Sequencing Legal DNA

NLP for Law and Political Economy

2. Tokens and N-Grams

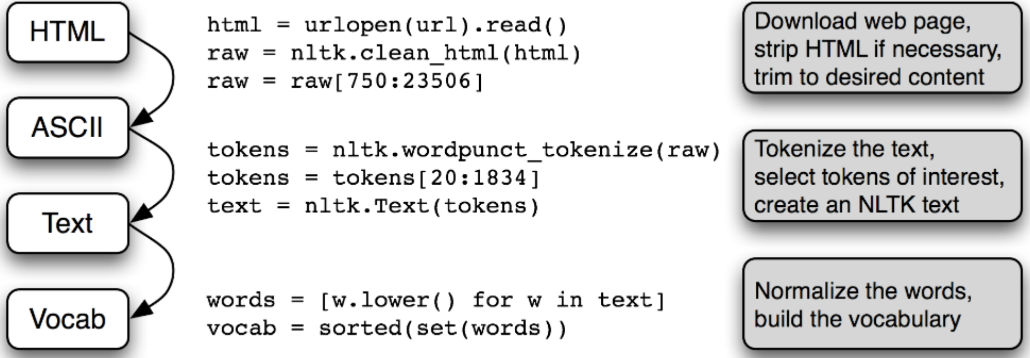
Overview

- These slides describe the process of transforming a corpus into numerical data that can be used in statistical analysis.
- Input:
  - A set of documents (e.g. text files), D.
- Output:
  - A matrix, X, containing statistics about word/phrase frequencies in those documents.

Goals of Featurization

- To summarize: A major goal of featurization is to produce features that are
  - **predictive** in the learning task
  - **interpretable** by human investigators
  - **tractable** enough to be easy to work with

The NLP Pipeline



Source: NLTK Book, Chapter 3.

Logistics: Homework Assignments

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th
- **Option A: Two Response Essays**
  - read and discuss/critique two of the *Applications* readings
    - 250+ words (half -page to a page) eac

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th
- **Option A: Two Response Essays**
  - read and discuss/critique two of the *Applications* readings
    - 250+ words (half -page to a page) eac
  - We will post anonymized essays to the forum; part of the grade is voting/commenting on other students' essays.

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th
- **Option A: Two Response Essays**
  - read and discuss/critique two of the *Applications* readings
    - 250+ words (half -page to a page) eac
  - We will post anonymized essays to the forum; part of the grade is voting/commenting on other students' essays.
- **Option B: Code Replication Exercise**
  - replicate analysis from a listed paper (either *Methods* or *Applications*).
    - If code is already available, have to apply it to new data.
  - Can be done in groups of two students.

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th
- **Option A: Two Response Essays**
  - read and discuss/critique two of the *Applications* readings
    - 250+ words (half -page to a page) eac
  - We will post anonymized essays to the forum; part of the grade is voting/commenting on other students' essays.
- **Option B: Code Replication Exercise**
  - replicate analysis from a listed paper (either *Methods* or *Applications*).
    - If code is already available, have to apply it to new data.
  - Can be done in groups of two students.
- **Option C: Problem Sets**
  - challenging problem sets for students who want coding practice.

Logistics: Homework Assignments

- There are three homework assignments
  - upload to assignment dropbox (see syllabus), first one is due March 11th
- **Option A: Two Response Essays**
  - read and discuss/critique two of the *Applications* readings
    - 250+ words (half -page to a page) eac
  - We will post anonymized essays to the forum; part of the grade is voting/commenting on other students' essays.
- **Option B: Code Replication Exercise**
  - replicate analysis from a listed paper (either *Methods* or *Applications*).
    - If code is already available, have to apply it to new data.
  - Can be done in groups of two students.
- **Option C: Problem Sets**
  - challenging problem sets for students who want coding practice.
- Can also do **A/C hybrid**: do one response essay, plus any 4 questions from problem set.

Outline

Split into paragraphs/sentences

- Many tasks should be done on sentences, rather than corpora as a whole.
  - NLTK and spaCy do a good (but not perfect) job of splitting sentences, while accounting for periods on abbreviations, etc.
  - spaCy is slower but significantly better.

Split into paragraphs/sentences

- Many tasks should be done on sentences, rather than corpora as a whole.
  - NLTK and spaCy do a good (but not perfect) job of splitting sentences, while accounting for periods on abbreviations, etc.
  - spaCy is slower but significantly better.
- There isn't a grammar-based paragraph tokenizer.
  - most corpora have new paragraphs annotated.
  - or use line breaks.

Pre-processing

- An important piece of the “art” of text analysis is deciding what data to throw out.
  - Uninformative data add noise and reduce statistical precision.
  - They are also computationally costly.

Pre-processing

- An important piece of the "art" of text analysis is deciding what data to throw out.
  - Uninformative data add noise and reduce statistical precision.
  - They are also computationally costly.
- Pre-processing choices can affect down-stream results, especially in unsupervised learning tasks (Denny and Spirling 2017).
  - some features are more interpretable: "judge has" / "has discretion" vs "judge has discretion".

Capitalization

- Removing capitalization is a standard corpus normalization technique
  - usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
  - $\rightarrow$ capitalization not informative.

Capitalization

- Removing capitalization is a standard corpus normalization technique
  - usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
  - $\rightarrow$ capitalization not informative.
  - but what about "the first amendment" versus "the First Amendment"?

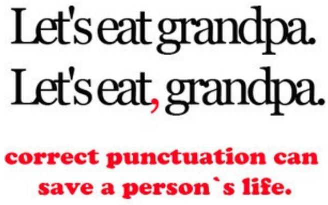
Capitalization

- Removing capitalization is a standard corpus normalization technique
  - usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
  - $\rightarrow$ capitalization not informative.
  - but what about "the first amendment" versus "the First Amendment"?
- Compromise: include capitalized version of words not at beginning of sentence.

Capitalization

- Removing capitalization is a standard corpus normalization technique
  - usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
  - $\rightarrow$ capitalization not informative.
  - but what about "the first amendment" versus "the First Amendment"?
- Compromise: include capitalized version of words not at beginning of sentence.
- For some tasks this is important – e.g. text generation.
  - Modern tokenizers take out capitalization but then add a "capitalized" token before the word.

Punctuation



Source: Chris Bail text data slides.

- inclusion of punctuation is a similar choice to capitalization.
  - usually, not informative – but needed for text generation, for example.

Numbers

1871

1949

1990

- can drop numbers, or replace with special characters; can encode magnitude for example.

Source: Chris Bail text data slides.

Drop Stopwords?

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Drop Stopwords?

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

- What about "not guilty"?
- Legal "memes" often contain stopwords:
  - "beyond a reasonable doubt"
  - "with all deliberate speed"

Drop Stopwords?

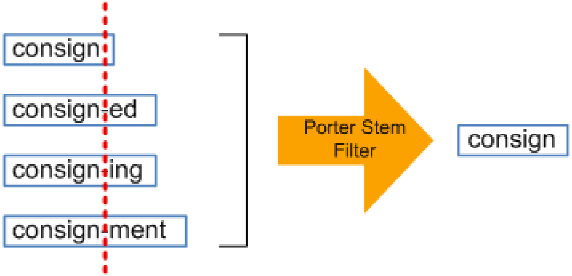
a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

- What about "<u>not</u> guilty"?
- Legal "memes" often contain stopwords:
  - "beyond a reasonable doubt"
  - "with all deliberate speed"
- can drop stopwords by themselves, but keep them as part of phrases.
- can filter out words and phrases using part-of-speech tags (later).

