

# Training Session: Supervised learning

**Pablo Barberá**  
Networked Democracy Lab  
University of Southern California

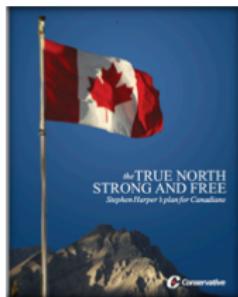
March 6, 2017

materials: [github.com/NetDem-USC/training](https://github.com/NetDem-USC/training)

# Text as data

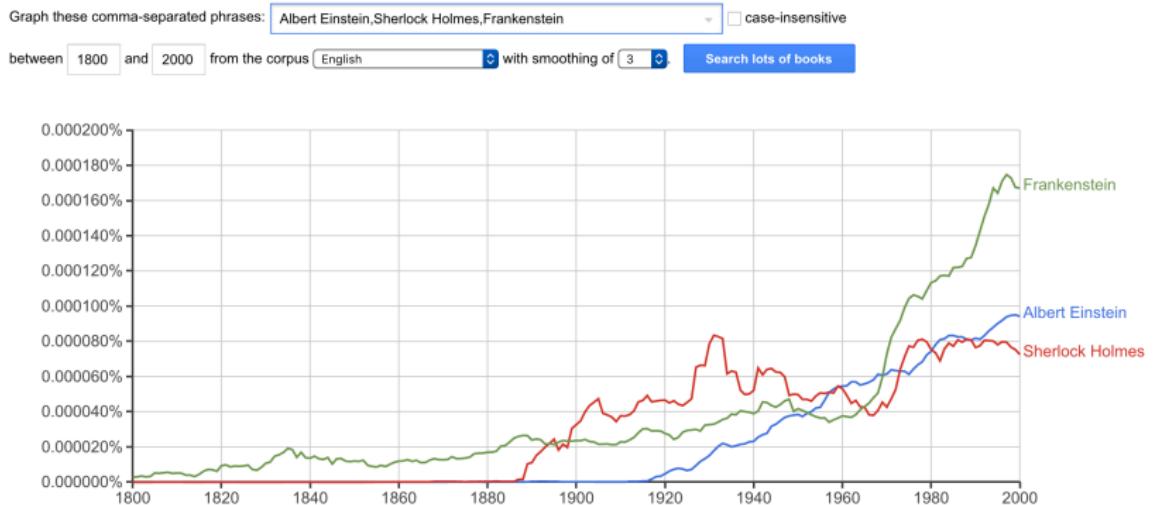


# Text as data



# Text as data

## Google Books Ngram Viewer



# Text as data



