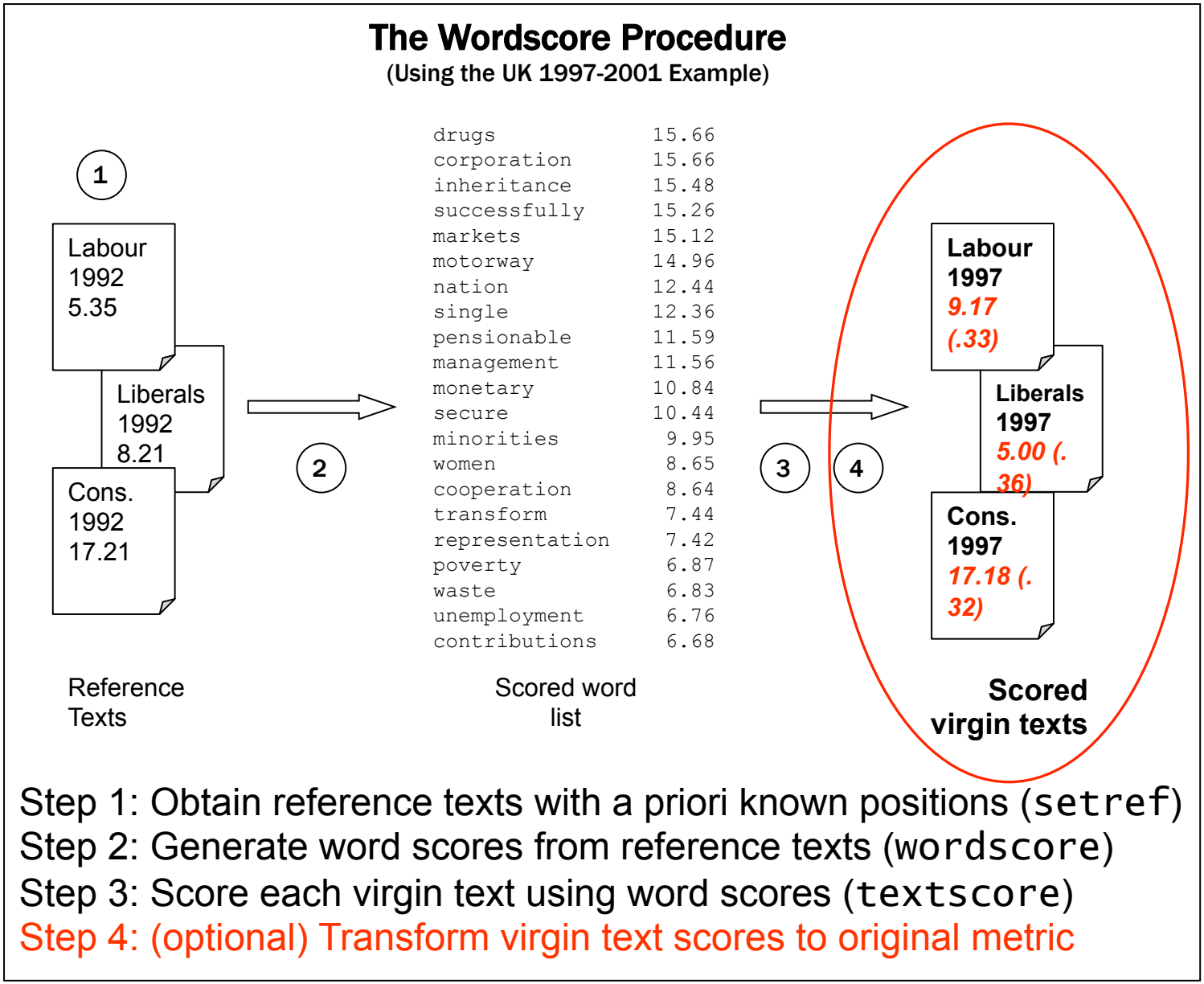
Step 4: (optional) Transform virgin text scores to original metric

Wordscores Procedure



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



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 mathematically: Reference texts

- Start with a set of I *reference* texts, represented by an $I \times J$ document-feature matrix C_{ij}, where i indexes the document and j indexes the J total word types
- Each text will have an associated "score" a_i, which is a single number locating this text on a single dimension of difference
  - This can be on a scale metric, such as 1–20
  - Can use arbitrary endpoints, such as -1, 1
- 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:

$$F_{ij} = \frac{C_{ij}}{C_{i\cdot}} \tag{2}$$

where $C_{i\cdot} = \sum_{j=1}^{J} C_{ij}$

Wordscores mathematically: Word scores

- Compute an $I \times J$ matrix of relative document probabilities P_{ij} for each word in each reference text, as

$$P_{ij} = \frac{F_{ij}}{\sum_{i=1}^{I} F_{ij}} \tag{3}$$

- This tells us the probability that given the observation of a specific word j, that we are reading a text of a certain reference document i

Wordscores mathematically: Word scores (example)