Stopwords lists (Kelly et al 2018)

http://www.ranks.nl/stopwords
https://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html
https://code.google.com/p/stop-words/
http://www.lextek.com/manuals/onix/stopwords1.html
http://www.lextek.com/manuals/onix/stopwords2.html
http://analytics101.com/2014/10/all-about-stop-words-for-text-mining.html
http://www.nlm.nih.gov/bsd/disted/pubmedtutorial/020_170.html
https://pypi.python.org/pypi/stop-words
https://msdn.microsof,t.com/zh-cn/library/bb164590
http://www.nltk.org/book/ch02.html

Stemming/lemmatizing



- Porter Stemmer is more aggressive than Snowball Stemmer.
- Lemmatizer produces real words, but N-grams won't make grammatical sense
  - e.g., "judges have been ruling" would become "judge have be rule"

Outline

Bag-of-words representation

- Recall the goal of this lecture:
  - Convert a corpus D to a matrix X
- In the “bag-of-words” representation, a row of X is just the frequency distribution over words in the document corresponding to that row.

Counts and frequencies

- **Document counts**: number of documents where a token appears.
- **Term counts**: number of total appearances of a token in corpus.

Counts and frequencies

- **Document counts**: number of documents where a token appears.
- **Term counts**: number of total appearances of a token in corpus.

- **Term frequency**:

$$\text{Term Frequency in document } k = \frac{\text{Term count in document } k}{\text{Total tokens in document } k}$$

Building a vocabulary

- An important featurization step is to build a vocabulary of words:
  - Compute document frequencies for all words
  - Inspect low-frequency words and determine a minimum document threshold.
    - e.g., 10 documents, or .25% of documents.

Building a vocabulary

- An important featurization step is to build a vocabulary of words:
  - Compute document frequencies for all words
  - Inspect low-frequency words and determine a minimum document threshold.
    - e.g., 10 documents, or .25% of documents.
- Can also impose more complex thresholds, e.g.:
  - appears twice in at least 20 documents
  - appears in at least 3 documents in at least 5 years

Building a vocabulary

- An important featurization step is to build a vocabulary of words:
  - Compute document frequencies for all words
  - Inspect low-frequency words and determine a minimum document threshold.
    - e.g., 10 documents, or .25% of documents.
- Can also impose more complex thresholds, e.g.:
  - appears twice in at least 20 documents
  - appears in at least 3 documents in at least 5 years
- Assign numerical identifiers to tokens to increase speed and reduce disk usage.

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log\left(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w}\right)}_{\text{Inverse Document Frequency}}$$

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w})}_{\text{Inverse Document Frequency}}$$

- The formula up-weights relatively rare words that do not appear in all documents.
  - These words are probably more distinctive of topics or differences between documents.
  - Example: A document contains 100 words, and the word appears 3 times in the document. The TF is .03. The corpus has 100 documents, and the word appears in 10 documents. the IDF is $\log(100/10) \approx 2.3$, so the TF-IDF for this document is $.03 \times 2.3 = .07$. Say the word appears in 90 out of 100 documents: Then the IDF is 0.105, with TF-IDF for this document equal to .003.

Log Entropy Weighting

- log entropy weighted frequency for term *i* in document *j* is

$$
\begin{aligned}
local\_weight_{i,j} &= log(frequency_{i,j} + 1) \\
P_{i,j} &= \frac{frequency_{i,j}}{\sum_j frequency_{i,j}} \\
global\_weight_i &= 1 + \frac{\sum_j P_{i,j} * log(P_{i,j})}{log(number\_of\_documents + 1)} \\
final\_weight_{i,j} &= local\_weight_{i,j} * global\_weight_i
\end{aligned}
$$

- Lee et al (2005) got best classification results using this weighting.

Other Transformations

- In principle could include all these weightings as separate predictors.

Other Transformations

- In principle could include all these weightings as separate predictors.
- Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently informative.

Other Transformations

- In principle could include all these weightings as separate predictors.
- Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently informative.
- Could add log counts, quadratics in counts, etc.
- Could also add pairwise interactions between word counts/frequencies.

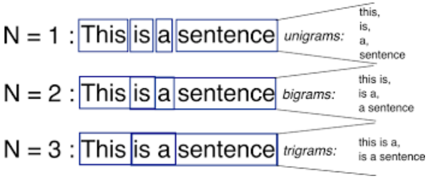
Other Transformations

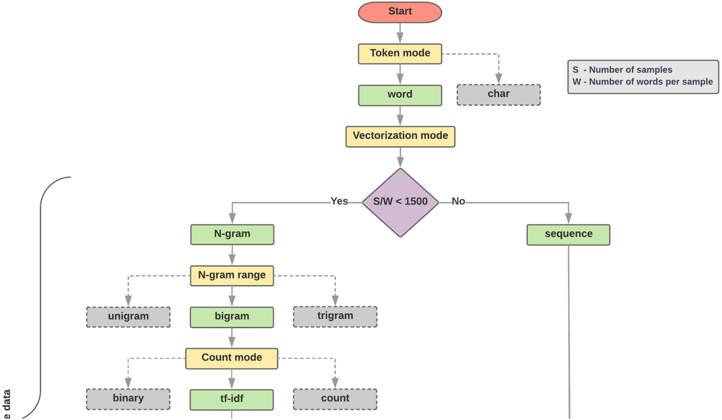
- In principle could include all these weightings as separate predictors.
- Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently informative.
- Could add log counts, quadratics in counts, etc.
- Could also add pairwise interactions between word counts/frequencies.
- These often are not done much because of the dimensionality problem.
  - could use feature selection or principal components to deal with that.

Outline

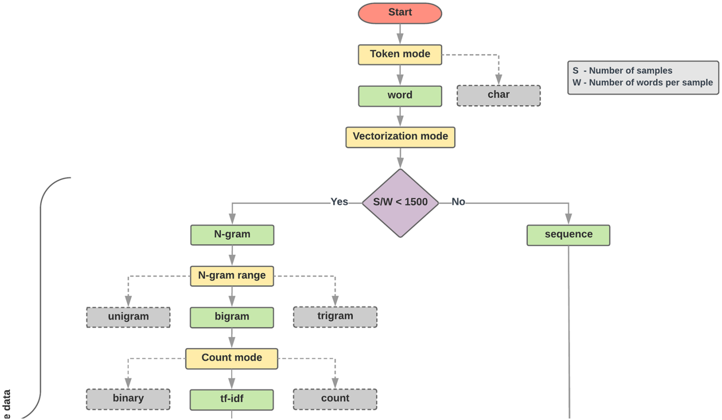
What are N-grams

- N-grams are phrases, sequences of words up to length N.
  - bigrams, trigrams, quadgrams, etc.





- Google Developers recommend **tf-idf-weighted bigrams** as a baseline specification for text classification tasks.
  - ideal for fewer, longer documents.



