EC999: Document Scaling

Thiemo Fetzer

University of Chicago & University of Warwick

December 6, 2016

Plan

Scaling Approaches

Wordscores approach

Bayesscore

Classification versus Scaling

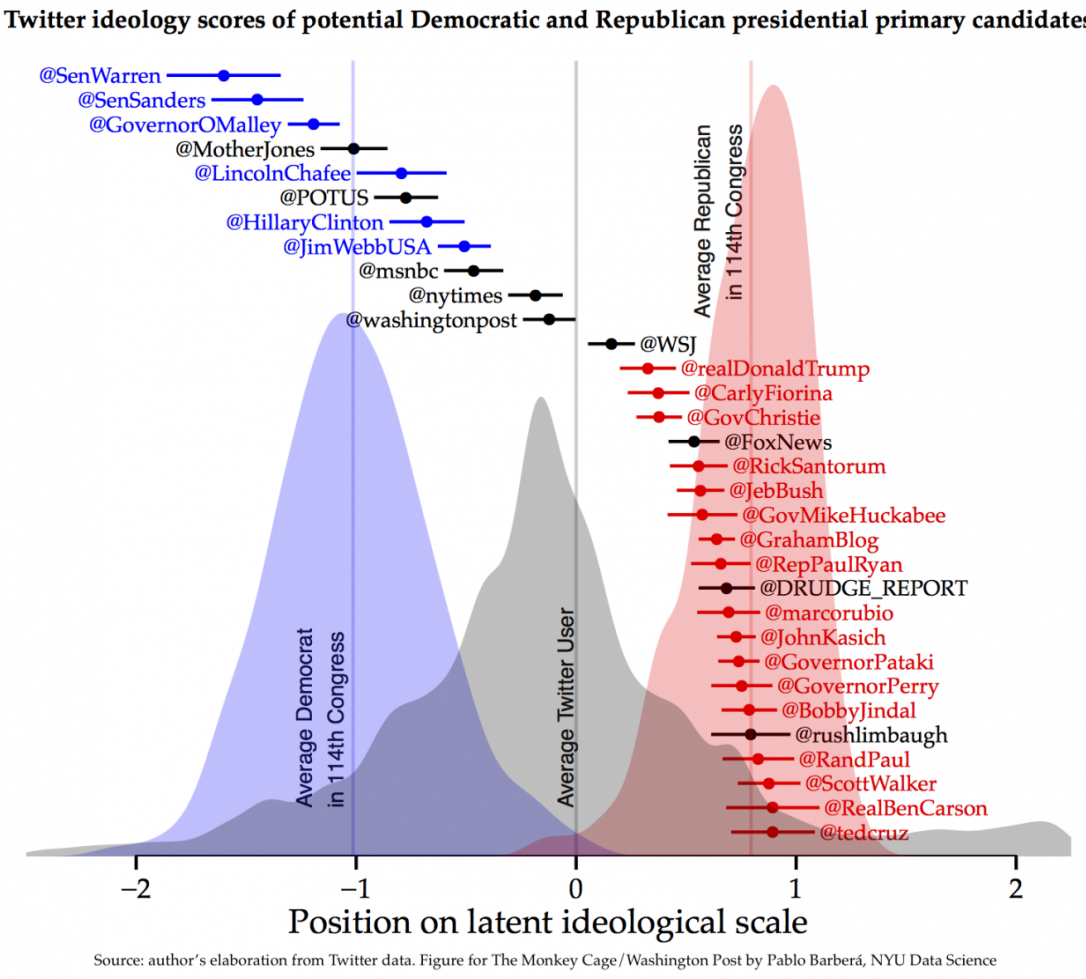
- We introduce the idea of scaling or scoring texts in relation to a reference text for which we know what position that text takes.
- We will embed this discussion within the framework presented here relating the document scaling approach to the conceptual framework presented here to think about more general classification problems.

Learning about latent variables

Generation and Inference

Underlying latent (unobserved) characteristic θ generates the distribution of word counts. Scaling methods attempt to reverse this process: using the word counts to *infer* the value of θ.

Learning about latent variables



Idealogical scaling, not based on text, but on Twitter follower network structure.

Document Scaling

A most common task for language processing is to compute a similarity or an overlap measure between a document and a query.