- Assume we have two reference texts, A and B
- The word "choice" is used 10 times per 1,000 words in Text A and 30 times per 1,000 words in Text B
- So $F_{i\ \text{"choice"}} = \{.010, .030\}$
- If we know only that we are reading the word choice in one of the two reference texts, then probability is 0.25 that we are reading Text A, and 0.75 that we are reading Text B

$$P_{A\ \text{"choice"}} = \frac{.010}{(.010 + .030)} = 0.25 \tag{4}$$

$$P_{B\ \text{"choice"}} = \frac{.030}{(.010 + .030)} = 0.75 \tag{5}$$

Wordscores mathematically: Word scores

- 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}:

$$S_j = \sum_{i=1}^{I} a_i P_{ij} \tag{6}$$

- In matrix algebra, $\underset{1\times J}{S} = \underset{1\times I}{a} \cdot \underset{I\times J}{P}$
- This procedure will yield a single "score" for every word that reflects the balance of the scores of the reference documents, weighted by the relative document frequency of its normalized term frequency

Wordscores mathematically: Word scores

- Continuing with our example:
  - We "know" (from independent sources) that Reference Text A has a position of -1.0, and Reference Text B has a position of $+1.0$
  - The score of the word "choice" is then $0.25(-1.0) + 0.75(1.0) = -0.25 + 0.75 = +0.50$

Wordscores mathematically: Scoring "virgin" texts

- Here the objective 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
- So the score v_k of a virgin document k consisting of the j word types is:

$$v_k = \sum_j (F_{kj} \cdot s_j) \tag{7}$$

where $F_{kj} = \frac{C_{kj}}{C_{k.}}$ as in the reference document relative word frequencies

- 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

Wordscores mathematically: Rescaling raw text scores

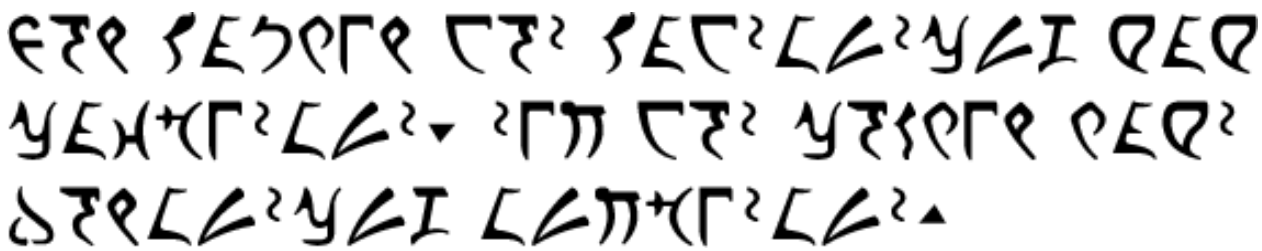
- Because of overlapping or non-discriminating words, the raw text scores will be dragged to the interior of the reference scores (we will see this shortly in the results)
- Some procedures can be applied to rescale them, either to a unit normal metric or to a more "natural" metric
- Martin and Vanberg (2008) have proposed alternatives to the LBG (2003) rescaling

Computing confidence intervals

- The score v_k of any text represents a weighted mean
- LBG (2003) used this logic to develop a standard error of this mean using a *weighted variance* of the scores in the virgin text
- Given some assumptions about the scores being fixed (and the words being conditionally independent), this yields approximately normally distributed errors for each v_k
- An alternative would be to bootstrap the textual data prior to constructing C_{ij} and C_{kj} — see Lowe and Benoit (2012)

Pros and Cons of the Wordscores approach

- Fully automated technique with minimal human intervention or judgment calls – only with regard to reference text selection
- Language-blind: all we need to know are reference scores
- Could potentially work on texts like this:



(See `http://www.kli.org`)

Pros and Cons of the Wordscores approach

- Estimates unknown positions on a priori scales – hence no inductive scaling with a posteriori interpretation of unknown policy space
- Very dependent on correct identification of:
  - appropriate reference texts
  - appropriate reference scores

Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

Suggestions for choosing reference texts

- Texts need to contain information representing a clearly dimensional position
- Dimension must be known a priori. Sources might include:
  - Survey scores or manifesto scores
  - Arbitrarily defined scales (e.g. -1.0 and 1.0)
- Should be as discriminating as possible: extreme texts on the dimension of interest, to provide reference anchors
- Need to be from the same lexical universe as virgin texts
- Should contain lots of words

Suggestions for choosing reference values

- Must be "known" through some trusted external source
- For any pair of reference values, all scores are simply linear rescalings, so might as well use (-1, 1)
- The "middle point" will not be the midpoint, however, since this will depend on the relative word frequency of the reference documents
- Reference texts if scored as virgin texts will have document scores more extreme than other virgin texts
- With three or more reference values, the mid-point is mapped onto a multi-dimensional simplex. The values now matter but only in relative terms (we are still investigating this fully)

Multinomial Bayes model of Class given a Word
Class posterior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the posterior probability of membership in class k for word j
- Under *certain conditions*, this is identical to what LBG (2003) called P_{wr}
- Under those conditions, the LBG "wordscore" is the linear difference between $P(c_k|w_j)$ and $P(c_{\neg k}|w_j)$