- Google Developers recommend **tf-idf-weighted bigrams** as a baseline specification for text classification tasks.
  - ideal for fewer, longer documents.
- With more numerous, shorter documents (rows / doclength $>$ 1500), better to use an embedded sequence (starting Week 5).

N-grams and high dimensionality

- N-grams will blow up your feature space:
  - filtering out uninformative n-grams is necessary.

N-grams and high dimensionality

- N-grams will blow up your feature space:
  - filtering out uninformative n-grams is necessary.
- Google Developers say that a feature space with $P = 20,000$ will work well for descriptive and prediction tasks.
  - I have gotten good performance with 10K or even 2K features.
  - For supervised learning tasks, a decent baseline is to build a vocabulary of 60K, then use feature selection to get down to 10K.

Feature selection using univariate comparisions

- χ^2 is a very fast feature selection routine for classification tasks
  - features must be non-negative
  - works on sparse matrices
  - works on multi-class problems

Feature selection using univariate comparisions

- χ^2 is a very fast feature selection routine for classification tasks
  - features must be non-negative
  - works on sparse matrices
  - works on multi-class problems
- With negative predictors:
  - use `f_classif`.

Feature selection using univariate comparisions

- χ^2 is a very fast feature selection routine for classification tasks
  - features must be non-negative
  - works on sparse matrices
  - works on multi-class problems
- With negative predictors:
  - use `f_classif`.
- For regression tasks:
  - use `f_regression or OLS coefficients`.

De-Meaning or Residualizing Variables

- For f_classif and f_regression, can de-mean predictors by groups, for example by year or location.
  - for regression, can also de-mean the outcome.

De-Meaning or Residualizing Variables

- For f_classif and f_regression, can de-mean predictors by groups, for example by year or location.
  - for regression, can also de-mean the outcome.
- What if you want to de-mean by both year and location?
  - $\rightarrow$ take residuals from OLS regression of each variable (outcome and predictor) on the category dummies.

De-Meaning or Residualizing Variables

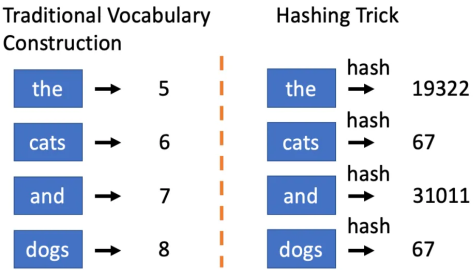
- For f_classif and f_regression, can de-mean predictors by groups, for example by year or location.
  - for regression, can also de-mean the outcome.
- What if you want to de-mean by both year and location?
  - $\rightarrow$ take residuals from OLS regression of each variable (outcome and predictor) on the category dummies.
- That is:
  - regress $Y_i = \text{FE}_1 + \text{FE}_2 + \epsilon_i$ and $x_i^w = \text{FE}_1 + \text{FE}_2 + \epsilon_i, \forall w$,
  - take residuals $\tilde{Y}_i = Y_i - \hat{Y}_i$ and $\tilde{x}_i^w = x_i^w - \hat{x}_i^w$

De-Meaning or Residualizing Variables

- For f_classif and f_regression, can de-mean predictors by groups, for example by year or location.
  - for regression, can also de-mean the outcome.
- What if you want to de-mean by both year and location?
  - $\rightarrow$ take residuals from OLS regression of each variable (outcome and predictor) on the category dummies.
- That is:
  - regress $Y_i = \text{FE}_1 + \text{FE}_2 + \epsilon_i$ and $x_i^w = \text{FE}_1 + \text{FE}_2 + \epsilon_i, \forall w$,
  - take residuals $\tilde{Y}_i = Y_i - \hat{Y}_i$ and $\tilde{x}_i^w = x_i^w - \hat{x}_i^w$
- Then use residuals as variables, in feature selection step or in machine learning task.

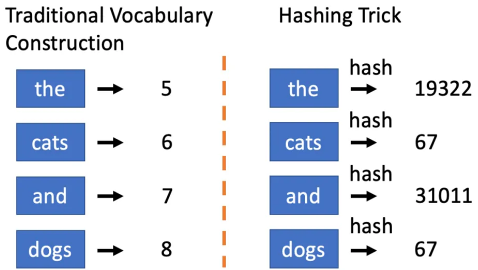
Hashing Vectorizer

- A very different approach to tokenizing documents:
  - rather than make a one-to-one lookup for each n-gram, put n-grams through a hashing function that takes an arbitrary string and outputs an integer in some range (e.g. 1 to 10,000).



Hashing Vectorizer

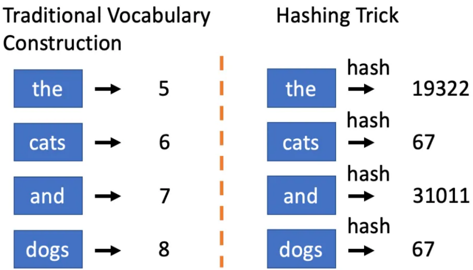
- A very different approach to tokenizing documents:
  - rather than make a one-to-one lookup for each n-gram, put n-grams through a hashing function that takes an arbitrary string and outputs an integer in some range (e.g. 1 to 10,000).



- Pros:
  - can have arbitrarilly small feature space
  - handles out-of-vocabulary words – any word or n-gram gets assigned to an arbitrary integer based on the hash function.

Hashing Vectorizer

- A very different approach to tokenizing documents:
  - rather than make a one-to-one lookup for each n-gram, put n-grams through a hashing function that takes an arbitrary string and outputs an integer in some range (e.g. 1 to 10,000).



- Pros:
  - can have arbitrarilly small feature space
  - handles out-of-vocabulary words – any word or n-gram gets assigned to an arbitrary integer based on the hash function.
- Cons:
  - cannot interpret features
    - at least not directly – could in principle keep track of the mapping
  - will have collisions – n-grams will randomly be paired with each other in the feature map.
    - usually innocuous, but could sum outputs of two hashing functions to minimize this.

Collocations are Familiar N-grams

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")

Collocations are Familiar N-grams

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")
  - Non-substitutable: cannot substitute components with synonyms ("fast food"≠"quick food")

Collocations are Familiar N-grams

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")
  - Non-substitutable: cannot substitute components with synonyms ("fast food"≠"quick food")
  - Non-modifiable: cannot modify with additional words or grammar: (e.g., "kick around the bucket", "kick the buckets")

Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\mathrm{PMI}(w_1, w_2) = \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.

Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\mathrm{PMI}(w_1, w_2) = \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.

  - ranks words by how often they collocate, relative to how often they occur apart.

Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\text{PMI}(w_1, w_2) = \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

  where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.
  - ranks words by how often they collocate, relative to how often they occur apart.
- Warning: Rare words that appear together once or twice will have high PMI.

Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\text{PMI}(w_1, w_2) = \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

  where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.
  - ranks words by how often they collocate, relative to how often they occur apart.
- Warning: Rare words that appear together once or twice will have high PMI.
  - Address this with minimum frequency thresholds.