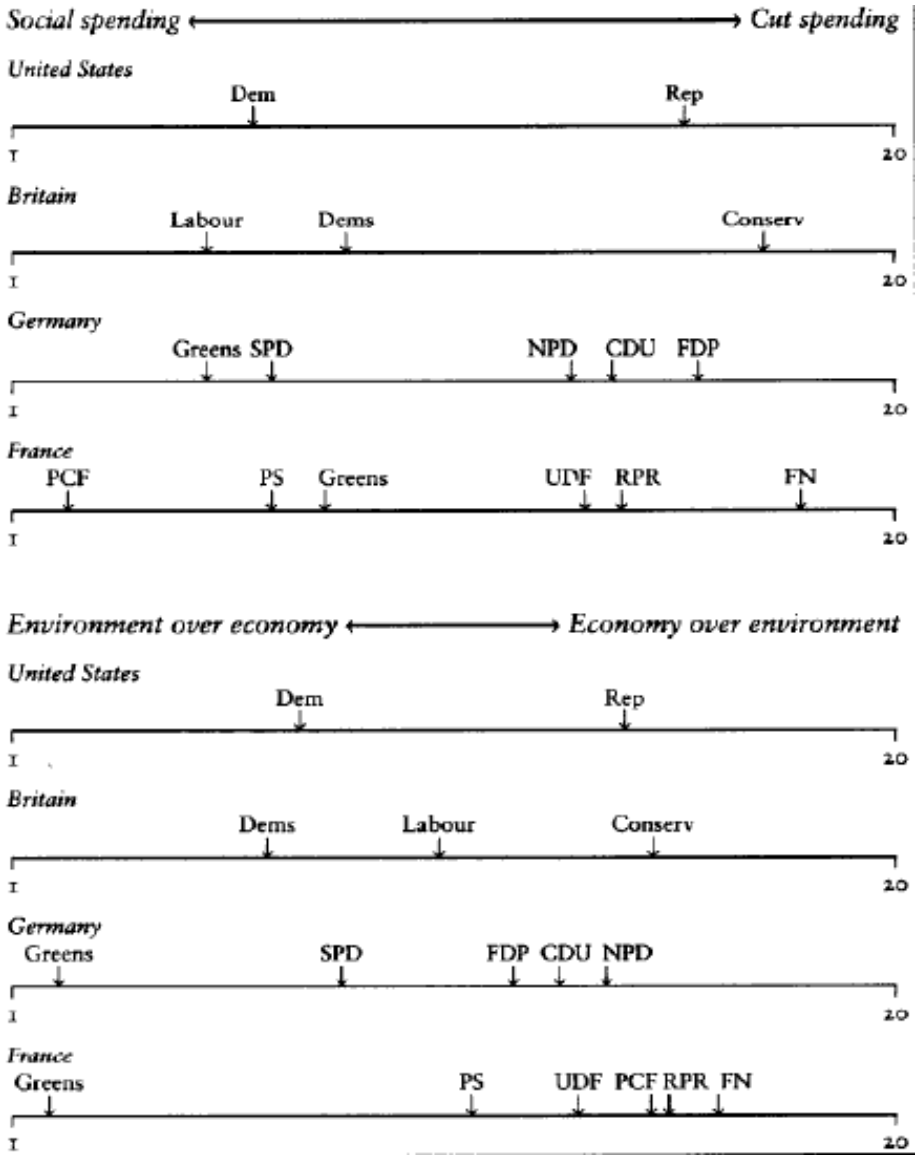
In applications, you may want to score a document relative to a reference text.

This is known as wordscoring approach, where you have two sets of texts:

1. Reference texts: texts which we understand well as representing some latent positions.
2. Virgin texts: texts for which you do not know what their latent position is.

Word scoring approaches are a form of querying where you define a vocabulary based on the training set of the reference texts and you use these to score the virgin texts.

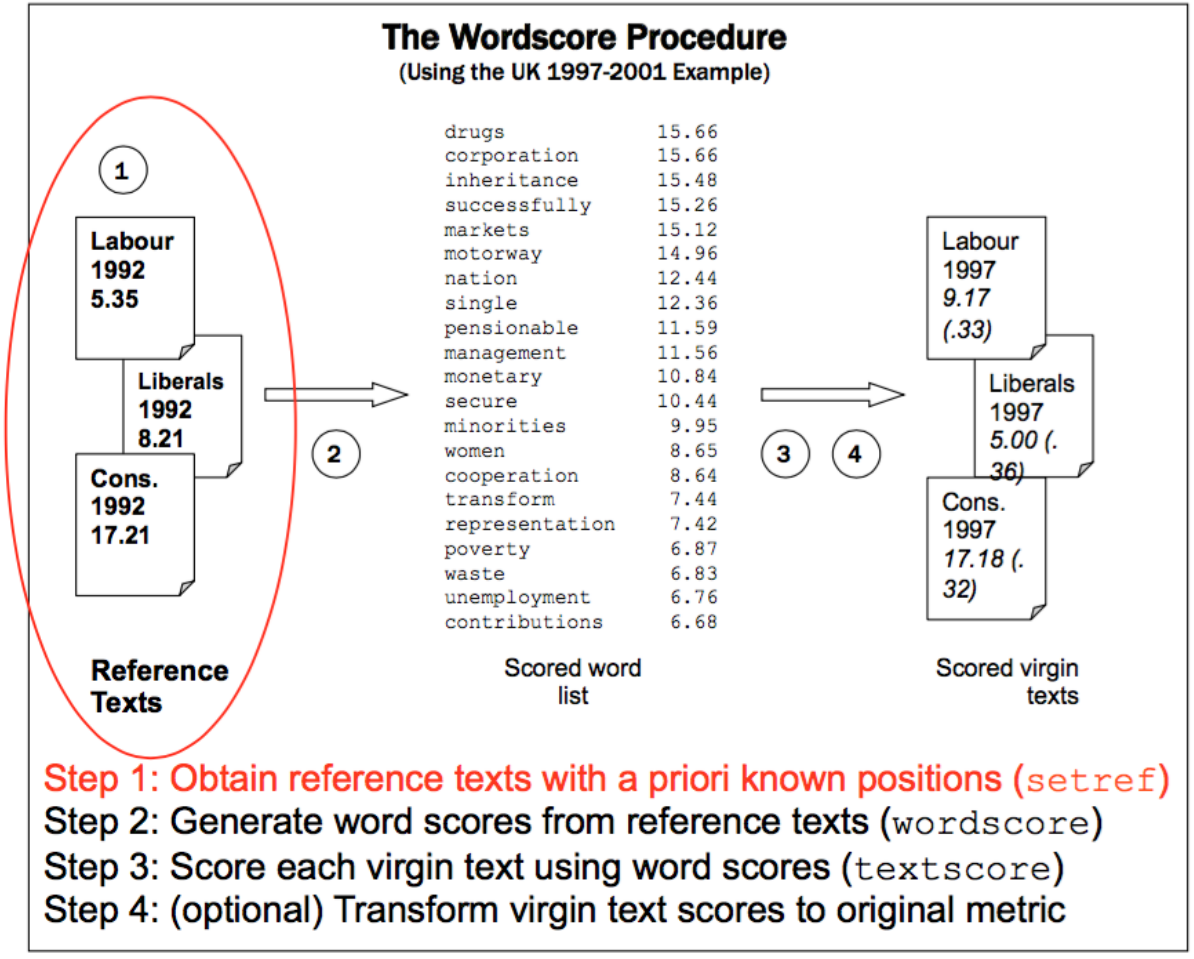
Measuring relative party positions along salient dimensions