"Certain conditions"

- The LBG approach required the identification not only of texts for each training class, but also "reference" scores attached to each training class
- Consider two "reference" scores s_1 and s_2 attached to two classes $k = 1$ and $k = 2$. Taking P_1 as the posterior $P(k = 1|w = j)$ and P_2 as $P(k = 2|w = j)$, A generalised score s_j^* for the word j is then

$$\begin{aligned} s_j^* &= s_1 P_1 + s_2 P_2 \\ &= s_1 P_1 + s_2(1 - P_1) \\ &= s_1 P_1 + s_2 - s_2 P_1 \\ &= P_1(s_1 - s_2) + s_2 \end{aligned}$$

"Certain conditions": More than two reference classes

- For more than two reference classes, if the reference scores are ordered such that $s_1 < s_2 < \cdots < s_K$, then

$$
\begin{aligned}
s_j^* &= s_1 P_1 + s_2 P_2 + \cdots + s_K P_K \\
&= s_1 P_1 + s_2 P_2 + \cdots + s_K (1 - \sum_{k=1}^{K-1} P_k) \\
&= \sum_{k=1}^{K-1} P_i (s_k - s_K) + s_I
\end{aligned}
$$

A simpler formulation:
Use reference scores such that $s_1 = -1.0, s_K = 1.0$

- From above equations, it should be clear that any set of reference scores can be linearly rescaled to endpoints of $-1.0, 1.0$
- This simplifies the "simple word score"

$$s_j^* = (1 - 2P_1) + \sum_{k=2}^{K-1} P_k(s_k - 1)$$

- which simplifies with just two reference classes to:

$$s_j^* = 1 - 2P_1$$

Implications

- LBG's "word scores" come from a linear combination of class posterior probabilities from a Bayesian model of class conditional on words
- We might as well always anchor reference scores at $-1.0, 1.0$
- There is a special role for reference classes in between $-1.0, 1.0$, as they balance between "pure" classes — more in a moment
- There are alternative scaling models, such that used in Beauchamp's (2012) "Bayesscore", which is simply the difference in logged class posteriors at the word level. For $s_1 = -1.0$, $s_2 = 1.0$,

$$\begin{aligned} s_j^B &= -\log P_1 + \log P_2 \\ &= \log \frac{1 - P_1}{P_1} \end{aligned}$$

Moving to the document level

- The "Naive" Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a "test" document, to produce:

$$P(c|d) = P(c) \frac{\prod_j P(w_j|c)}{P(w_j)}$$

- So we *could* consider a document-level relative score, e.g. $1 - 2P(c_1|d)$ (for a two-class problem)
- But this turns out to be *useless*, since the predictions of class are highly separated

Moving to the document level

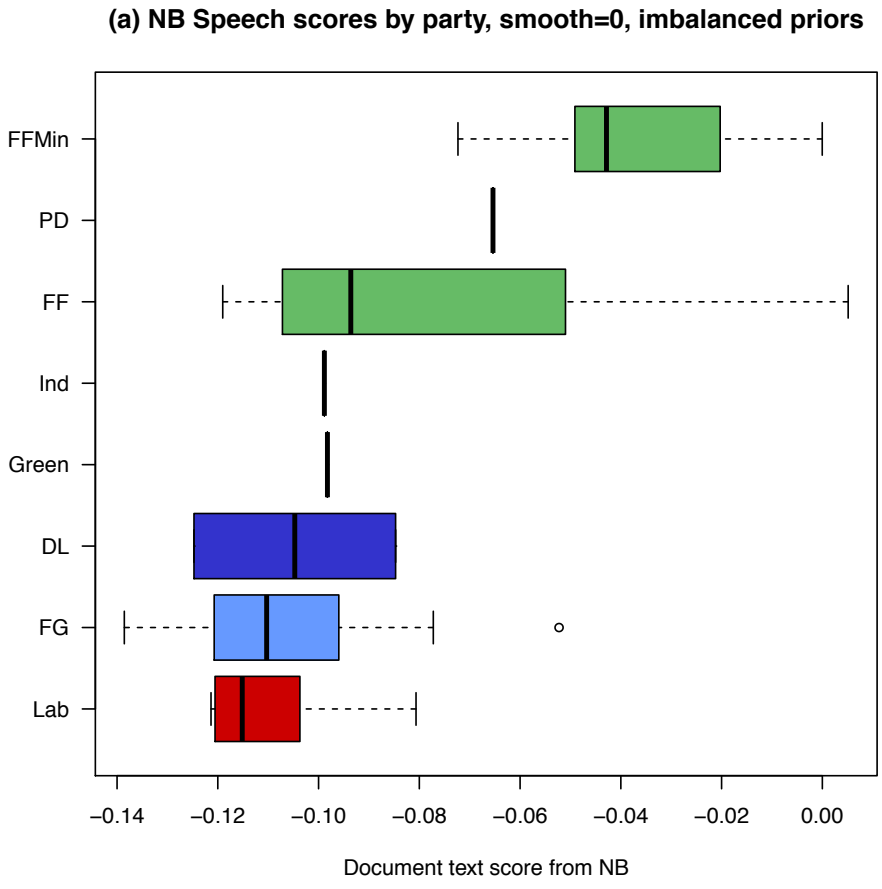
- A better solution is to score a test document as the arithmetic mean of the scores of its words
- This is exactly the solution proposed by LBG (2003)
- Beauchamp (2012) proposes a "Bayesscore" which is the arithmetic mean of the log difference word scores in a document – which yields extremely similar results