Geometric Mean: Normalized PMI for $N \geq 2$

- PMI can be generalized to arbitrary N as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, \ldots, w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

Geometric Mean: Normalized PMI for $N \geq 2$

- PMI can be generalized to arbitrary N as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, \ldots, w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

- E.g., for trigrams:

$$\frac{\Pr(w_1, w_2, w_3)}{\sqrt[3]{\Pr(w_1)\Pr(w_2)\Pr(w_3)}}$$

Geometric Mean: Normalized PMI for $N \geq 2$

- PMI can be generalized to arbitrary N as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, \ldots, w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

- E.g., for trigrams:

$$\frac{\Pr(w_1, w_2, w_3)}{\sqrt[3]{\Pr(w_1)\Pr(w_2)\Pr(w_3)}}$$

- The n-root normalizer is not necessary (it does not change the ranking), but makes scores for bigrams/trigrams/quadgrams/etc. more comparable.

Phrase Dictionaries

syntax	phrase1	phrase2	entailment
[VP/NNP]	proposed by the president of the	proposed by the chairman of the	Equivalence
[VP/NNP]	proposed by the chairman of the	proposed by the president of the	Equivalence
[VP]	referred to in this report	referred to in the present report	Equivalence
[VP/NNP]	addressed to the president of the	addressed to the chairman of the	ReverseEntailment
[VP/NNP]	addressed to the chairman of the	addressed to the president of the	ForwardEntailment
[SQ/.]	are you all right , sir	is everything all right , sir	Equivalence
[VP/NNP]	submitted by the president of the	submitted by the chairman of the	ForwardEntailment
[VP/NNP]	submitted by the chairman of the	submitted by the president of the	ReverseEntailment
[PP]	in various parts of the world	in different parts of the world	Equivalence
[SQ/.]	is everything all right , sir	are you all right , sir	Equivalence
[VP]	described in this report	described in the present report	Equivalence
[X]	purposes of this agreement ,	purposes of the present agreement ,	Equivalence
[VP]	contained in this report	contained in the present report	Equivalence
[VP]	proposed in this report	proposed in the present report	Equivalence
[VP/NN]	voted in favour of the draft	voted in favour of the	Equivalence

- The Paraphrase Database 2.0 (PPDB, `paraphrase.org/#/download`) has a large database of equivalent/related words/phrases.
  - could be used to make a vocabulary, or for dimension reduction.

Domain dictionaries

- Could take wikipedia article names as lists of multi-word expressions.
  - in law, could use legal dictionaries (e.g., “first amendment”, “beyond a reasonable doubt”).

Named Entity Recognition

- refers to the task of identifying named entities such as "ETH Zurich" and "Marie Curie".

  [PER John Smith] , president of [ORG McCormik Industries] visited his niece [PER Paris] in [LOC Milan], reporters say .

  BIO tags for named entity recognition

Tag	Meaning
O	Not part of a named entity
B-PER	First word of a person name
I-PER	Continuation of a person name
B-LOC	First word of a location name
I-LOC	Continuation of a location name
B-ORG	First word of an organization name
I-ORG	Continuation of an organization name
B-MISC	First word of another kind of named entity
I-MISC	Continuation of another kind of named entity

- can tokenize named entities.

Sub-Word Units

- Advanced NLP tasks (text generation, question answering) benefit from encoding sub-word information.
- Tokenizers like **SentencePiece** do tokenizing at the character level, with white space and punctuation treated equivalently to alphanumeric characters.
  - requires lots of data/compute but learns word endings etc

Outline

Parts of speech tags

- Parts of speech (POS) tags provide useful word categories corresponding to their functions in sentences:
  - Eight main parts of speech: verb (VB), noun (NN), pronoun (PR), adjective (JJ), adverb (RB), determinant (DT), preposition (IN), conjunction (CC).
  - The Penn TreeBank POS tag set (used in many applications) has 36 tags: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

Parts of speech tags

- Parts of speech (POS) tags provide useful word categories corresponding to their functions in sentences:
  - Eight main parts of speech: verb (VB), noun (NN), pronoun (PR), adjective (JJ), adverb (RB), determinant (DT), preposition (IN), conjunction (CC).
  - The Penn TreeBank POS tag set (used in many applications) has 36 tags: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`
- Parts of speech vary in their informativeness for various functions:
  - For categorizing topics, nouns are usually most important
  - For sentiment, adjectives are usually most important.

Parts of speech as features

- Can produce n-grams from parts of speech tags:
  - counts over NV, VN, AN, etc.

Parts of speech as features

- Can produce n-grams from parts of speech tags:
  - counts over NV, VN, AN, etc.
- POS n-gam frequencies are good stylistic features for authorship detection.
  - not biased by topics/content
  - for function words, can use the word itself rather than the POS tag.

Constructing "Memes" with POS

- A: Adjective, N: Noun, V: Verb, P: Preposition, D: Determinant, C: Conjunction.
- 2-grams: AN, NN, VN, VV, NV, VP.
  - tax credit, magistrate judge
- 3-grams: NNN, AAN, ANN, NAN, NPN, VAN, VNN, AVN, VVN, VPN, ANV, NVV, VDN, VVV, NNV, VVP, VAV, VVN, NCN, VCV, ACA, PAN.
  - armed and dangerous, stating the obvious

Constructing "Memes" with POS

- A: Adjective, N: Noun, V: Verb, P: Preposition, D: Determinant, C: Conjunction.
- 2-grams: AN, NN, VN, VV, NV, VP.
  - tax credit, magistrate judge
- 3-grams: NNN, AAN, ANN, NAN, NPN, VAN, VNN, AVN, VVN, VPN, ANV, NVV, VDN, VVV, NNV, VVP, VAV, VVN, NCN, VCV, ACA, PAN.
  - armed and dangerous, stating the obvious
- 4-grams: NCVN, ANNN, NNNN, NPNN, AANN, ANNN, ANPN, NNPN, NPAN, ACAN, NCNN, NNCN, ANCN, NCAN, PDAN, PNPN, VDNN, VDAN, VVDN.
  - `Beyond a reasonable doubt (preposition, article, adjective, noun)`
  - `Earned income tax credit (adjective, noun, noun, noun)`

How to set statistical thresholds

- Potentially complex thresholds for vocabulary inclusion based on frequency, parts of speech, and point-wise mutual information.
  - could use domain dictionaries as a source for tuning these statistics.

Outline

Ranking Partisan language:
Monroe et al (2009)

- This paper systematically explores a number of methods for identifying words that are distinctive of groups of speakers
  - in this case, whether U.S. congressmen are Republicans are Democrats.

Ranking Partisan language:

Monroe et al (2009)

- This paper systematically explores a number of methods for identifying words that are distinctive of groups of speakers
  - in this case, whether U.S. congressmen are Republicans are Democrats.
- First, they separate speeches by topic using latent dirichlet allocation (next lecture).
  - they then test a number of methods for ranking partisanship of words.

