RECSM Summer School: Text Analysis

Pablo Barberá

School of International Relations
University of Southern California
`pablobarbera.com`

Networked Democracy Lab
`www.netdem.org`

Course website:

github.com/pablobarbera/big-data-upf

Text as data

Text as data


Kukuriku
koalicija
Plan 21


the TRUE NORTH
STRONG AND FREE
Stephen Harper's plan for Canadians
Conservative


Et retfærdigt
Danmark


ΕΚΛΟΓΙΚΗ ΔΙΑΚΗΡΥΞΗ
ΣΥΡΙΖΑ
ισχυρός
στη Βουλή
και στους αγώνες
Είναι στο χέρι μας!
ΣΥΡΙΖΑ
ΣΥΝΑΣΠΙΣΜΟΣ ΡΙΖΟΣΠΑΣΤΙΚΗΣ ΑΡΙΣΤΕΡΑΣ
www.syriza.gr

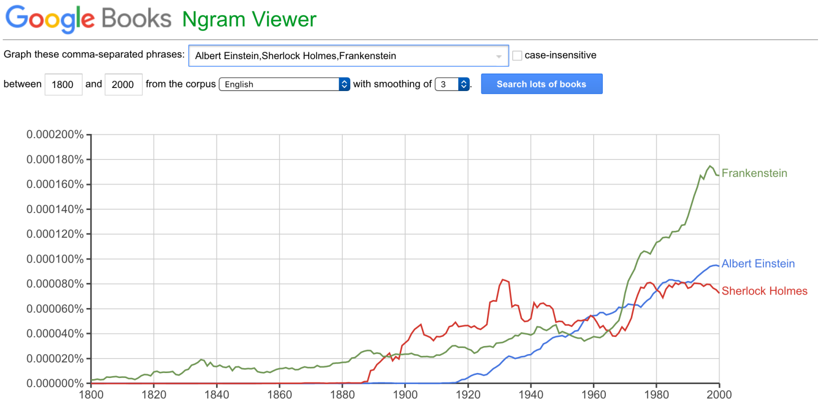

SPÖ
WAHLMANIFEST


GRÜNE NEUE
GESELLSCHAFTS-
VERTRAG
KLIMA • ARBEIT • GERECHTIGKEIT • FREIHEIT


WIR HABEN DIE KRAFT
GEMEINSAM
FÜR UNSER LAND.
REGIERUNGSPROGRAMM 2009 – 2013
CDU CSU


Sozial und
Demokratisch.
Anpacken. Für Deutschland.
Das Regierungsprogramm der SPD
SPD

Text as data



Google Books Ngram Viewer
Graph these comma-separated phrases:
Albert Einstein,Sherlock Holmes,Frankenstein
case-insensitive
between
1800
and
2000
from the corpus
English
with smoothing of
3
Search lots of books
0.000200%
0.000180%
0.000160%
0.000140%
0.000120%
0.000100%
0.000080%
0.000060%
0.000040%
0.000020%
0.000000%
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
Frankenstein
Albert Einstein
Sherlock Holmes

Text as data


facebook
XING
Gefällt mir
Instagram
twitter
Linked in
Pinterest
skype
flickr
You Tube