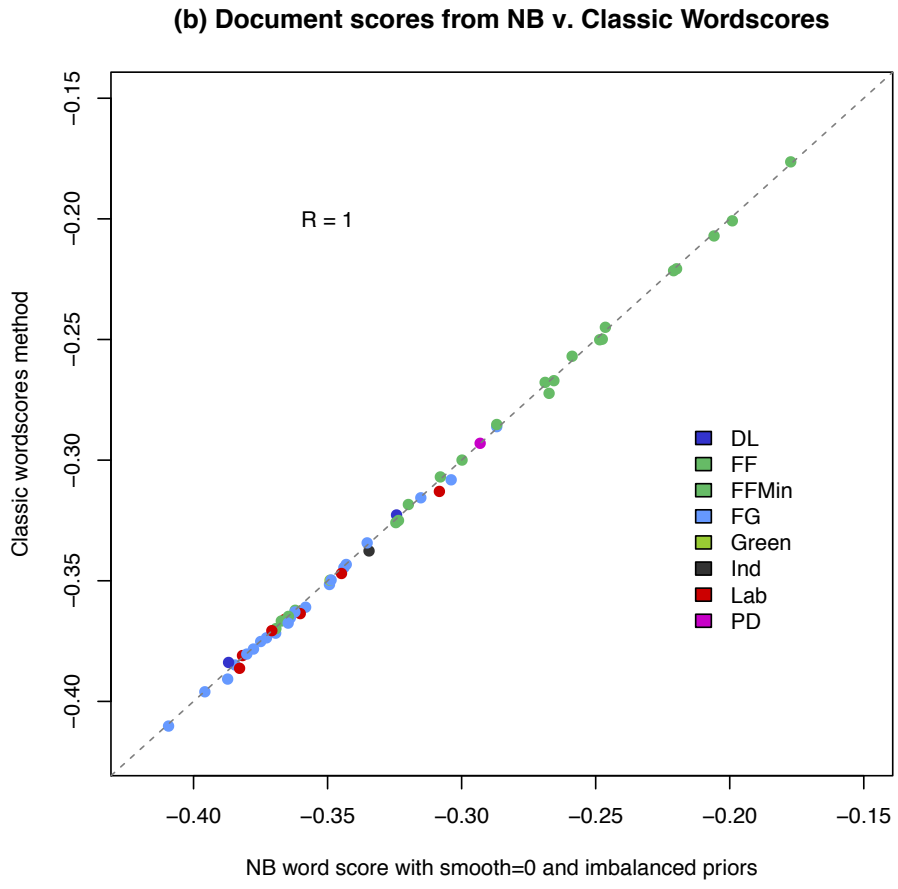
And now for some demonstrations with data...

Outline

- Types of classifiers:
  - Naive Bayes
  - Regularized regression
- From classification to scaling
  - Basics of supervised scaling methods
  - Wordscores
  - Practical aspects
  - Examples

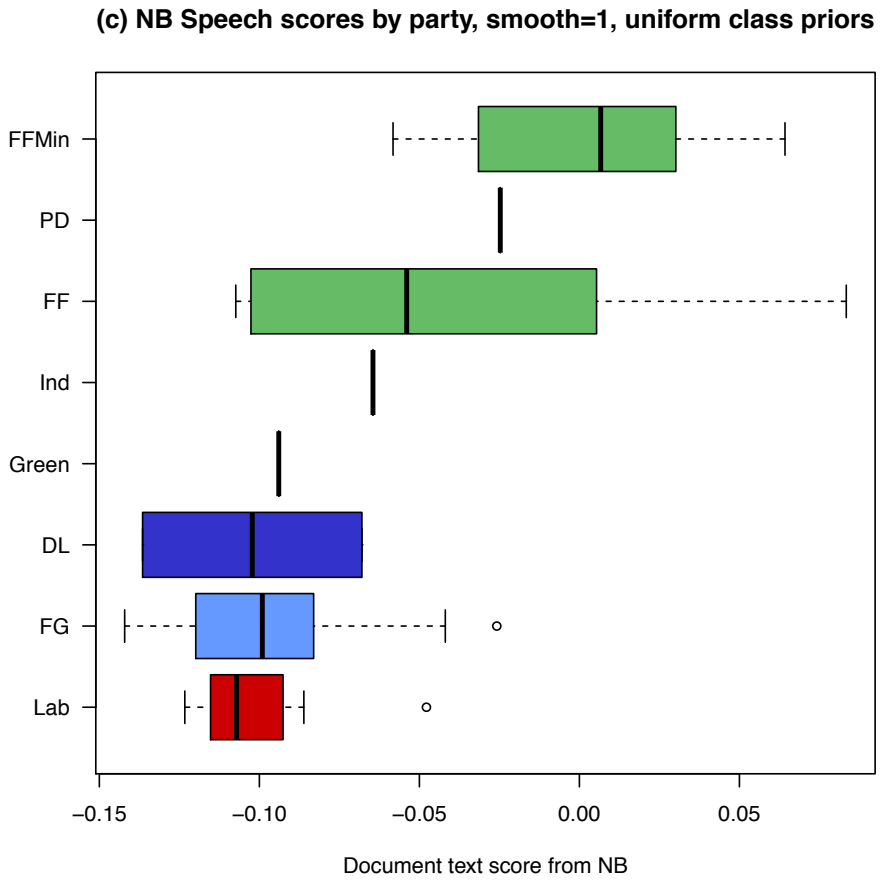
Application 1: Dail speeches from LBG (2003)