Relative Frequency of Words

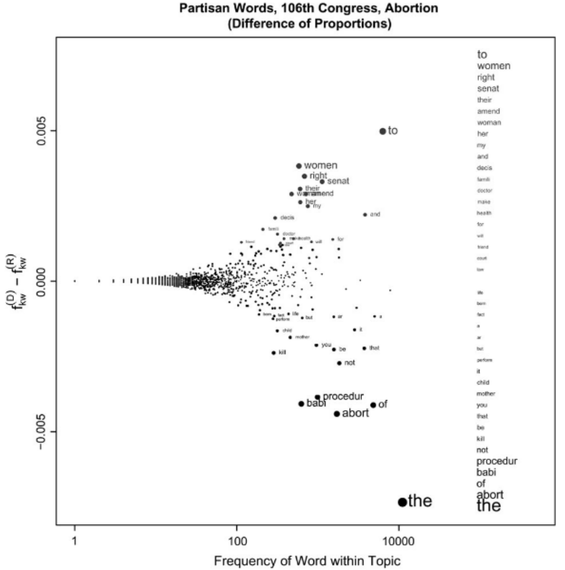


Fig. 1 Feature evaluation and selection using $f_{kw}^{(D)} - f_{kw}^{(R)}$. Plot size is proportional to evaluation weight, $|f_{kw}^{(D)} - f_{kw}^{(R)}|$. The top 20 Democratic and Republican words are labeled and listed in rank order to the right. The results are almost identical for two other measures discussed in the text: unlogged *tf.idf* and frequency-weighted WordScores.

Log Odds Ratio Between Groups

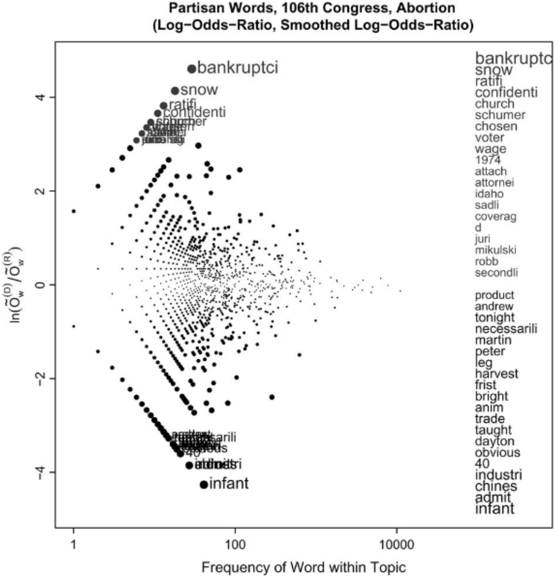


Fig. 2 Feature evaluation and selection using $\hat{\delta}_{kw}^{(D-R)}$. Plot size is proportional to evaluation weight, $\left|\hat{\delta}_{kw}^{(D-R)}\right|$. Top 20 Democratic and Republican words are labeled and listed in rank order. The results are identical to another measure discussed in the text: the log-odds-ratio with uninformative Dirichlet prior.

Bayesian Multinomial Model

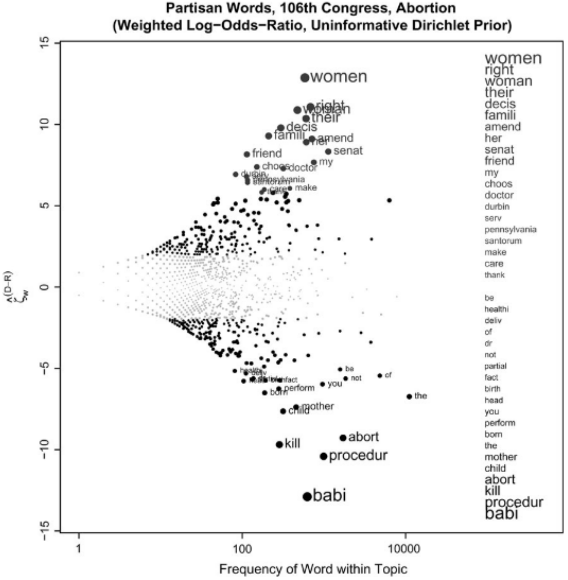


Fig. 4 Feature evaluation and selection using $\hat{\zeta}_{kw}^{(D-R)}$. Plot size is proportional to evaluation weight, $\left|\hat{\zeta}_{kw}^{(D-R)}\right|$; those with $\left|\hat{\zeta}_{kw}^{(D-R)}\right| < 1.96$ are gray. The top 20 Democratic and Republican words are labeled and listed in rank order to the right.

Bayesian Multinomial Model, LaPlace Prior

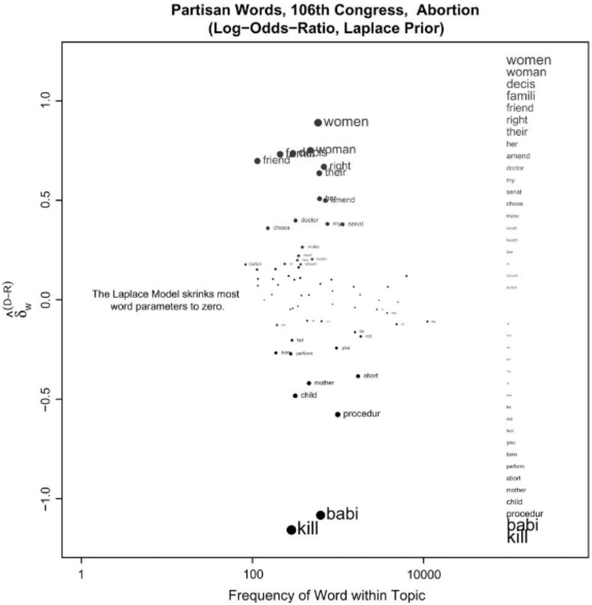


Fig. 6 Feature evaluation and selection using $\hat{\delta}_{kw}^{(D-R)}$. Plot size is proportional to evaluation weight, $\hat{\delta}_{kw}^{(D-R)}$. The top 20 Democratic and Republican words are labeled and listed in rank order to the right.

Gentzkow and Shapiro (2010)

Gentzkow and Shapiro (2010)

- Corpora:
  - news text from large sample of US daily newspapers.
  - congressional text is 2005 Congressional Record.

Gentzkow and Shapiro (2010)

- Corpora:
  - news text from large sample of US daily newspapers.
  - congressional text is 2005 Congressional Record.
- Pre-process text, stripping away prepositions, conjunctions, pronouns, and common words
  - get bigrams and trigrams

Gentzkow and Shapiro (2010)

- Corpora:
  - news text from large sample of US daily newspapers.
  - congressional text is 2005 Congressional Record.
- Pre-process text, stripping away prepositions, conjunctions, pronouns, and common words
  - get bigrams and trigrams
- Identify polarizing phrases using χ^2 metric. For each phrase w, let D_w be frequency for Democrats, R_w be frequency for Republicans. Let D_w^- and R_w^- be frequencies of *other* phrases.
- Then:

$$\chi^2_w = \frac{(R_w D_w^- - D_w R_w^-)^2}{(D_w + R_w)(D_w + D_w^-)(R_w + R_w^-)(D_w^- + R_w^-)}$$

  - this is the test statistic for equality between parties of phrase use if they were both drawn from multinomial distributions.
  - in sklearn, it is feature_selection.chi2

TABLE I

MOST PARTISAN PHRASES FROM THE 2005 *CONGRESSIONAL RECORD*[a]