Overview of text as data methods

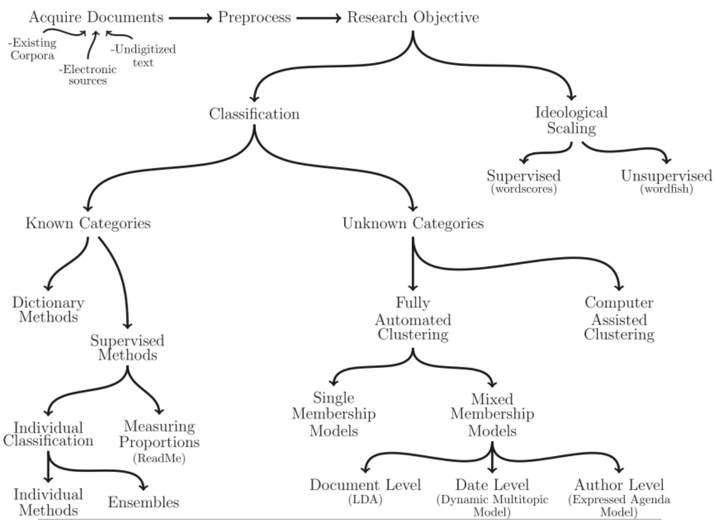


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

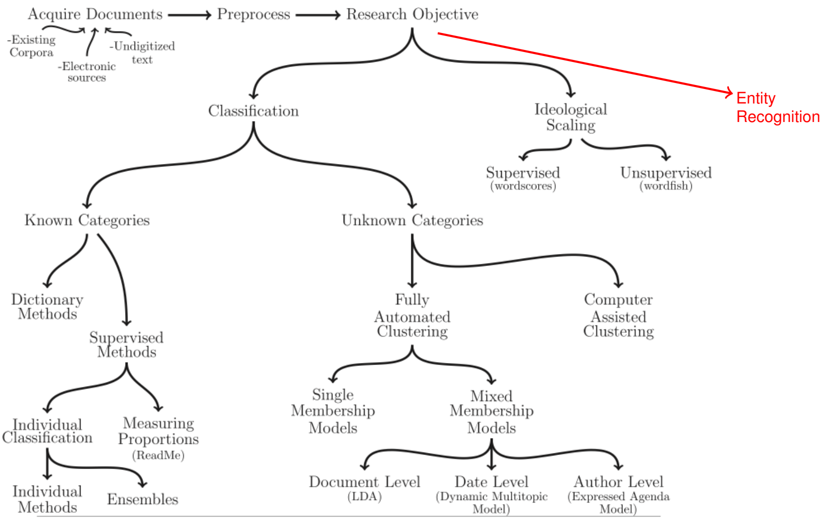


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

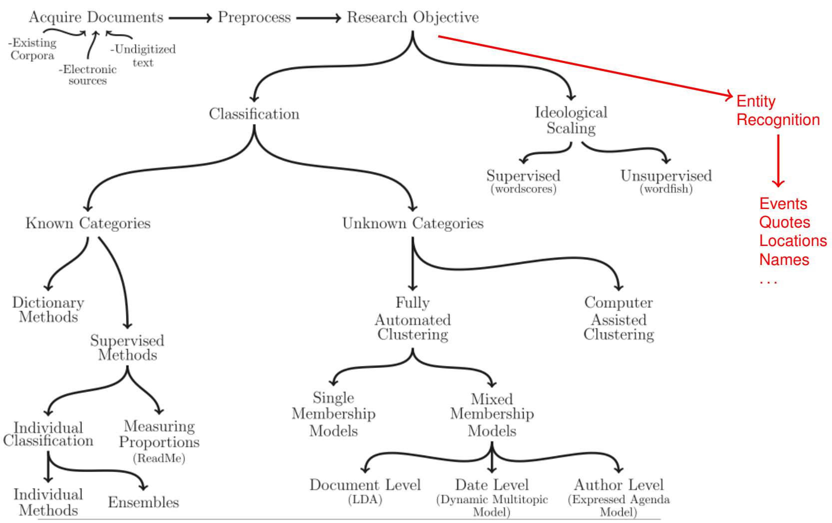


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

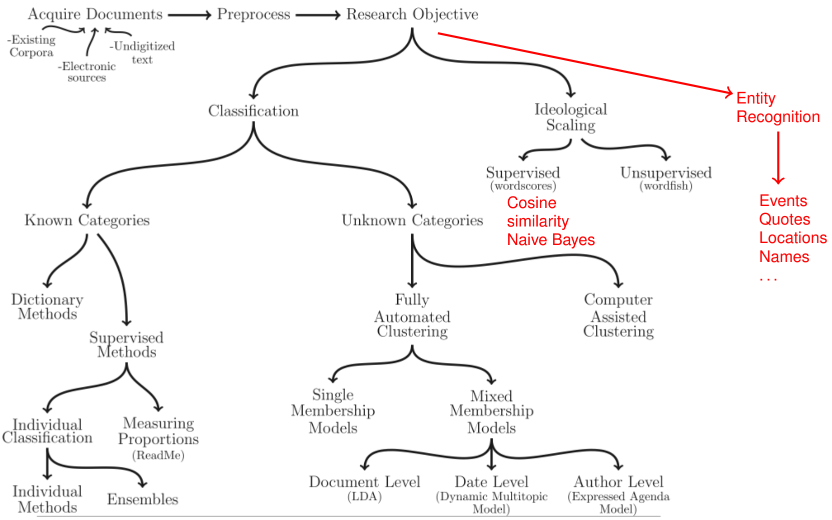