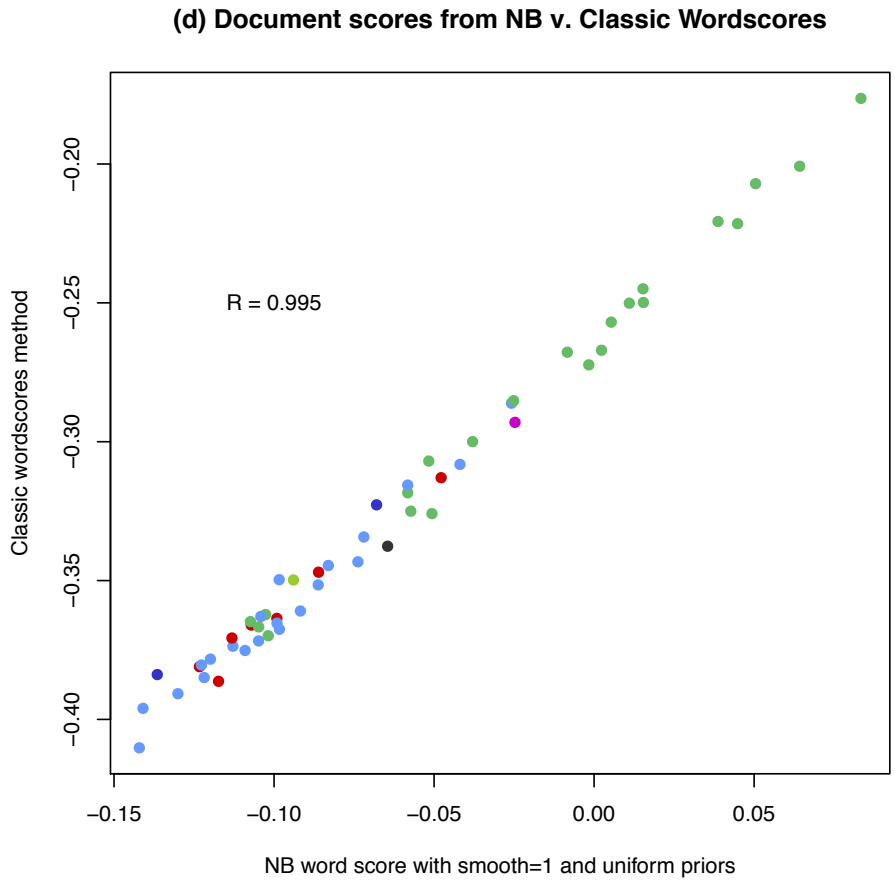




- three reference classes (Opposition, Opposition, Government) at {-1, -1, 1}
- no smoothing

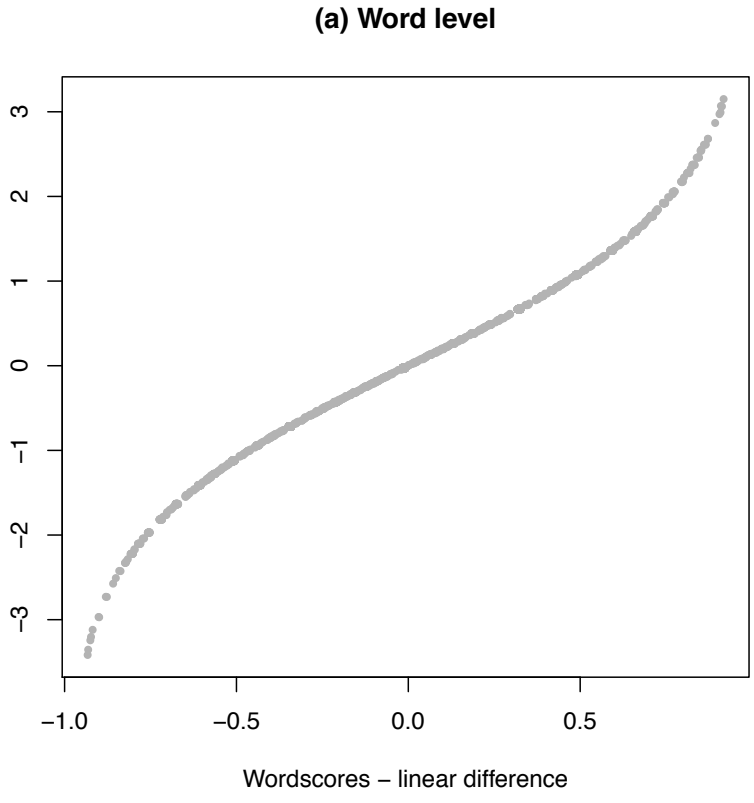
Application 1: Dail speeches from LBG (2003)