Panel A: Phrases Used More Often by Democrats

Two-Word Phrases		
private accounts	Rosa Parks	workers rights
trade agreement	President budget	poor people
American people	Republican party	Republican leader
tax breaks	change the rules	Arctic refuge
trade deficit	minimum wage	cut funding
oil companies	budget deficit	American workers
credit card	Republican senators	living in poverty
nuclear option	privatization plan	Senate Republicans
war in Iraq	wildlife refuge	fuel efficiency
middle class	card companies	national wildlife
Three-Word Phrases		
veterans health care	corporation for public broadcasting	cut health care
congressional black caucus	additional tax cuts	civil rights movement
VA health care	pay for tax cuts	cuts to child support
billion in tax cuts	tax cuts for people	drilling in the Arctic National
credit card companies	oil and gas companies	victims of gun violence
security trust fund	prescription drug bill	solvency of social security
social security trust	caliber sniper rifles	Voting Rights Act
privatize social security	increase in the minimum wage	war in Iraq and Afghanistan
American free trade	system of checks and balances	civil rights protections
central American free	middle class families	credit card debt

TABLE I—*Continued*

Panel B: Phrases Used More Often by Republicans

Two-Word Phrases		
stem cell	personal accounts	retirement accounts
natural gas	Saddam Hussein	government spending
death tax	pass the bill	national forest
illegal aliens	private property	minority leader
class action	border security	urge support
war on terror	President announces	cell lines
embryonic stem	human life	cord blood
tax relief	Chief Justice	action lawsuits
illegal immigration	human embryos	economic growth
date the time	increase taxes	food program
Three-Word Phrases		
embryonic stem cell	Circuit Court of Appeals	Tongass national forest
hate crimes legislation	death tax repeal	pluripotent stem cells
adult stem cells	housing and urban affairs	Supreme Court of Texas
oil for food program	million jobs created	Justice Priscilla Owen
personal retirement accounts	national flood insurance	Justice Janice Rogers
energy and natural resources	oil for food scandal	American Bar Association
global war on terror	private property rights	growth and job creation
hate crimes law	temporary worker program	natural gas natural
change hearts and minds	class action reform	Grand Ole Opry
global war on terrorism	Chief Justice Rehnquist	reform social security

[a]The top 60 Democratic and Republican phrases, respectively, are shown ranked by χ^2_{pl}. The phrases are classified as two or three word after dropping common "stopwords" such as "for" and "the." See Section 3 for details and see Appendix B (online) for a more extensive phrase list.

Consumers drive media slant (GS 2010)

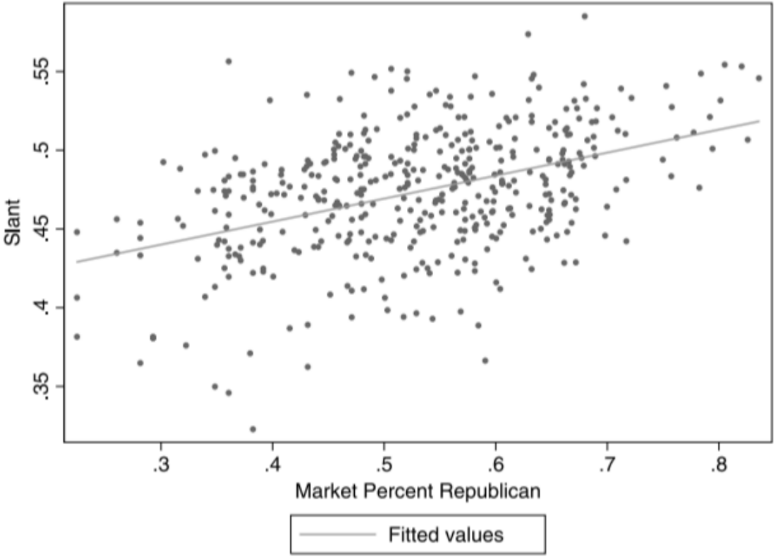


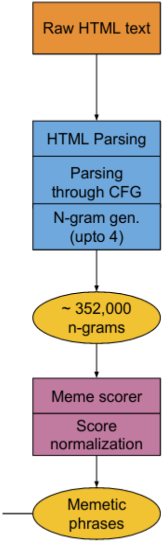
FIGURE 4.—Newspaper slant and consumer ideology. The newspaper slant index against Bush's share of the two-party vote in 2004 in the newspaper's market is shown.

Detecting Memes with Citation Networks

Kuhn, Perc, and Helbing (2014); Chen, Parthasaratha, and Verma (2017)

Detecting Memes with Citation Networks

Kuhn, Perc, and Helbing (2014); Chen, Parthasaratha, and Verma (2017)



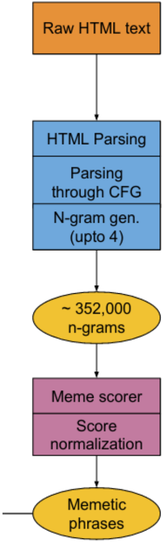
- **meme score** for phrase m from Kuhn et al (2014) is

$$M_m = f_m P_m$$

- frequency of m, $f_m = \frac{\#\text{ cases mentioning } m}{\#\text{ total cases}}$

Detecting Memes with Citation Networks

Kuhn, Perc, and Helbing (2014); Chen, Parthasaratha, and Verma (2017)



- **meme score** for phrase m from Kuhn et al (2014) is

$$M_m = f_m P_m$$

- frequency of m, $f_m = \frac{\#\text{ cases mentioning } m}{\#\text{ total cases}}$
- Propagation score for m:

$$P_m = \underbrace{\frac{d_{m\to m}}{d_{\to m}}}_{\text{"sticking"}} \bigg/ \underbrace{\frac{d_{m\to m'}}{d_{\to m'}}}_{\text{"sparking"}}.$$

  - "sticking factor":
    - $d_{m\to m}$, # cases with m that cite a case with m, divided by
    - $d_{\to m}$, # cases without m that cite a case with m
  - "sparking factor":
    - $d_{m\to m'}$, # cases with m that do not cite a case with m, divided by
    - $d_{\to m'}$, # cases without m that do not cite a case with m

Extracted Memes

Kuhn, Perc, and Helbing (2014)

1. loop quantum cosmology^{+*}
2. unparticle^{+*}
3. sonoluminescence^{+*}
4. $MgB_2$$^{+}$
5. stochastic resonance^{+*}
6. carbon nanotubes^{+*}
7. $NbSe_3$$^{+}$
8. black hole^{+*}
9. nanotubes^{+}
10. lattice Boltzmann^{+*}
11. dark energy^{+*}
12. Rashba
13. $CuGeO_3$$^{+}$
14. strange nonchaotic
15. in $NbSe_3$
16. spin Hall^{+}
17. elliptic flow^{+*}
18. quantum Hall^{+*}
19. $CeCoIn_5$$^{+}$
20. inflation^{+}
21. exchange bias^{+*}
22. $Sr_2RuO_4$$^{+}$
23. traffic flow^{+*}
24. TiOCl
25. key distribution^{+}
26. graphene^{+*}

Chen, Parthasaratha, and Verma (2017)