# Overview of text as data methods

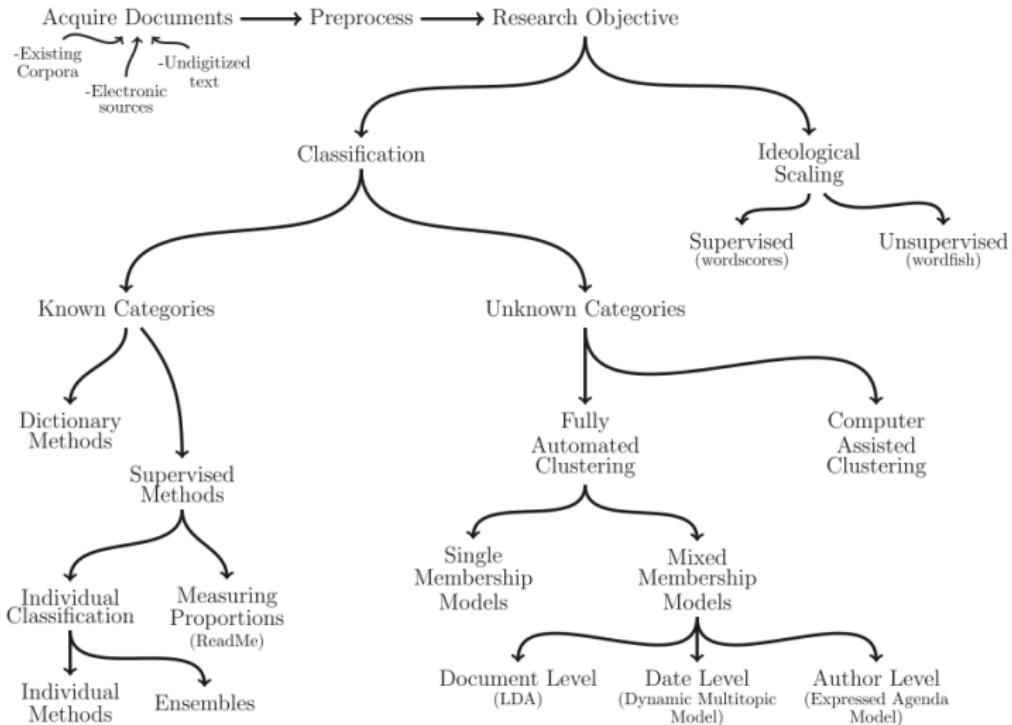


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

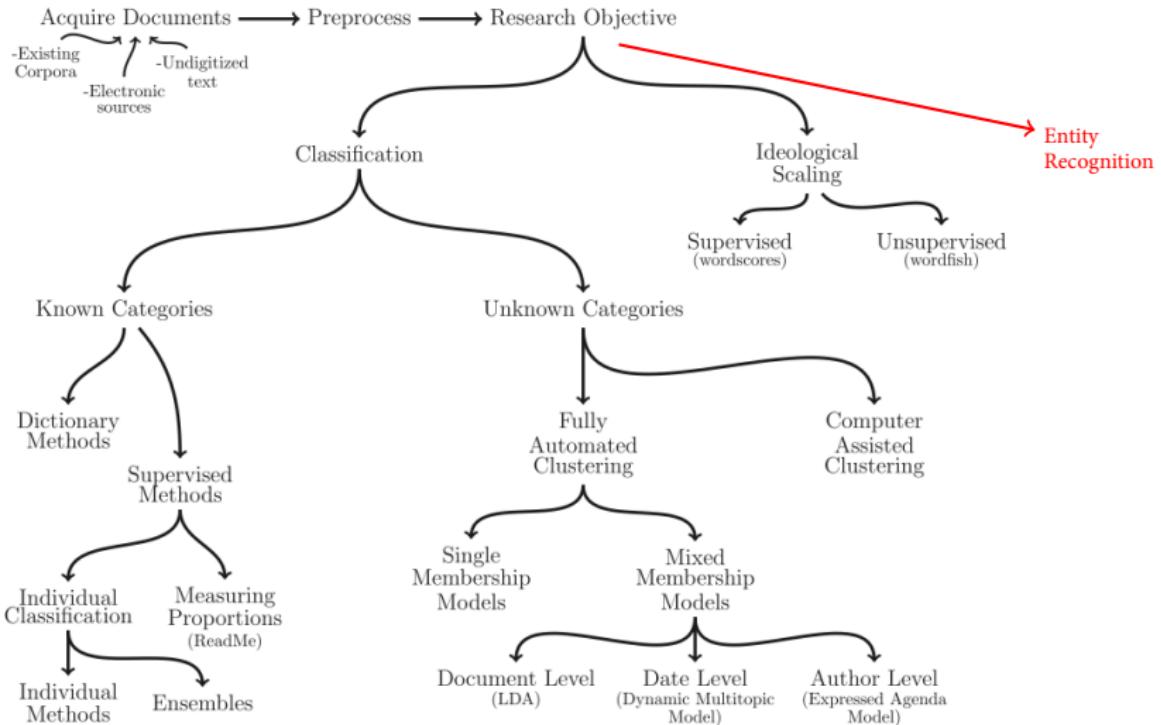