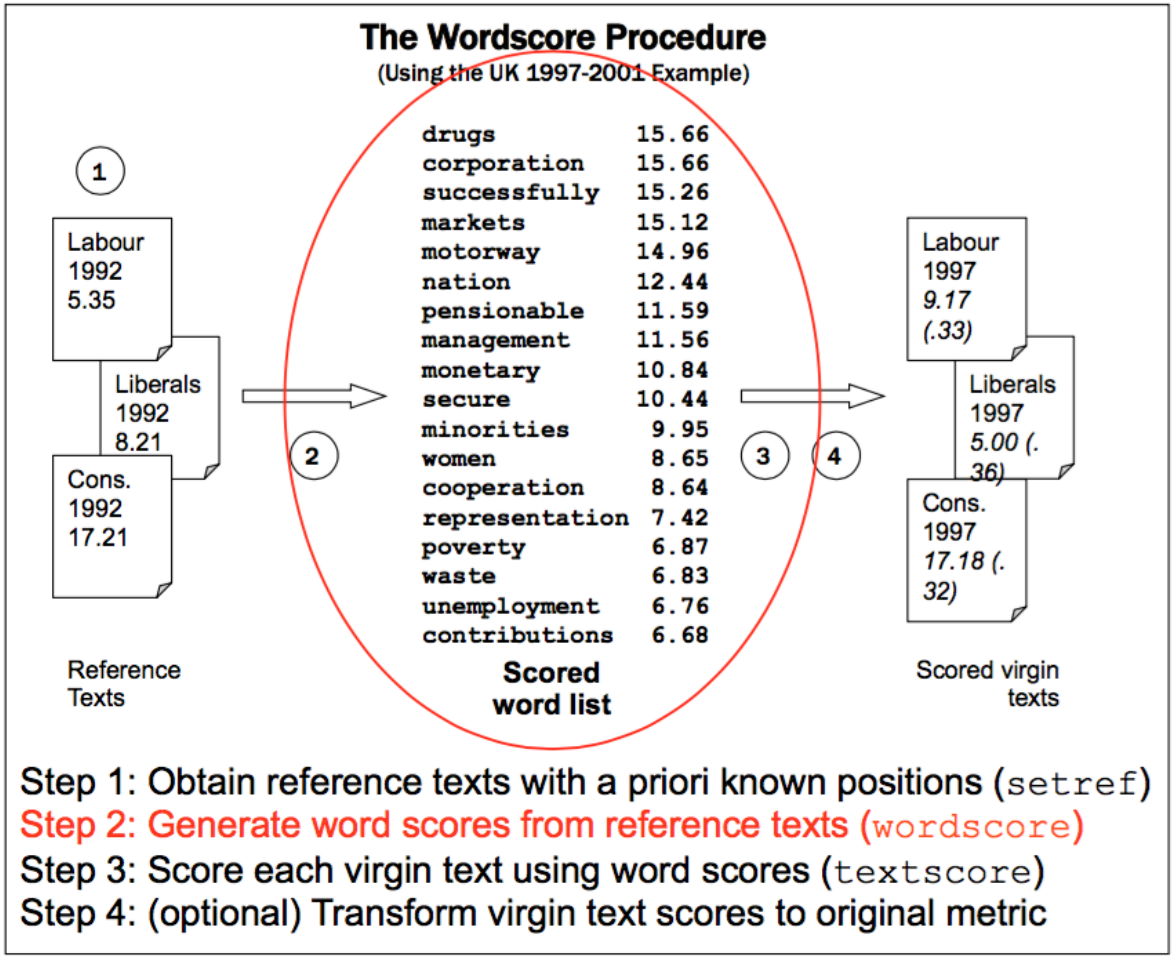
Alignment of political parties along issues from Dalton (1996)

Plan

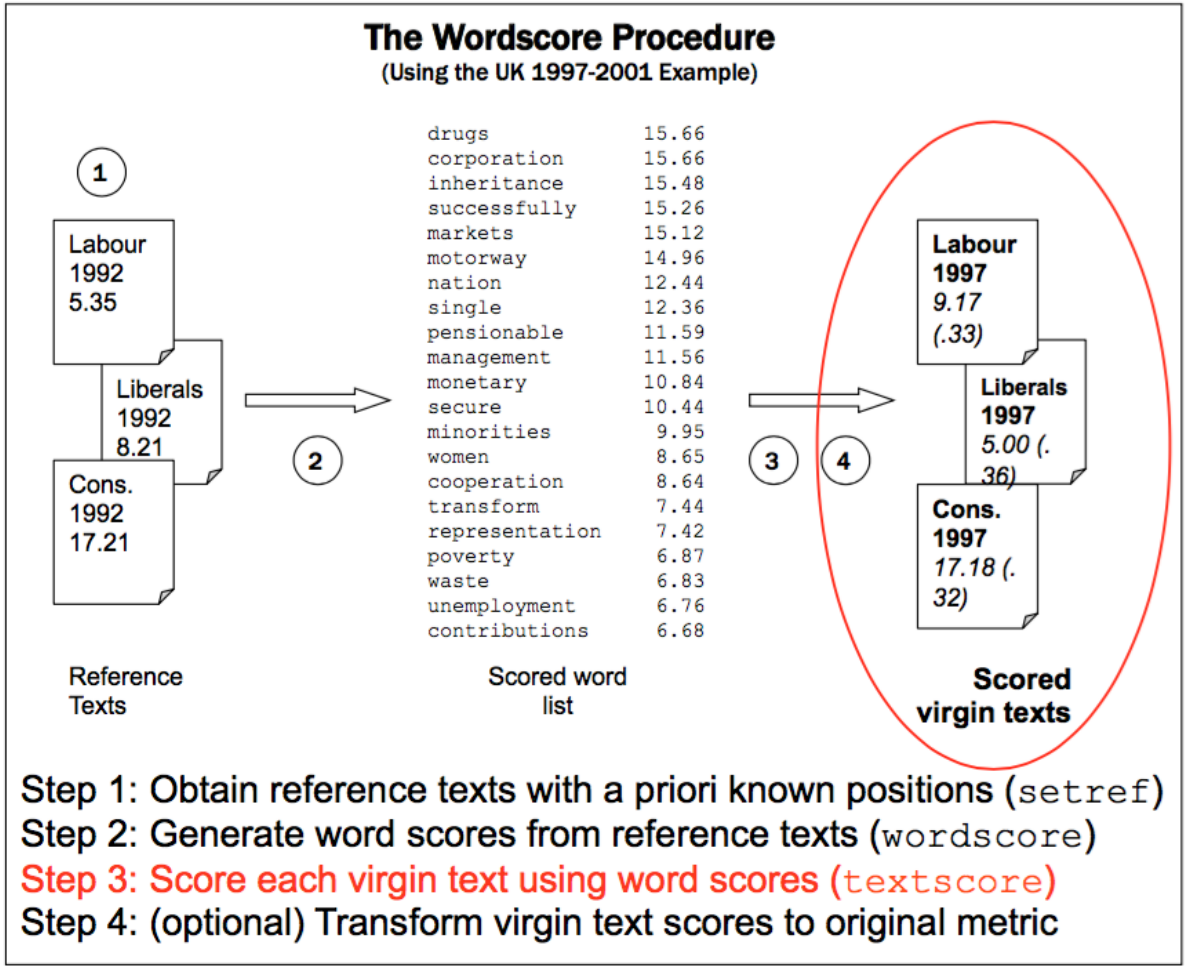
Wordscores as Heuristic Document Scaling Approach