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

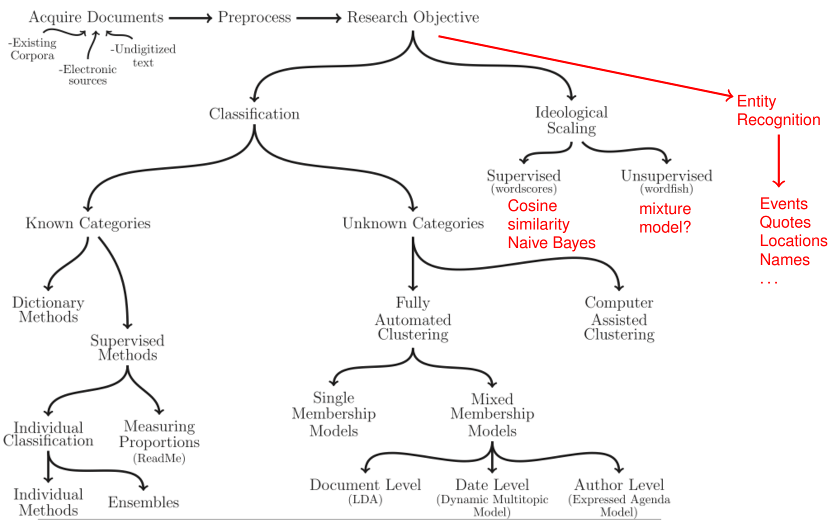


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

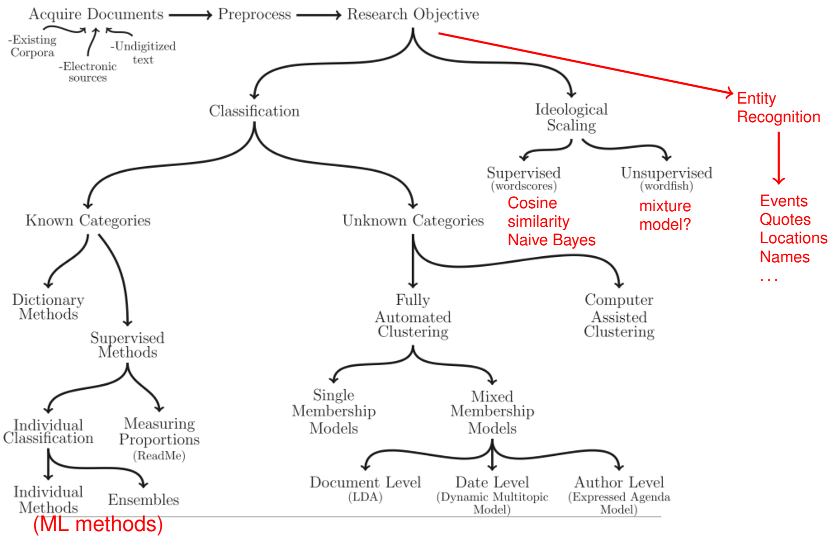


Fig. 1 in Grimmer and Stewart (2013)

Overview of text as data methods

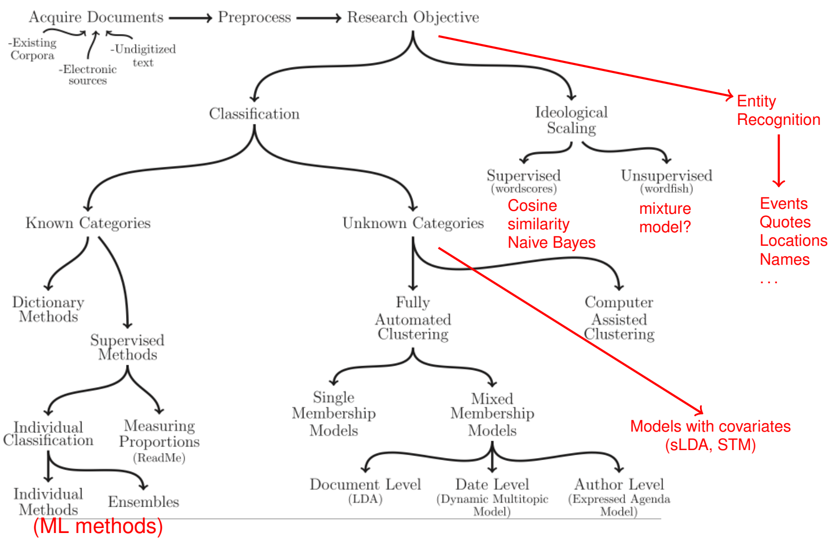


Fig. 1 in Grimmer and Stewart (2013)