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

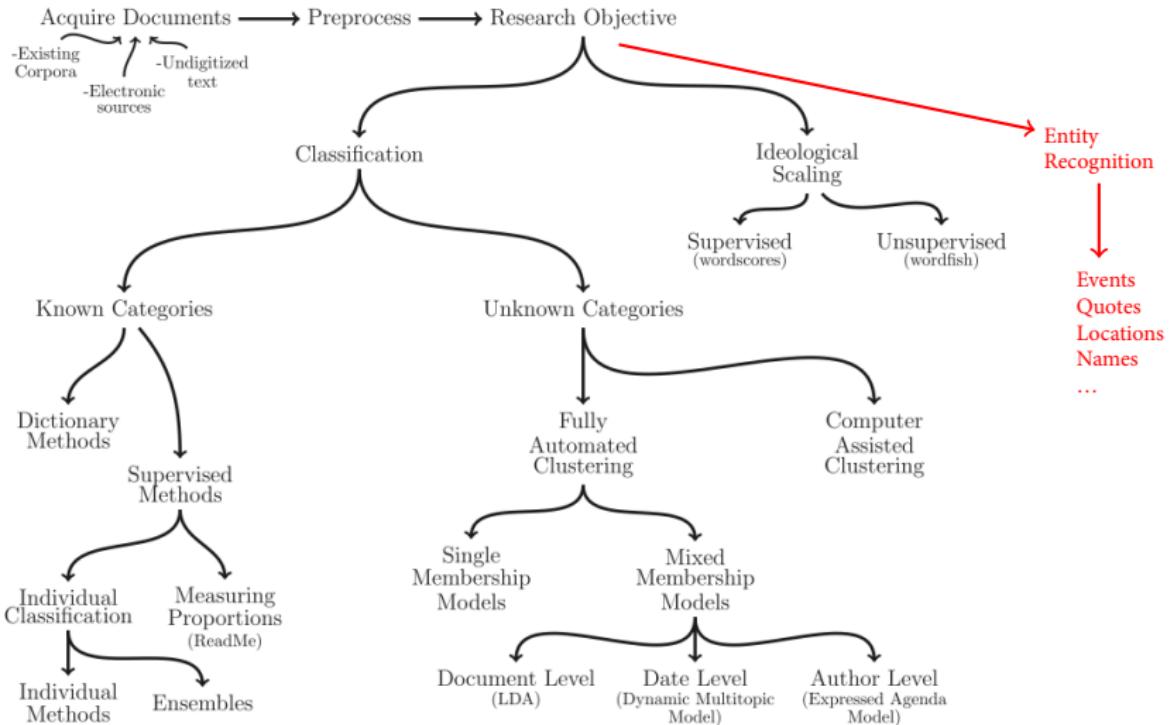


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

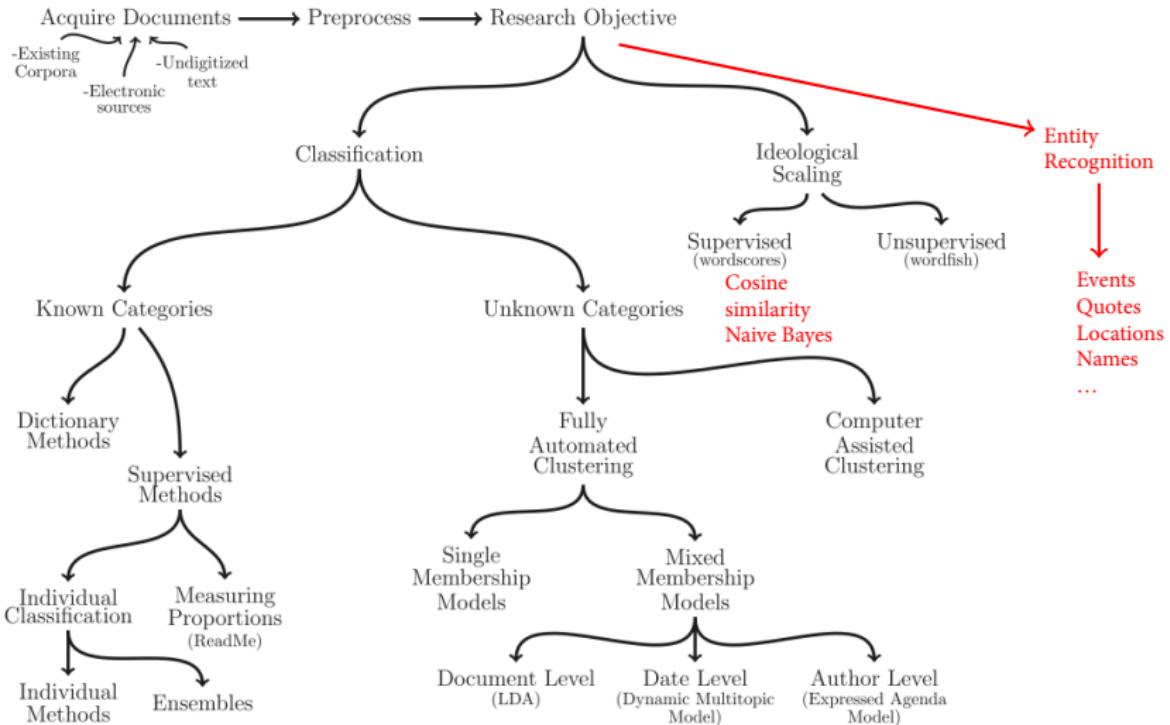


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