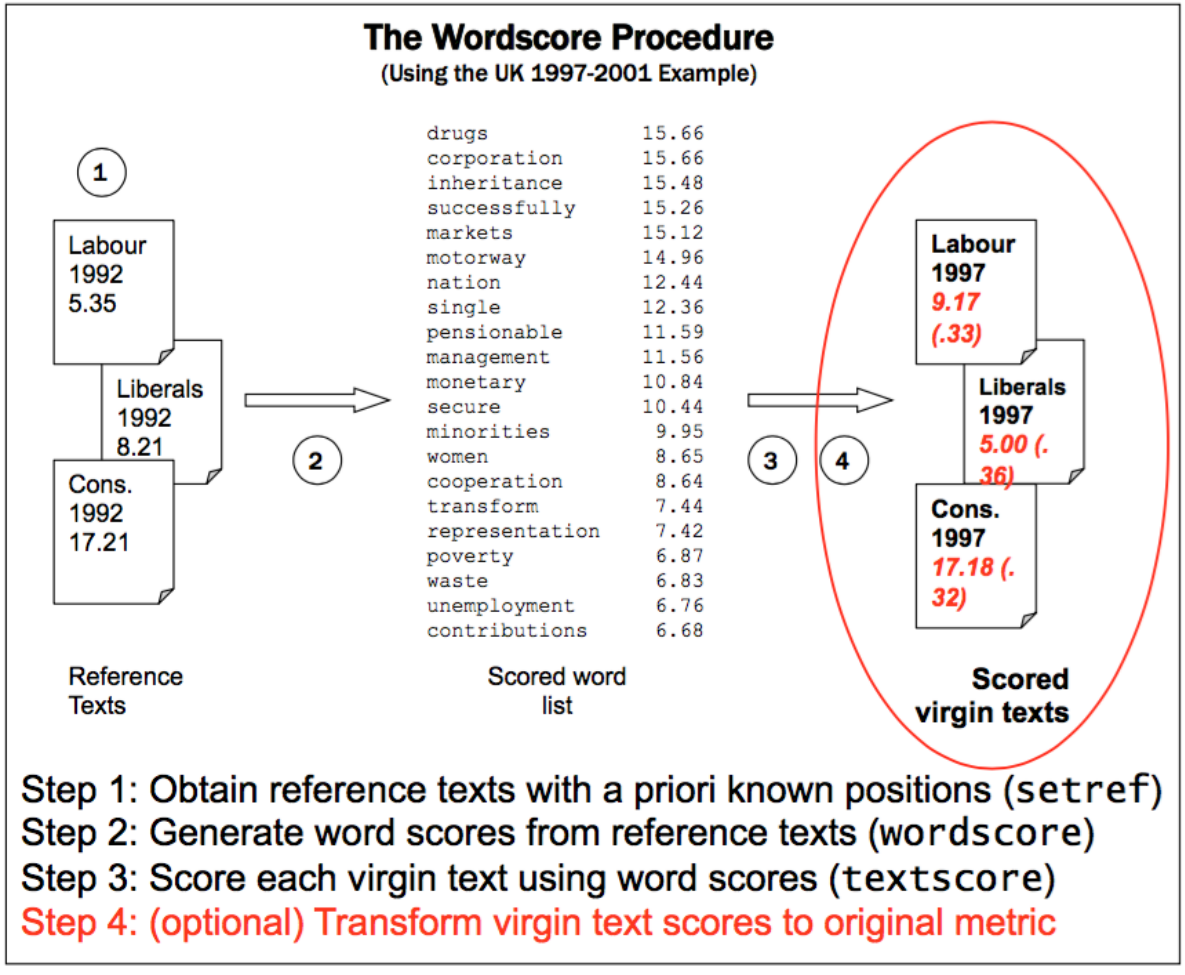
Wordscores as Heuristic Document Scaling Approach



Wordscores as Heuristic Document Scaling Approach



Wordscores as Heuristic Document Scaling Approach



Wordscores: Reference Text Normalization

- Start with a set of J reference texts, represented by an $I \times J$ document-term frequency matrix C_{ij} , where i indexes the word types and j indexes the document.
- Each reference text will have an associated "score" a_i , which is a single number locating this text on a single dimension of difference (this could be a scale metric, such as 120 or use arbitrary endpoints, such as -1, 1)
- We normalize the document-term frequency matrix within each document by converting C_{ij} into a relative document-term frequency matrix (within document), by dividing C_{ij} by its word total marginals:

$$F_{ij} = \frac{C_{ij}}{C_j}$$

where $C_j = \sum_{i=1}^{I} C_{ij}$ is the total number of words in reference document j.