From words to numbers

1. Bag-of-words assumption

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
    - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
    - Stemming / lemmatization

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
   - Stemming / lemmatization
   - Part-of-speech tagging

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
   - Stemming / lemmatization
   - Part-of-speech tagging
3. Document-term matrix

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
   - Stemming / lemmatization
   - Part-of-speech tagging
3. Document-term matrix
   - $\mathbf{W}$: matrix of N documents by M unique words

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
   - Stemming / lemmatization
   - Part-of-speech tagging
3. Document-term matrix
   - **W**: matrix of N documents by M unique words
   - W_{im}= number of times m-th words appears in i-th document.

From words to numbers

1. Bag-of-words assumption
2. Pre-processing text
   - Capitalization, cleaning digits/URLs, removing stopwords and sparse words, etc.
   - Stemming / lemmatization
   - Part-of-speech tagging
3. Document-term matrix
   - $\mathbf{W}$: matrix of N documents by M unique words
   - W_{im}= number of times m-th words appears in i-th document.
   - Usually large matrix, but sparse (so it fits in memory)

From words to numbers

From words to numbers

1. Preprocess text:

“@MEPcandidate thank you and congratulations, you’re the best #EP2014”

“@MEPcandidate You’re an idiot, I would never vote for you”

From words to numbers

From words to numbers

1. Preprocess text: lowercase,

“@mepcandidate thank you and congratulations, you’re the best #ep2014”

“mepcandidate you’re an idiot, i would never vote for you”

From words to numbers

From words to numbers

1. Preprocess text: lowercase, remove stopwords and punctuation,

   “@mepcandidate thank ~~you~~ ~~and~~ congratulations~~,~~ you’re ~~the~~ best #ep2014”

   “@mepcandidate you’re ~~an~~ idiot~~,~~ ~~i~~ ~~would~~ never vote ~~for~~ ~~you~~”

From words to numbers

From words to numbers

1. Preprocess text: lowercase, remove stopwords and punctuation, stem,

   “@ thank congratul~~ations~~, you’r~~e~~ best #ep2014”

   “@ you’r~~e~~ idiot never vote”

From words to numbers

From words to numbers

1. Preprocess text: lowercase, remove stopwords and punctuation, stem, tokenize into unigrams and bigrams (bag-of-words assumption)

   [@, thank, congratul, you’r, best, #ep2014, @ thank, thank congratul, congratul you’r, you’r best, best, best #ep2014]

   [@, you’r, idiot, never, vote, @ you’r, you’r idiot, idiot never, never vote]

From words to numbers

From words to numbers

1. Preprocess text: lowercase, remove stopwords and punctuation, stem, tokenize into unigrams and bigrams (bag-of-words assumption)

   [@, thank, congratul, you'r, best, #ep2014, @ thank, thank congratul, congratul you'r, you'r best, best, best #ep2014]

   [@, you'r, idiot, never, vote, @ you'r, you'r idiot, idiot never, never vote]

2. Document-term matrix:
   - **W**: matrix of N documents by M unique n-grams
   - w_{im}= number of times m-th n-gram appears in i-th document.

	@	thank	congratul	you'r	#ep2014	@ thank	...	M words
Document 1	1	1	1	1	1	1	...	
Document 2	1	0	0	1	0	0	...	
...								
Document n	0	1	1	0	0	0	...	

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
    - Positive or negative, for sentiment
    - Sad, happy, angry, anxious... for emotions
    - Insight, causation, discrepancy, tentative... for cognitive processes

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

  many others: see LIWC, VADER, SentiStrength, LexiCoder...

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

    many others: see LIWC, VADER, SentiStrength, LexiCoder...
- Count number of times they appear in each document

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

    many others: see LIWC, VADER, SentiStrength, LexiCoder...
- Count number of times they appear in each document
- Normalize by document length (optional)

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

  many others: see LIWC, VADER, SentiStrength, LexiCoder...
- Count number of times they appear in each document
- Normalize by document length (optional)
- Validate, validate, validate.