Phrase	**Normalized Meme Score**
red heat	0.138
salvage services	0.0039
said cars	0.0029
Atlantic coast	0.00216
citizens of different states	0.00212
insurance effected	0.0020
separable controversy	0.0018
taken in tow	0.0017
schooner was	0.00126
fourteenth amendment	0.00125
contract of affreightment	0.00119
patented design	0.0011
constitution or laws	0.0009
mere transient or sojourner	0.0008

Loan Application Words Predicting Repayment (Netzer, Lemaire, and Herzenstein 2019)

Loan Application Words Predicting Repayment (Netzer, Lemaire, and Herzenstein 2019)

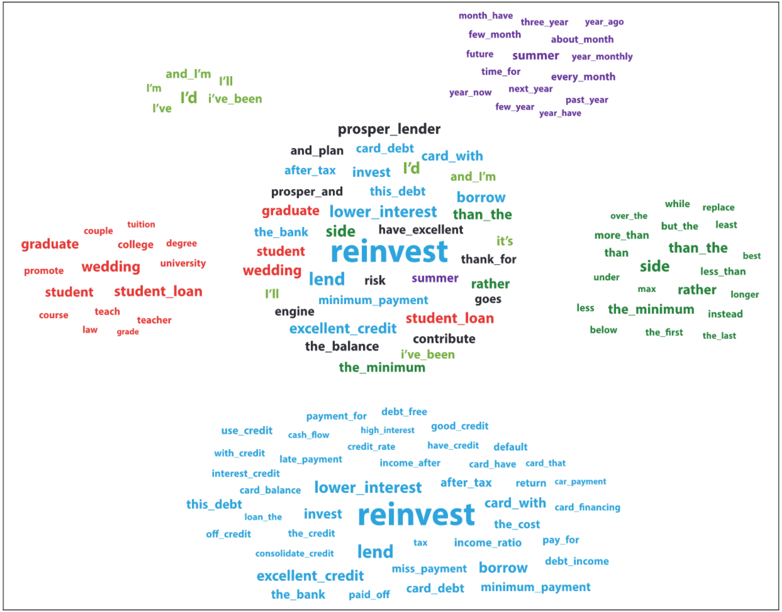


Figure 2. Words indicative of loan repayment.

Notes: The most common words appear in the middle cloud (cutoff = 1:1.5) and are then organized by themes. Starting on the right and moving clockwise: relative words, financial literacy words, words related to a brighter financial future, "I" words, and time-related words.

Loan Application Words Predicting Default (Netzer, Lemaire, and Herzenstein 2019)

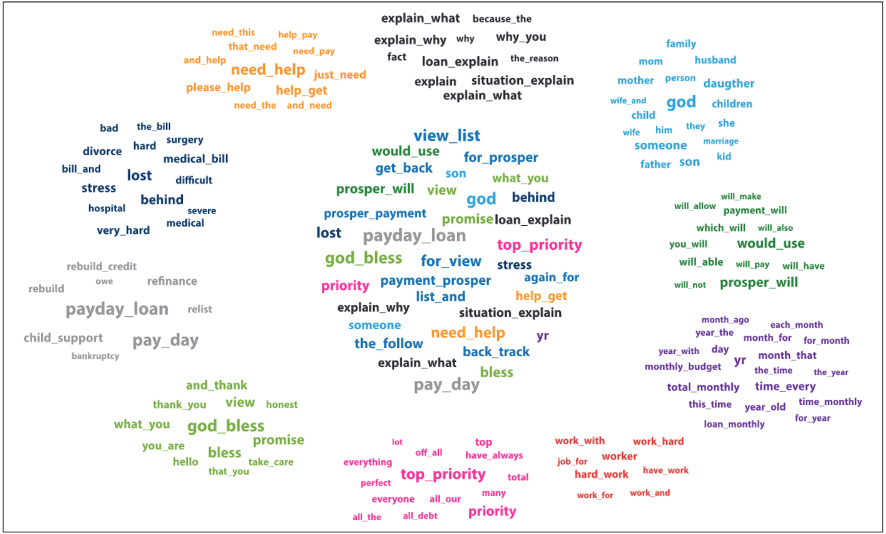


Figure 3. Words indicative of loan default.

Notes: The most common words appear in the middle cloud (cutoff = 1:1.5) and are then organized by themes. Starting on the top and moving clockwise: words related to explanations, external influence words and others, future-tense words, time-related words, work-related words, extremity words, words appealing to lenders, words relating to financial hardship, words relating to general hardship, and desperation/plea words.

Outline

Computing Geometric Mean with N-gram Counts

- Probability of a token is the frequency in the corpus:

$$\Pr(w_1) = \frac{\text{Count}(w_1)}{\sum_{i=1}^{P} \text{Count}(w_i)}$$

where P is vocabulary size.

Computing Geometric Mean with N-gram Counts

- Probability of a token is the frequency in the corpus:

$$\Pr(w_1) = \frac{\text{Count}(w_1)}{\sum_{i=1}^{P} \text{Count}(w_i)}$$

where P is vocabulary size.

- Let $f_i = \text{Count}(w_i)$ and $F = \sum_{i=1}^{P} f_i$. Then we have

$$\text{PMI}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\Pr(w_1)\Pr(w_2)} = \frac{\frac{f_{12}}{F}}{\frac{f_1}{F} \cdot \frac{f_2}{F}} = \frac{1}{F}\frac{f_{12}}{f_1 f_2}$$

Computing Geometric Mean with N-gram Counts

- Probability of a token is the frequency in the corpus:

$$\Pr(w_1) = \frac{\text{Count}(w_1)}{\sum_{i=1}^{P} \text{Count}(w_i)}$$

  where P is vocabulary size.

- Let $f_i = \text{Count}(w_i)$ and $F = \sum_{i=1}^{P} f_i$. Then we have

$$\text{PMI}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\Pr(w_1)\Pr(w_2)} = \frac{\frac{f_{12}}{F}}{\frac{f_1}{F} \cdot \frac{f_2}{F}} = \frac{1}{F}\frac{f_{12}}{f_1 f_2}$$

- Note that the leading $\frac{1}{F}$ does not affect the ranking of bigrams, and cancels out with the geometric mean formula:

$$\text{gmean}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\sqrt{\Pr(w_1)\Pr(w_2)}} = \frac{\frac{f_{12}}{F}}{\sqrt{\frac{f_1}{F} \cdot \frac{f_2}{F}}} = \frac{f_{12}}{\sqrt{f_1 f_2}}$$

- Similarly, it cancels out for $N > 2$.
- Therefore PMI can be computed directly from term counts (rather than frequencies).