Wordscores: Computing the Wordscores

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

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

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

Scoring a virgin document

- 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
- The score v_k of a virgin document k consisting of the j word types is:

$$v_k = \sum_j F_{kj} s_j$$

where as in the reference document $F_{kj} = \frac{C_{kj}}{C_k}$ is the relative word frequency.

- 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)

Two Reference Text Example

- Assume we have two reference texts, D and R, Reference Text D has a position of 1.0, and Reference Text R has a position of +1.0.
- For example:

	test	C_{iD}	C_{iR}
w_i	climate	30	10
	marriage	14	40
	Wall Street	35	18
	energy	10	65
	wealth	31	20
	ISIS	15	5
	Social Security	25	54
	Medicare	22	43
	economy	15	18
	Washington	5	20
...			
	C_j	1000	1500

$\Rightarrow$ Scale the C_{ij} counts by the column sum to normalize for document size.

Two Reference Text Example: Normalizing by document size

This measures the share of document j's words that are of word type i.

		F_{iD}	F_{iR}
w_i	climate	0.030	0.007
	marriage	0.014	0.027
	Wall Street	0.035	0.012
	energy	0.010	0.043
	wealth	0.031	0.013
	ISIS	0.015	0.003
	Social Security	0.025	0.036
	Medicare	0.022	0.029
	economy	0.015	0.012
	Washington	0.005	0.013

For reference document D, 3% of the total number of words are "climate".

$\Rightarrow$ Next step, compute word scores P_{ij}.

Two Reference Text Example: Compute P_{ij}

		P_{iD}	P_{iR}
w_i	climate	0.818	0.182
	marriage	0.344	0.656
	Wall Street	0.745	0.255
	energy	0.188	0.813
	wealth	0.699	0.301
	ISIS	0.818	0.182
	Social Security	0.410	0.590
	Medicare	0.434	0.566
	economy	0.556	0.444
	Washington	0.273	0.727

$\Rightarrow$ conditional on word climate being observed, the chance is more than four times higher that the document is of type D.
$\Rightarrow$ Divide the relative frequency F_{ij} by the row-sum to obtain a measure of likelihood that word w_j is appearing in document i. In the two reference text case, one is the complement of the other.

Scoring a new "virgin" document

Let the reference scores be $A_R = +1$ and $A_D = -1$ (or any other arbitrary value). A word's score is now

$$S_i = A_R P_{iR} + A_D P_{iD}$$

		P_{jD}	P_{jR}	S_i
w_i	climate	0.818	0.182	-0.636
	marriage	0.344	0.656	0.311
	Wall Street	0.745	0.255	-0.489
	energy	0.188	0.813	0.625
	wealth	0.699	0.301	-0.398
	ISIS	0.818	0.182	-0.636
	Social Security	0.410	0.590	0.180
	Medicare	0.434	0.566	0.132
	economy	0.556	0.444	-0.111
	Washington	0.273	0.727	0.455
	Score	-1	1	

Scoring a new "virgin" document

A "virgin" document is scored as

$$S_V^{wordscore} = \sum_{i=1}^{I} \frac{C_{iV}}{C_V} S_i$$

	Virgin document	Virgin document normalized	S_i
climate	14	0.034	-0.636
marriage	2	0.005	0.311
Wall Street	4	0.010	-0.489
energy	4	0.010	0.625
wealth	4	0.010	-0.398
ISIS	6	0.015	-0.636
Social Security	3	0.007	0.180
Medicare	0	0.000	0.132
economy	2	0.005	-0.111
Washington	0	0.000	0.455
			-0.031
Document length	412		

Relating Scaling to Classification Terminology

- Wordscores algorithm is not explicitly derived from any statistical model of word generation.
- many aspects of the method can support such interpretations.
- Methods for assigning scores to words and documents have a symmetric probabilistic interpretation
- It turns out (and we will confirm) that there are some Bayesian elements to the algorithm, which is why it compares reasonably well.

Some Issues with Wordscores

- Note that new words outside of the set J may appear in the "virgin documents" - these are simply ignored (because we have no information on their 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)
- The score S_V 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
- An alternative would be to bootstrap the textual data prior to constructing C_{ij} and C_{kj} see Lowe and Benoit (2012)

Scaling illustration: Trumpi- versus Hillariness

- We obtain a corpus of Donald Trump's (+1) campaign speeches and a corpus of Hillary Clinton's (-1) speeches and make these our reference texts.
- We want to then look at speeches given in congress (or any other text) to see to what extent they score on this dimension score.
- This will illustrate the usefulness of word scaling of texts and some of its draw backs.