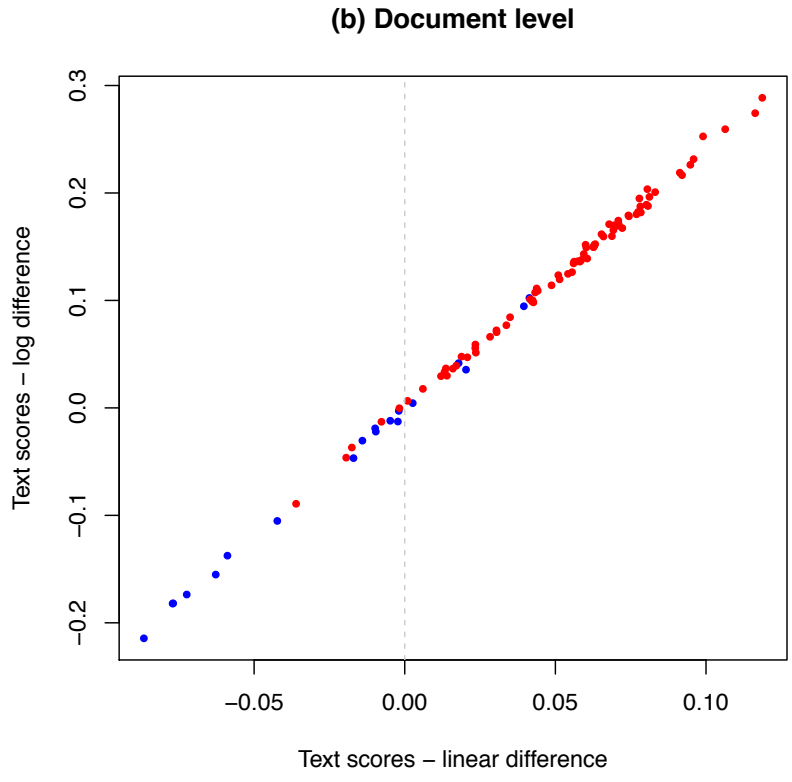




- two reference classes (Opposition+Opposition, Government) at {-1, 1}
- Laplace smoothing

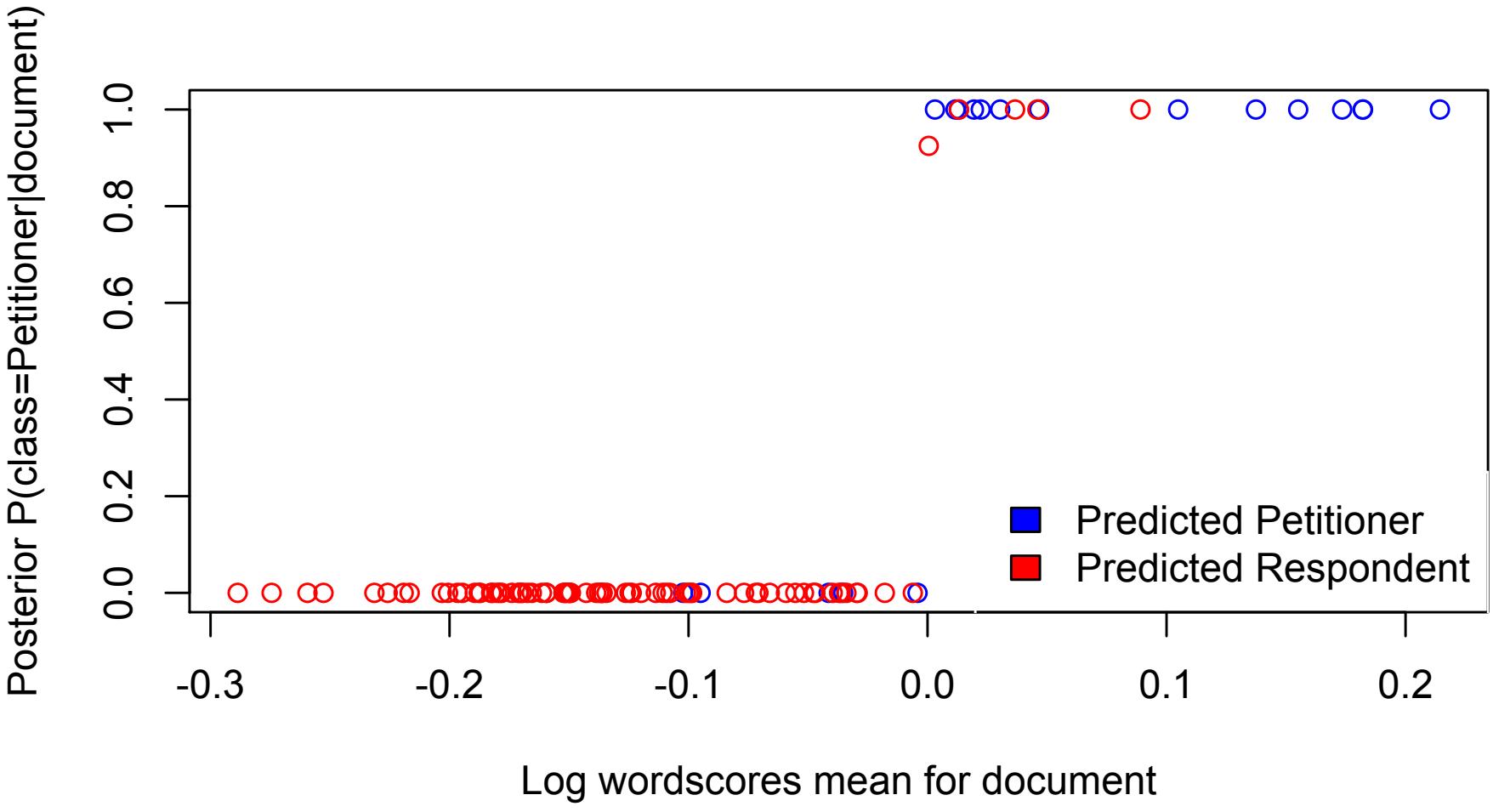
Application 2: Classifying legal briefs (Evans et al 2007) Wordscores v. Bayesscore