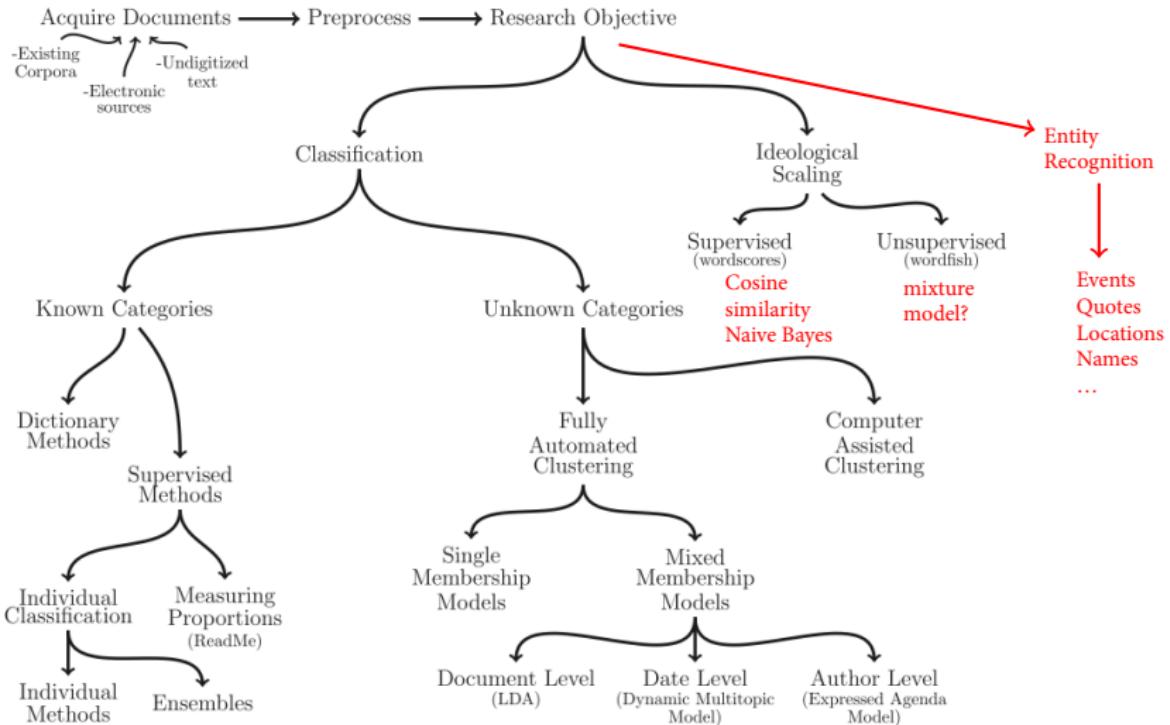


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

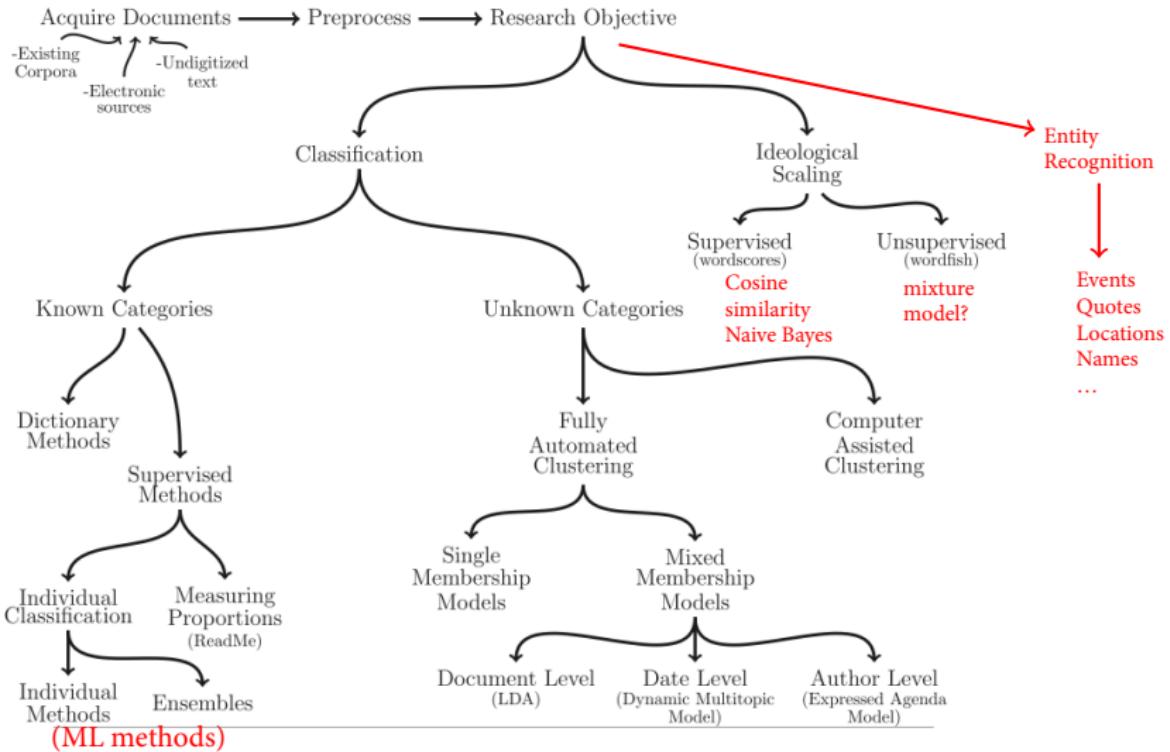


Fig. 1 in Grimmer and Stewart (2013)