Scaling illustration: Trumpi- versus Hillariness

```
REFERENCE <- corpus(textfile("../Data/speeches/*.txt", docvarsfrom = "filenames"))
REFERENCE.dfm <- dfm(REFERENCE, toLower = TRUE, removeNumbers = TRUE, removePunct = TRUE, removeSeparators = T
    stem = TRUE, ignoredFeatures = stopwords("english"))
## first 7 elements are clinton speeches, next 6 are Trump
refscores <- c(rep(-1, 7), rep(1, 6))
ws <- textmodel(REFERENCE.dfm, refscores, model = "wordscores", smooth = 1)
## most trumpy words
ws@Sw[ws@Sw > 0.5][1:15]
```

```
##      go  realli    deal   bring  number  border   total hillari     bad  happen  nobodi
##   0.588   0.517   0.649   0.564   0.543   0.512   0.575   0.657   0.514   0.545   0.570
##   china  mexico    folk   islam
##   0.634   0.521   0.533   0.597
```

```
## most hillary words
ws@Sw[ws@Sw < -0.5][1:15]
```

```
##   colleg  student    futur     walk     face opportun   effort stronger   famili     isra
##   -0.646   -0.619   -0.643   -0.501   -0.579   -0.558   -0.623   -0.509   -0.614   -0.580
##   togeth    organ      men    women    still
##   -0.535   -0.520   -0.601   -0.714   -0.513
```

Scaling illustration: Trumpi- versus Hillariness

```
## scoring the reference documents
predict(ws)
```

```
## Predicted textmodel of type: wordscores
##
##                              textscore LBG se   ci lo   ci hi
## clinton-aipac.txt              -0.1150 0.0026 -0.1201 -0.1100
## clinton-economy.txt            -0.1216 0.0027 -0.1269 -0.1164
## clinton-launch.txt             -0.1238 0.0027 -0.1291 -0.1186
## clinton-nominee.txt            -0.1119 0.0029 -0.1175 -0.1062
## clinton-orlando.txt            -0.1116 0.0027 -0.1168 -0.1063
## clinton-supertuesday.txt       -0.1142 0.0030 -0.1201 -0.1083
## clinton-women.txt              -0.1295 0.0030 -0.1353 -0.1237
## trump-aipac.txt                -0.0524 0.0030 -0.0584 -0.0464
## trump-foreignpolicy.txt        -0.0285 0.0030 -0.0343 -0.0226
## trump-launch.txt               -0.0019 0.0029 -0.0076  0.0038
## trump-orlando.txt              -0.0361 0.0029 -0.0418 -0.0303
## trump-supertuesday.txt          0.0081 0.0031  0.0020  0.0143
## trump-victory-newjersey.txt    -0.0617 0.0032 -0.0679 -0.0555
```

Rescaling

- The LBG method was developed to study the evolution of party positions over time, relative to a baseline party position.
- For virgin text, not all words overlap and most likely non discriminating words overlap with the words for which we have word-scores.
- This will drag the distribution of the average scores to the interior, while still preserving the ranking.
- We need to rescale the scores to allow a sustantive interpretation of the scores relative to the reference text.
- 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

Scaling illustration: Trumpi- versus Hillariness

```
## rescale the scores
predict(ws, rescaling = "lbg")
```

```
## Predicted textmodel of type: wordscores
##
##                             textscore LBG se   ci lo   ci hi LBG rescaled LBG lo LBG hi
## clinton-aipac.txt             -0.1150 0.0026 -0.1201 -0.1100       -0.869 -0.975 -0.763
## clinton-economy.txt           -0.1216 0.0027 -0.1269 -0.1164       -1.005 -1.114 -0.896
## clinton-launch.txt            -0.1238 0.0027 -0.1291 -0.1186       -1.051 -1.160 -0.942
## clinton-nominee.txt           -0.1119 0.0029 -0.1175 -0.1062       -0.803 -0.921 -0.685
## clinton-orlando.txt           -0.1116 0.0027 -0.1168 -0.1063       -0.797 -0.906 -0.687
## clinton-supertuesday.txt      -0.1142 0.0030 -0.1201 -0.1083       -0.852 -0.974 -0.729
## clinton-women.txt             -0.1295 0.0030 -0.1353 -0.1237       -1.169 -1.290 -1.049
## trump-aipac.txt               -0.0524 0.0030 -0.0584 -0.0464        0.433  0.309  0.557
## trump-foreignpolicy.txt       -0.0285 0.0030 -0.0343 -0.0226        0.930  0.807  1.052
## trump-launch.txt              -0.0019 0.0029 -0.0076  0.0038        1.482  1.363  1.601
## trump-orlando.txt             -0.0361 0.0029 -0.0418 -0.0303        0.772  0.653  0.891
## trump-supertuesday.txt         0.0081 0.0031  0.0020  0.0143        1.690  1.563  1.817
## trump-victory-newjersey.txt   -0.0617 0.0032 -0.0679 -0.0555        0.239  0.110  0.368
```

Smoothing or no smoothing?

- If reference documents have non-overlap in vocabularly (words exclusively used by one speaker, but not the other), this can move scores around significantly.
- Words exclusively used by one speaker should carry significant information about the underlying θ.
- However, if a word is exclusively used by one speaker this could also reflect a rare event not necessarily carrying information.
- Laplace smoothing adds $+1$ to the word count, so that an incidence of a word being used once by one speaker and never by the other does not get a wordscore of 1, but of $1/2$.
- Scores for words more frequently (but exclusively) used by a speaker are not dramatically affected by smoothing.

Smoothing or no smoothing?

```
ws2 <- textmodel(REFERENCE.dfm, refscores, model = "wordscores", smooth = 0)
## most trumpy words
ws2@Sw[ws2@Sw == 1][1:15]
```

```
##          cheer         newcom          frank        perpetr fundamentalist           rudi
##              1              1              1              1              1              1
##       giuliani         height           40th          salut  largest-singl          glad
##              1              1              1              1              1              1
##         pander        strateg        unbreak
##              1              1              1
```

```
## most hillary words
ws2@Sw[ws2@Sw == -1][1:15]
```

```
##  spoken   aipac  colleg student  campus  deepen   unwav  unshak allianc    iron    dome
##      -1      -1      -1      -1      -1      -1      -1      -1      -1      -1      -1
##  coalit yitzhak   rabin  forgav
##      -1      -1      -1      -1
```

```
predict(ws2)
```

```
## Predicted textmodel of type: wordscores
##
##                             textscore LBG se  ci lo  ci hi
## clinton-aipac.txt              -0.311 0.0107 -0.333 -0.290
## clinton-economy.txt            -0.308 0.0086 -0.325 -0.291
## clinton-launch.txt             -0.339 0.0095 -0.357 -0.320
## clinton-nominee.txt            -0.377 0.0147 -0.406 -0.349
## clinton-orlando.txt            -0.327 0.0113 -0.349 -0.305
## clinton-supertuesday.txt       -0.387 0.0157 -0.418 -0.356
## clinton-women.txt              -0.402 0.0114 -0.424 -0.379
## trump-aipac.txt                 0.257 0.0124  0.233  0.281
## trump-foreignpolicy.txt         0.231 0.0100  0.211  0.251
## trump-launch.txt                0.259 0.0088  0.242  0.276
## trump-orlando.txt               0.197 0.0100  0.177  0.216
## trump-supertuesday.txt          0.271 0.0088  0.253  0.288
## trump-victory-newjersey.txt     0.237 0.0145  0.208  0.265
```