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

    many others: see LIWC, VADER, SentiStrength, LexiCoder...
- Count number of times they appear in each document
- Normalize by document length (optional)
- Validate, validate, validate.
  - Check sensitivity of results to exclusion of specific words

Dictionary methods

Classifying documents when categories are known using dictionaries:

- Lists of words that correspond to each category:
  - Positive or negative, for sentiment
  - Sad, happy, angry, anxious... for emotions
  - Insight, causation, discrepancy, tentative... for cognitive processes
  - Sexism, homophobia, xenophobia, racism... for hate speech

    many others: see LIWC, VADER, SentiStrength, LexiCoder...
- Count number of times they appear in each document
- Normalize by document length (optional)
- Validate, validate, validate.
  - Check sensitivity of results to exclusion of specific words
  - Code a few documents manually and see if dictionary prediction aligns with human coding of document

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...
What we need:

- Hand-coded dataset (labeled), to be split into:

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...
What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...
What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier
  - Validation/Test set: used to validate the classifier

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...
What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier
  - Validation/Test set: used to validate the classifier
- Method to extrapolate from hand coding to unlabeled documents (classifier):

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...

What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier
  - Validation/Test set: used to validate the classifier
- Method to extrapolate from hand coding to unlabeled documents (classifier):
  - SVM, Naive Bayes, regularized regression, BART, ensemble methods...

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...

What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier
  - Validation/Test set: used to validate the classifier
- Method to extrapolate from hand coding to unlabeled documents (classifier):
  - SVM, Naive Bayes, regularized regression, BART, ensemble methods...
- Approach to validate classifier: cross-validation

Supervised machine learning

Goal: classify documents into pre existing categories.
e.g. authors of documents, sentiment of tweets, ideological position of parties based on manifestos, tone of movie reviews...

What we need:

- Hand-coded dataset (labeled), to be split into:
  - Training set : used to train the classifier
  - Validation/Test set: used to validate the classifier
- Method to extrapolate from hand coding to unlabeled documents (classifier):
  - SVM, Naive Bayes, regularized regression, BART, ensemble methods...
- Approach to validate classifier: cross-validation
- Performance metric to choose best classifier and avoid overfitting: confusion matrix, AUC, accuracy, precision, recall...

Performance metrics

Confusion matrix:

	Actual Label	
Classification (algorithm)	Liberal	Conservative
Liberal	True Liberal	False Liberal
Conservative	False Conservative	True Conservative

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

$$\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}$$

$$\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}$$

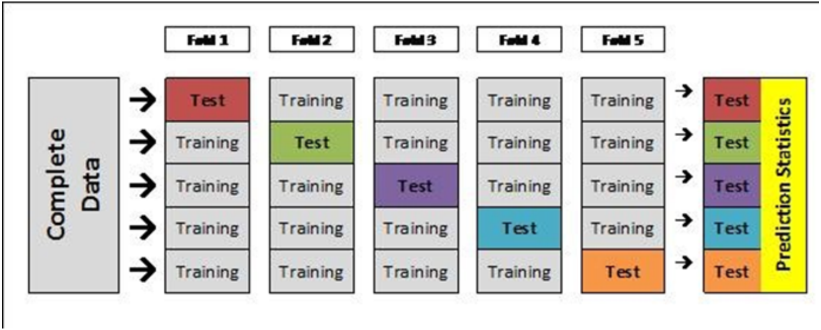
$$F_{\text{Liberal}} = \frac{2\text{Precision}_{\text{Liberal}}\text{Recall}_{\text{Liberal}}}{\text{Precision}_{\text{Liberal}} + \text{Recall}_{\text{Liberal}}}$$