GST: Generative Model of Text

Gentzkow, Shapiro, and Taddy (*Econometrica* 2019)

GST: Generative Model of Text

Gentzkow, Shapiro, and Taddy (*Econometrica* 2019)

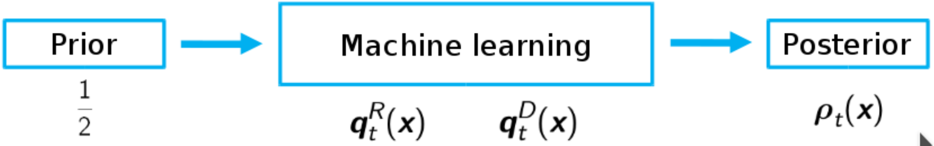
$\boldsymbol{c}_{it}^{p}$, vector of phrase frequences for speaker i at year t, by party $p \in D, R$, drawn from multinomial distribution

$$\boldsymbol{c}_{it}^{D} \sim \mathrm{MN}(\boldsymbol{q}_t^{D})$$
$$\boldsymbol{c}_{it}^{R} \sim \mathrm{MN}(\boldsymbol{q}_t^{R})$$

- $\boldsymbol{q}_t^{D}$ and $\boldsymbol{q}_t^{R}$ are party-specific vectors of probabilities

Bayesian Learning of Partisanship

Gentzkow, Shapiro, and Taddy (2019)

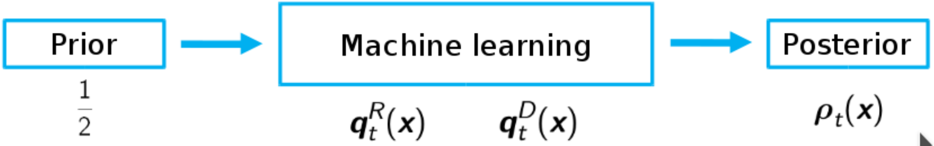


$\rho_{jt} = \frac{q_{jt}^R}{q_{jt}^R + q_{jt}^D}$, **posterior probability** that observer with neutral prior assigns to speaker being Republican if see phrase j in year t

- $\boldsymbol{\rho}_t$ is the vector of posteriors associated with each phrase

Bayesian Learning of Partisanship

Gentzkow, Shapiro, and Taddy (2019)



$\rho_{jt} = \frac{q_{jt}^R}{q_{jt}^R + q_{jt}^D}$, **posterior probability** that observer with neutral prior assigns to speaker being Republican if see phrase j in year t

- $\boldsymbol{\rho}_t$ is the vector of posteriors associated with each phrase

Define π_t = **partisanship** at time t:

$$\pi_t = \frac{1}{2}\boldsymbol{q}_t^R \cdot \boldsymbol{\rho}_t + \frac{1}{2}\boldsymbol{q}_t^D \cdot (1 - \boldsymbol{\rho}_t)$$

- Weighted average of posteriors – that is, is text informative of affiliation?

Language Choice Model

Gentzkow, Shapiro, and Taddy (2019)

- Let speaker i's "utility" from speaking phrase j at time t be

$$u_{ijt} = \alpha_{jt} + \boldsymbol{x}'_{it}\gamma_t + R_i\varphi_j$$

  - α_{jt}, baseline utility
  - γ_t, utility associated to speaker characteristics
  - R_i=Republican, so φ_j indexes party difference.

Language Choice Model

Gentzkow, Shapiro, and Taddy (2019)

- Let speaker i's "utility" from speaking phrase j at time t be

$$u_{ijt} = \alpha_{jt} + \boldsymbol{x}'_{it}\gamma_t + R_i\varphi_j$$

  - α_{jt}, baseline utility
  - γ_t, utility associated to speaker characteristics
  - R_i=Republican, so φ_j indexes party difference.

- If speaker chooses phrases to maximize utility u_{it} with respect to a choice-specific i.i.d. type 1 extreme value shock, then

$$q_{jt}(\boldsymbol{x}_{it}) = \frac{\exp(u_{ijt})}{\sum_l \exp(u_{ilt})}$$

Regularized cost function

Gentzkow, Shapiro, and Taddy (2019)

Learn $(\alpha_{jt}, \gamma_t, \varphi_j)$ to minimize

$$\sum_j \left\{ \sum_t \sum_i [\exp(\alpha_{jt} + \boldsymbol{x}'_{it}\gamma_t + R_i\varphi_j)m_{it} - (\alpha_{jt} + \boldsymbol{x}'_{it}\gamma_t + R_i\varphi_j)c_{ijt} + \lambda_j|\varphi_j|] \right\}$$

where m_{it}= number of phrases spoken; c_{ijt}= count for phrase j

- Approximate multinomial with Poisson model

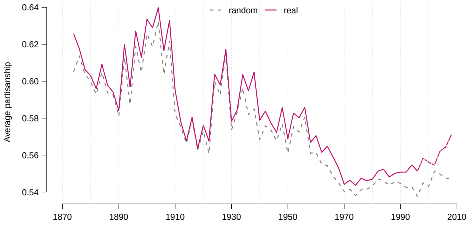
  $$c_{ijt} \sim \text{Pois}(\exp[\log(m_{it}) + u_{ijt}])$$

  allowing parallel computation across phrases.
- λ_j=phrase-specific lasso penalty, chosen to maximize information criterion.

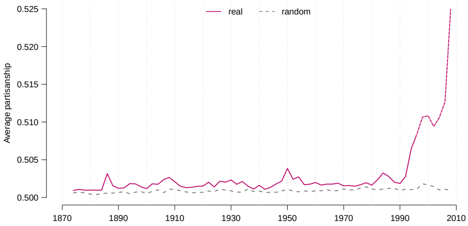
Regularization and Permutation

Gentzkow, Shapiro, and Taddy (2019)

Maximum Likelihood Estimator	**Preferred (Penalized) Estimator**



Usual method: Plug-in MLE w/ Congress speech



Regularized method w/ permutation inference

Figure 3: Informativeness of Speech by Speech Length and Session

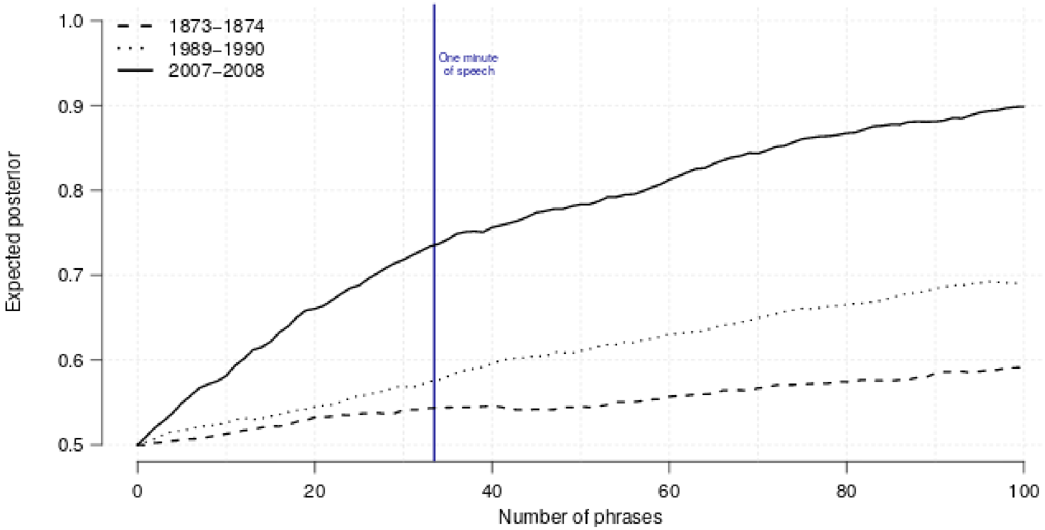


Figure 8: Partisanship vs. Roll-Call Voting

Panel A: Over Time