Who sounds more like Hillary?

```
# LOAD SOME OTHER SPEECHES, TURN INTO DFM AND PREDICT BASED ON WORD SCORES
TEXT <- readLines(con = "../Data/speaches-2016-election.json")
VIRGIN <- lapply(TEXT, function(x) data.frame(fromJSON(x)))
VIRGIN <- data.table(rbindlist(VIRGIN))
VIRGIN <- VIRGIN[year(date) > 2008]
VIRGIN <- VIRGIN[, list(text = paste(text, collapse = " ")), by = c("speaker_name", "speaker_party")]
VIRGIN.corp <- corpus(VIRGIN$text)
docnames(VIRGIN.corp) <- VIRGIN$speaker_name

VIRGIN.dfm <- dfm(VIRGIN.corp, toLower = TRUE, removeNumbers = TRUE, removePunct = TRUE, removeSeparators = TR
    stem = TRUE, ignoredFeatures = stopwords("english"))

predict(ws2, VIRGIN.dfm)
```

```
## Predicted textmodel of type: wordscores
##
##                 textscore LBG se   ci lo   ci hi
## Hillary Clinton   -0.2311 0.0105 -0.2516 -0.2106
## Mike Pence        -0.1105 0.0025 -0.1155 -0.1056
## Bernie Sanders    -0.0936 0.0011 -0.0958 -0.0915
## Jim Webb          -0.0888 0.0032 -0.0951 -0.0824
## Joseph Biden      -0.1052 0.0122 -0.1291 -0.0812
## Lindsey Graham    -0.0544 0.0018 -0.0578 -0.0509
## Rand Paul         -0.0264 0.0015 -0.0295 -0.0234
## Marco Rubio       -0.0836 0.0017 -0.0871 -0.0802
## Ted Cruz          -0.0886 0.0017 -0.0919 -0.0853
```

```
VIRGIN$speaker_party
```

```
## [1] "D" "R" "I" "D" "D" "R" "R" "R" "R"
```

```
##
```

Formalizing Wordscores

- Lets derive Wordscores scoring function
- Suppose there be two reference texts, labeled R and D.
- In Laver, Benoit and Garry (2003) approach is to start with $P(R|w_i)$, i.e. the probability that a document is of type R if we observe word w_i.
- Using Bayes Rule we can write:

$$P(R|w_i) = \frac{P(w_i|R)P(R)}{P(w_i)} = \frac{P(w_i|R)P(R)}{P(w_i|R)P(R) + P(w_i|D)P(D)}$$

- How would we estimate this? $P(w_i|R) = \frac{C_{iR}}{C_R}$ and $P(R) = \frac{C_R}{C_R + C_D}$
- Substituting yields

$$P(R|w_i) = \frac{\frac{C_{iR}}{C_R}\frac{C_R}{C_R+C_D}}{\frac{C_{iR}}{C_R}\frac{C_R}{C_R+C_D} + \frac{C_{iD}}{C_D}\frac{C_D}{C_R+C_D}} = \frac{C_{iR}}{C_{iR} + C_{iD}}$$

Formalizing Wordscores

- So we can estimate

$$P(R|w_i) = \frac{P(w_i|R)P(R)}{P(w_i)} = \frac{P(w_i|R)P(R)}{P(w_i|R)P(R) + P(w_i|D)P(D)}$$

- By

$$P(R|w_i) = \frac{C_{iR}}{C_{iR} + C_{iD}}$$

- What does the wordscore algorithm ask us to compute in the two label case?

$$P_{iR} = \frac{F_{iR}}{F_{iR} + F_{iD}} = \frac{\frac{C_{iR}}{C_R}}{\frac{C_{iR}}{C_R} + \frac{C_{iD}}{C_D}}$$

- For similar length documents $C_R \approx C_D$, so the two are nearly identical.
- So we can think of the P_{ij} as being the posterior probabilites.

Formalizing Wordscores

- So how does our wordscore actually look like in the two reference case with reference scores (+1) and (-1)?

$$S_i = 1P_{iR} - 1P_{iD}$$

- I.e. we can think of this as simply being

$$S_i \approx P(R|w_i) - P(D|w_i)$$

- The difference in the word level posteriors.

Wordscores

- We have briefly introduced wordscores that are most commonly used to infer underlying positions along policy dimensions.
- These are quite popular nowadays and are easily implementable.
- We next turn to Bayesscore which is an alternative scaling method that draws upon the Naive Bayes classifier logic.
- We will highlight how Bayesscoreing and Wordscores are related by formally comparing them.

Plan

Scaling Approaches

Wordscores approach

Bayesscore

Bayesscore at the word level

- As indicated, Wordscores method does not actually constitute a formal statistical method, but rather, describes an algorithm.
- Formalization suggested that wordscores are essentially differences in posterior probabilities of class given a word.
- Bayesscores are more formally derived and closely related to Wordscores

Bayesscore at the word level

Suppose there are two classes, here R and D. We observe some word count w_i and want to know what is

$$P(R|w_i) = \frac{P(w_i|R)P(R)}{P(w_i)}$$

Similarly, we have

$$P(D|w_i) = \frac{P(w_i|D)P(D)}{P(w_i)}$$

Then the likelihood ratio for word w_i becomes

$$\frac{P(R|w_i)}{P(D|w_i)} = \frac{P(w_i|R)P(R)}{P(w_i|D)P(D)}$$

Beauchamp (2012) calls this Bayesscore.