Source: Grimmer, 2014, “Text as Data” course week 14

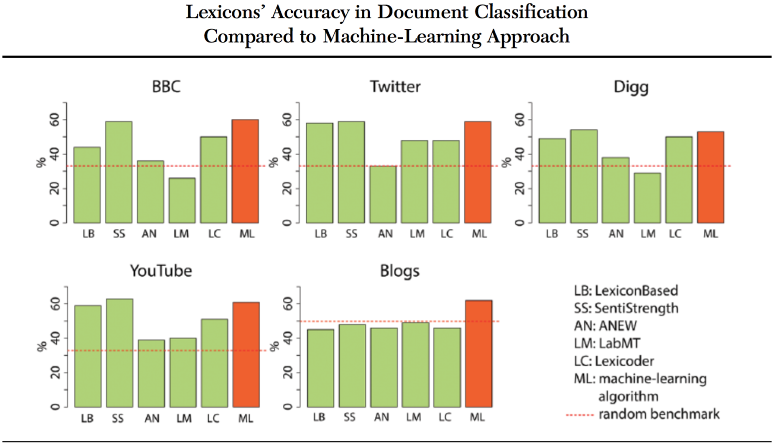
Cross-validation

Intuition:

- Create K training and test sets (“folds”) within training set.
- For each k in K, run classifier and estimate performance in test set within fold.
- Why? Find best classifier and avoid overfitting



Dictionaries vs supervised learning



Source: González-Bailón and Paltoglou (2015)

Regularized regression

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow$ {liberal, conservative}
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$
\begin{aligned}
f(\beta, \boldsymbol{X}, \boldsymbol{Y}) &= \sum_{i=1}^{N} \left(y_i - \beta' \boldsymbol{x}_i\right)^2 \\
\widehat{\beta} &= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta' \boldsymbol{x}_i\right)^2 \right\} \\
&= \left(\boldsymbol{X}' \boldsymbol{X}\right)^{-1} \boldsymbol{X}' \boldsymbol{Y}
\end{aligned}
$$

Problem:

- J will likely be large (perhaps $J > N$)
- There many correlated variables

Source: Grimmer, 2014, “Text as Data” course week 15

Regularized regression

Penalty for model complexity

$$f(\beta, \boldsymbol{X}, \boldsymbol{Y}) \;=\; \sum_{i=1}^{N}\left(y_i - \;\beta_0 + \sum_{j=1}^{J}\beta_j x_{ij}\right)^2 + \underbrace{\lambda\sum_{j=1}^{J}\beta_j^2}_{\text{Penalty}}$$

where:

- $\beta_0 \leadsto$ intercept
- $\lambda \leadsto$ penalty parameter

Source: Grimmer, 2014, “Text as Data” course week 15

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:
  - $P_{rm} = \frac{F_{rm}}{\sum_r F_{rm}} \rightarrow$ (Prob. we are reading r if we observe m)

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of “reference” texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:
  - $P_{rm} = \frac{F_{rm}}{\sum_r F_{rm}} \rightarrow$ (Prob. we are reading r if we observe m)
  - Wordscore $S_{md} = \sum_r (P_{rm} \times A_{rd})$

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:
  - $P_{rm} = \frac{F_{rm}}{\sum_r F_{rm}} \rightarrow$ (Prob. we are reading r if we observe m)
  - Wordscore $S_{md} = \sum_r (P_{rm} \times A_{rd})$
- Scores for "virgin" texts:

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:
  - $P_{rm} = \frac{F_{rm}}{\sum_r F_{rm}} \rightarrow$ (Prob. we are reading r if we observe m)
  - Wordscore $S_{md} = \sum_r (P_{rm} \times A_{rd})$
- Scores for "virgin" texts:
  - $S_{vd} = \sum_w (F_{vm} \times S_{md}) \rightarrow$ (weighted average of scored words)

Wordscores (Laver, Benoit, Garry, 2003, APSR)

- Goal: estimate positions on a latent ideological scale
- Data = document-term matrix $\mathbf{W_R}$ for set of "reference" texts, each with known A_{rd}, a policy position on dimension d.
- Compute $\mathbf{F}$, where F_{rm} is relative frequency of word m over the total number of words in document r.
- Scores for individual words:
  - $P_{rm} = \frac{F_{rm}}{\sum_r F_{rm}} \rightarrow$ (Prob. we are reading r if we observe m)
  - Wordscore $S_{md} = \sum_r (P_{rm} \times A_{rd})$
- Scores for "virgin" texts:
  - $S_{vd} = \sum_w (F_{vm} \times S_{md}) \rightarrow$ (weighted average of scored words)
  - $S^*_{vd} = (S_{vd} - \overline{S_{vd}}) \left(\frac{SD_{rd}}{SD_{vd}} \right) + \overline{S_{vd}} \rightarrow$ Rescaled scores.