# Overview of text as data methods

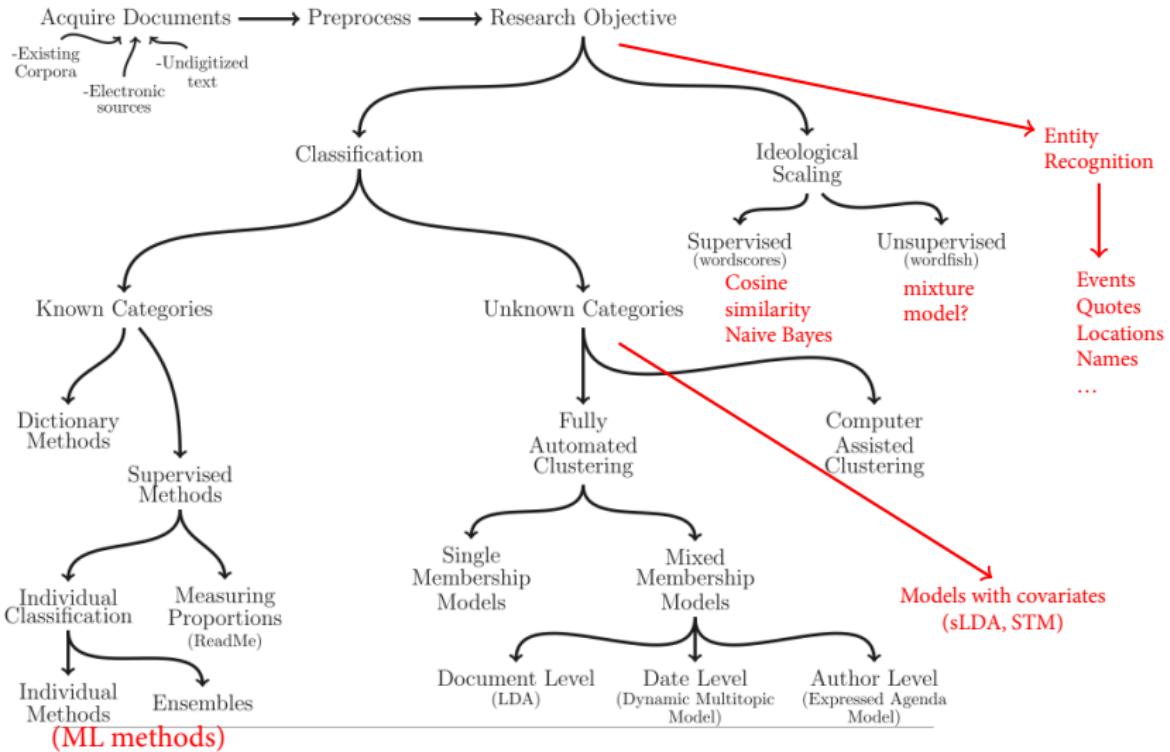


Fig. 1 in Grimmer and Stewart (2013)

# 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
  - ▶  $\mathbf{W}$ : 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 congratulations, you’re best #ep2014”

“ you’re 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:

- $\mathbf{W}$ : matrix of  $N$  documents by  $M$  unique words
- $W_{im}$  = number of times  $m$ -th word 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	n	0	1	1	0	0	...

# 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...

## Regularized regression

Suppose we have  $N$  documents, with each document  $i$  having label  $y_i \in \{-1, 1\} \rightsquigarrow \{\text{liberal, conservative}\}$

We represent each document  $i$  is  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iJ})$ .

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

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

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

where:

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

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

# Performance metrics

Confusion matrix:

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

## 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.

## 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.
- ▶ Pick value of  $\lambda$  (regularization parameter) that gives better performance

## 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.
- ▶ Pick value of  $\lambda$  (regularization parameter) that gives better performance
- ▶ Train classifier with full dataset and compute performance on test set

## 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.
- ▶ Pick value of  $\lambda$  (regularization parameter) that gives better performance
- ▶ Train classifier with full dataset and compute performance on test set
- ▶ Why? Avoids overfitting.

## 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:
  - ▶  $S_{vd} = \sum_w (F_{wm} \times S_{md}) \rightarrow$  (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:
  - ▶  $S_{vd} = \sum_w (F_{wm} \times S_{md}) \rightarrow$  (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.