Bayesscore at the word level

Converting this to to logs

$$\log \frac{P(R|w_i)}{P(D|w_i)} = \log \frac{P(R)}{P(D)} + \log \frac{P(w_i|R)}{P(w_i|D)}$$

The priors $P(R)$ and $P(D)$ are fixed for all w_i and thus do not affect the relative ordering, when computing a score.

- The individual word score $S_i = \log \frac{P(w_i|R)}{P(w_i|D)}$.
- This word score is thus proportional to the ratio $\log \frac{P(R|w_i)}{P(D|w_i)}$.
- Typically we assume priors $P(R) = P(D)$, in which case the ratio of logs is 0 (the constant).
- This implies that the LHS and RHS are identical.

$\Rightarrow$ we estimate $P(w_i|R) \approx F_{iR}$ as the share of words that word i makes up in reference document R.

Bayesscore at word level for our example

```
bs <- textmodel(REFERENCE.dfm, refscores, model = "NB", smooth = 1)

## F_ij are presented in this matrix - the probability of a word given class
head(t(bs$PwGc))
```

```
## 6 x 2 Matrix of class "dgeMatrix"
##          docs
## features        -1       1
##   clinton 0.002329 0.00522
##   thank   0.003594 0.00406
##   much    0.001930 0.00374
##   applaus 0.014308 0.00993
##   wonder  0.000732 0.00110
##   see     0.001730 0.00181
```

```
# Hillary
sort(log(bs$PwGc[1, ]/bs$PwGc[2, ]), decreasing = TRUE)[1:20]
```

```
##   colleg  student   effort      men   deserv   growth progress    north    incom  centuri
##     3.17     2.98     2.98     2.80     2.80     2.74     2.67     2.67     2.67     2.60
##    break     walk     code  teacher     isra  partner     21st   belong   invest  economi
##     2.52     2.43     2.43     2.43     2.33     2.33     2.33     2.33     2.30     2.29
```

```
# Trump
sort(log(bs$PwGc[1, ]/bs$PwGc[2, ]), decreasing = FALSE)[1:20]
```

```
##  billion    cheer   disast     nice       ok  probabl     amaz somebodi    islam  tremend
##    -3.49    -3.19    -2.91    -2.91    -2.86    -2.80    -2.74    -2.74    -2.68    -2.68
##  thought   nobodi    china     ford   mexico     beat   border     folk   stupid  destroy
##    -2.68    -2.61    -2.61    -2.61    -2.53    -2.53    -2.45    -2.41    -2.37    -2.37
```

Priors versus Posteriors: Bayesscore at word level

$$S_i = \log \frac{P(w_i|R)}{P(w_i|D)}$$

```
# Hillary
sort(log(bs$PwGc[1, ]/bs$PwGc[2, ]), decreasing = TRUE)[1:20]
```

```
##   colleg  student   effort      men   deserv   growth progress    north    incom  centuri
##     3.17     2.98     2.98     2.80     2.80     2.74     2.67     2.67     2.67     2.60
##    break     walk     code  teacher     isra  partner     21st   belong   invest  economi
##     2.52     2.43     2.43     2.43     2.33     2.33     2.33     2.33     2.30     2.29
```

$$\log \frac{P(R|w_i)}{P(D|w_i)}$$

```
# Hillary this ranking is identical when looking at the ratio of the posteriors
sort(log(bs$PcGw[1, ]/bs$PcGw[2, ]), decreasing = TRUE)[1:20]
```

```
##   colleg  student   effort      men   deserv   growth progress    north    incom  centuri
##     3.17     2.98     2.98     2.80     2.80     2.74     2.67     2.67     2.67     2.60
##    break     walk     code  teacher     isra  partner     21st   belong   invest  economi
##     2.52     2.43     2.43     2.43     2.33     2.33     2.33     2.33     2.30     2.29
```

Using Bayesscores to scoring a new document

We can score reference documents as

$$S^{bayesscore} = \sum_{i=1}^{I} \log \frac{P(w_i|R)}{P(w_i|D)}$$

Scoring a new document V as

$$S_V^{bayesscore} = \sum_{i=1}^{I} C_{iV} \log \frac{F_{iR}}{F_{iD}}$$

Practically - to normalize by document length - we usually compute

$$S_V^{bayesscore} = \sum_{i=1}^{I} \frac{C_{iV}}{C_V} \log \frac{F_{iR}}{F_{iD}}$$

The log transformation implies that we have an issue with zero counts in numerator and denominator. Hence its common to use Laplace smoothing or to restrict set of features to those that have non zero count in *both* reference documents.

Implicit assumption: Naive Bayes Assumption

When computing the scores, we make the simplifying assumption that the words making w_i making up our texts are conditionally independently drawn from some distribution.
Suppose our document vector is S, we assume that:

$$P(S|R) = \prod_{i=1}^{V} P(w_i|R)$$

which yields

$$P(R|S) = \frac{\prod_{i=1}^{V} P(w_i|R)P(R)}{P(S)}$$

Wordscores vs Bayesscore

- Remember that Worscores are approximately (in the two class case with reference class values -1 and +1) are

$$S_i^{wordscore} \approx P(R|w_i) - P(D|w_i)$$

- Here our Bayesscore is given by

$$S_i^{bayesscore} = \log \frac{P(w_i|R)}{P(w_i|D)} = \log P(w_i|R) - \log P(w_i|D)$$

- So in the two reference class case, Bayes scores are just the log differences, while wordscores are the absolute differences.
- Thats why word scores and bayesscores are very similar.

Plotting Wordscores against Bayesscores - no smoothing