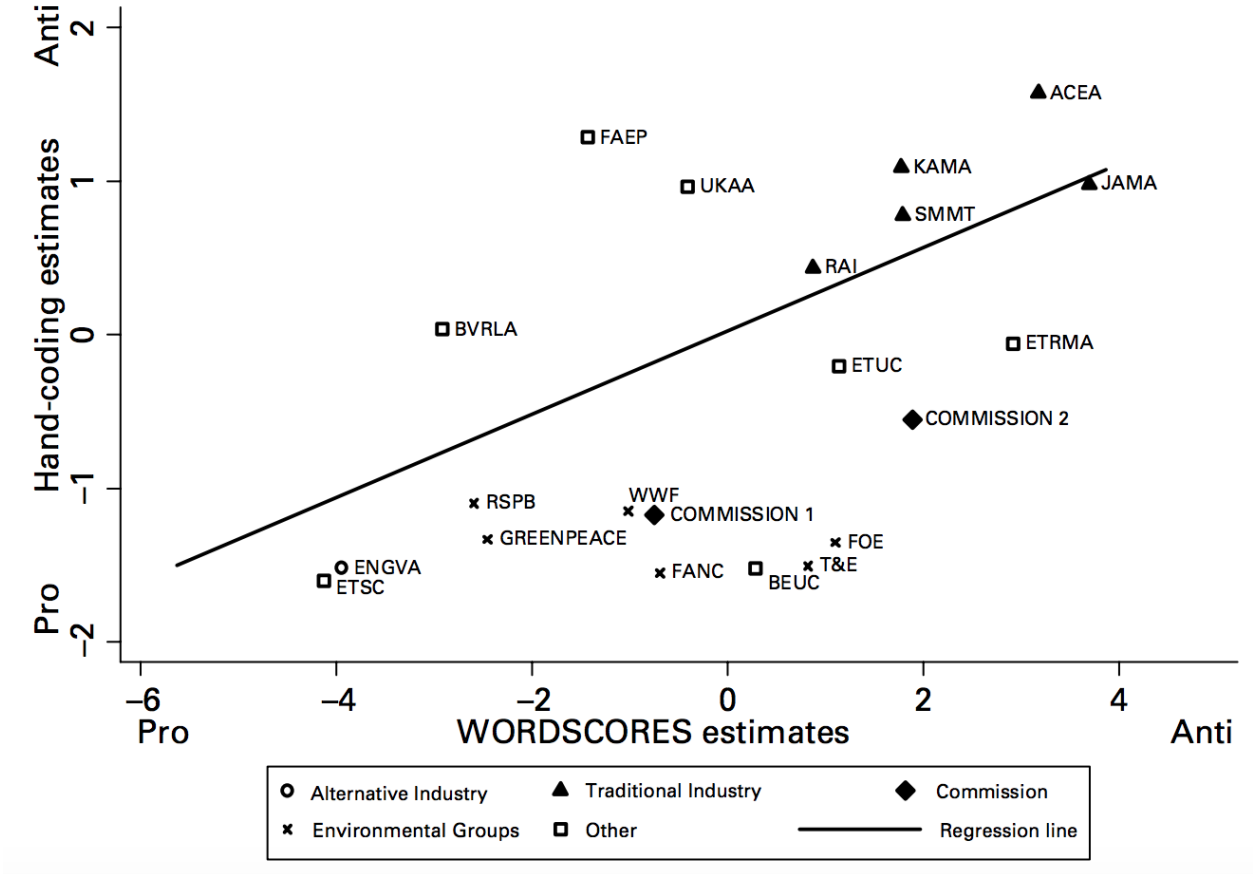


- Training set: Petitioner and Respondent litigant briefs from *Grutter/Gratz v. Bollinger* (a U.S. Supreme Court case)
- Test set: 98 amicus curiae briefs (whose P or R class is known)

Application 2: Classifying legal briefs (Evans et al 2007)
Posterior class prediction from NB versus log wordscores

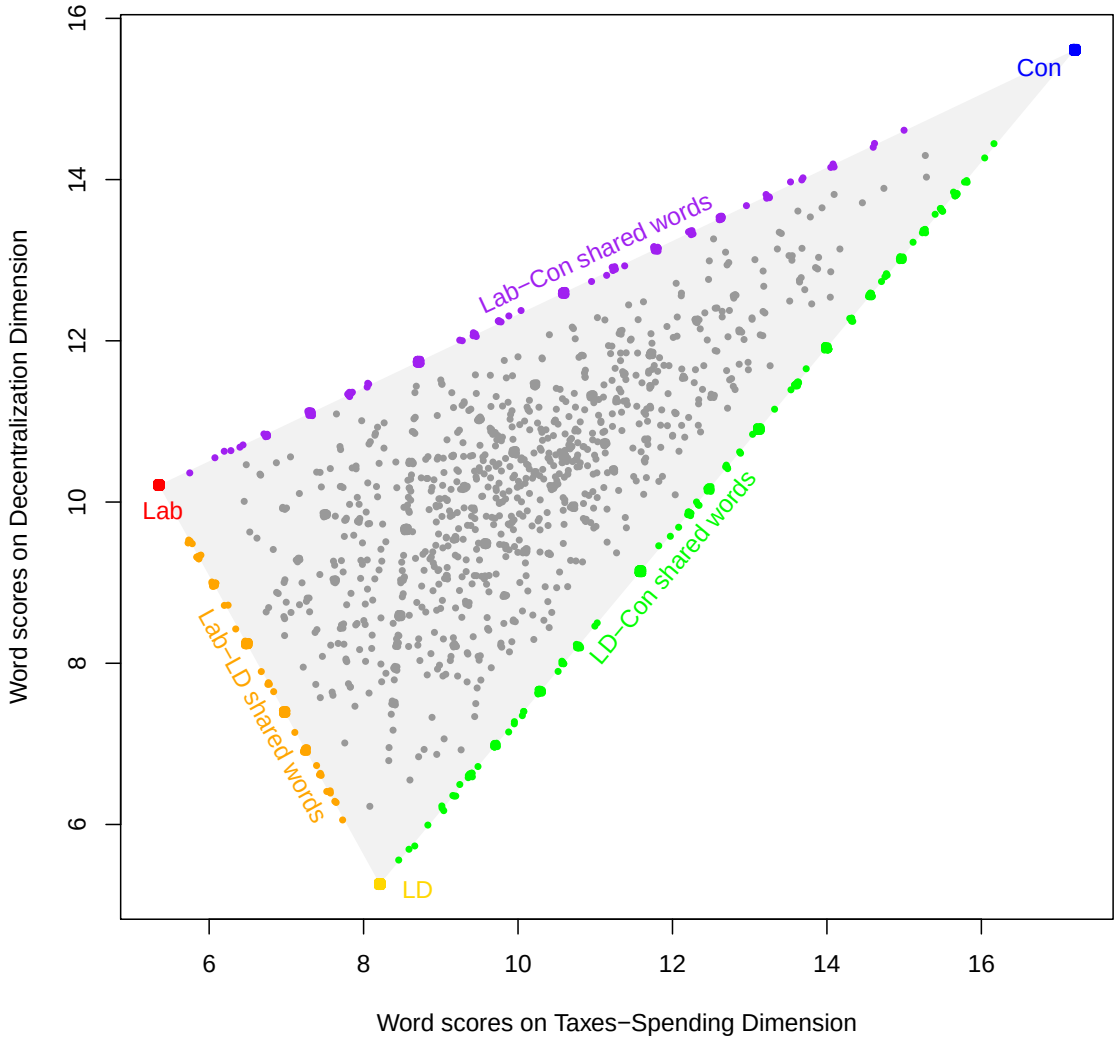


Application 3: Scaling environmental interest groups (Klüver 2009)



- Dataset: text of online consultation on EU environmental regulations
- Reference texts: most extreme pro- and anti-regulation groups

Application 4: LBG's British manifestos
More than two reference classes



- x-axis: Reference scores of $\{5.35, 8.21, 17.21\}$ for Lab, LD, Conservatives
- y-axis: Reference scores of $\{10.21, 5.26, 15.61\}$