Big Data for Public Policy

4. Introduction to Text Data

Elliott Ash & Malka Guillot

Outline

Text as Data

- Text data is a sequence of characters called **documents**.
- The set of documents is the **corpus**.

Text as Data

- Text data is a sequence of characters called **documents**.
- The set of documents is the **corpus**.

- Text data is **unstructured**:
  - the information we want is mixed together with (lots of) information we don't.
  - How to separate the two?

Diversification of Text Data Methods

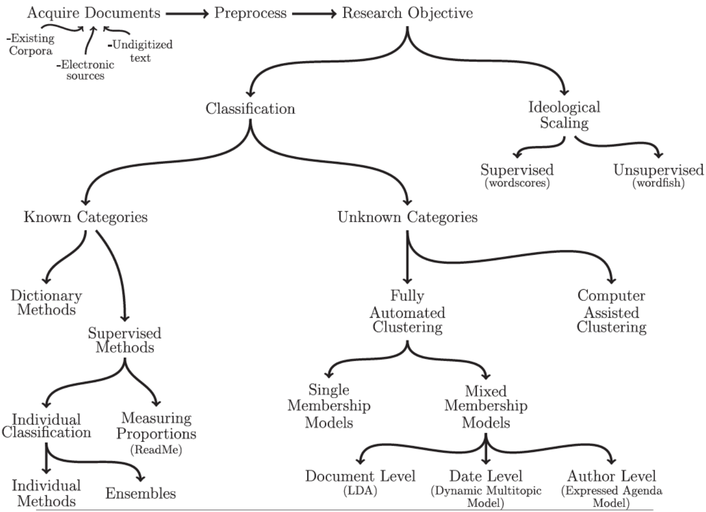


Fig. 1 An overview of text as data methods.

Source: Stewart and Grimmer (2013).

Outline

Corpus cleaning

- Pre-Processing Steps:
  - Remove HTML markup, extra white space, and unicode

Corpus cleaning

- Pre-Processing Steps:
  - Remove HTML markup, extra white space, and unicode
- But HTML markup is often valuable:
  - HTML markup for section header names.
  - Legal database web sites often have HTML tags for citations to other cases.

Corpus cleaning

- Pre-Processing Steps:
  - Remove HTML markup, extra white space, and unicode
- But HTML markup is often valuable:
  - HTML markup for section header names.
  - Legal database web sites often have HTML tags for citations to other cases.
- Other cleaning steps:
  - page numbers
  - hyphenations at line breaks
  - table of contents, indexes, etc.
- These are all corpus-specific, so inspect ahead of time.

OCR (Optical Character Recognition)

- Your data might be in PDF's or images. Needs to be converted to text
- The best solution (that I know of) is ABBYY FineReader, which is expensive but might be available at your university library.
- My colleague Joe Sutherland at Columbia has a nice open-source package for OCR:
  - `https://github.com/jlsutherland/doc2text`

Other Languages

- All of the tools that we discuss in this class are available in many languages.
- spaCy has full functionality in English, German, Spanish, Portuguese, French, Italian, and Dutch.
  - beta functionality in dozens of other languages including Chinese and Arabic
  - See `https://spacy.io/usage/models`.
- Can also translate (e.g., API links to google translate and DeepL).
- The machine learning models are language-independent.

Outline

Dictionary Methods

- Dictionary methods use a pre-selected list of words or phrases to analyze a corpus.
- Corpus-specific
  - count words related to your analysis

Dictionary Methods

- Dictionary methods use a pre-selected list of words or phrases to analyze a corpus.
- Corpus-specific
  - count words related to your analysis
- General
  - e.g. LIWC (`liwc.wpengine.com`) has lists of words across categories.

Dictionary Methods

- Dictionary methods use a pre-selected list of words or phrases to analyze a corpus.
- Corpus-specific
  - count words related to your analysis
- General
  - e.g. LIWC (`liwc.wpengine.com`) has lists of words across categories.
  - Sentiment Analysis: count sets of positive and negative words (doesn't work very well)

Regular Expressions

- Regular Expressions, implemented in the Python package **re**, provide a powerful string matching tool.
  - A systematic string matching protocal – can match arbitrary string patterns
  - e.g., use utilit* to match utility, utilities, utilitarian, ...
  - Important for identifying speaker names (in political documents) section headers (in statutes), citations (in judicial opinions), etc.

Regular Expressions

- Regular Expressions, implemented in the Python package **re**, provide a powerful string matching tool.
  - A systematic string matching protocal – can match arbitrary string patterns
  - e.g., use utilit* to match utility, utilities, utilitarian, ...
  - Important for identifying speaker names (in political documents) section headers (in statutes), citations (in judicial opinions), etc.
- Also quite tedious, so we will not cover it here.
  - See NLTK book Chapter 3.4-3.5 for an introduction.

Sentiment Analysis in Python

- The vader class in nltk provides positive, negative, and neutral scores for a document, and a composite score that combines all three.

Sentiment Analysis in Python

- The vader class in nltk provides positive, negative, and neutral scores for a document, and a composite score that combines all three.
  - vader works best on raw text – capitalization and punctuation are used in the calculus.

Sentiment Analysis in Python

- The vader class in nltk provides positive, negative, and neutral scores for a document, and a composite score that combines all three.
  - vader works best on raw text – capitalization and punctuation are used in the calculus.
- Designed for online writing – hard to say how well it works on legal text, for example.
  - Hamilton-Clark-Leskovec-Jurafsky (2016) provide a method for making domain-specific sentiment lexicons using word embeddings (more on this later).

Limitations of sentiment analysis

I'd hate to be the president

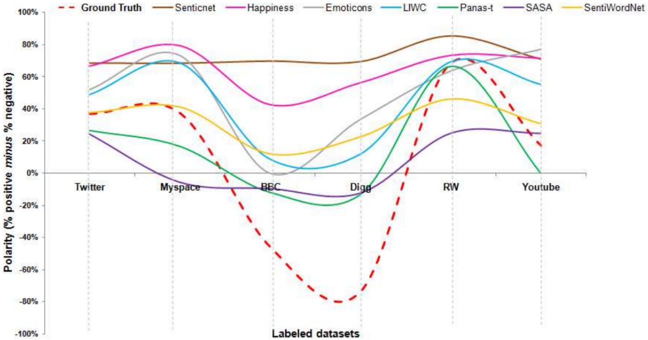


Figure 2: Polarity of the eight sentiment methods across the labeled datasets, indicating that existing methods vary widely in their agreement.

Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis

- Baker, Bloom, and Davis measure economic policy uncertainty using Boolean search of newspaper articles. (See `http://www.policyuncertainty.com/`).

Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis

- Baker, Bloom, and Davis measure economic policy uncertainty using Boolean search of newspaper articles. (See `http://www.policyuncertainty.com/`).
- For each paper on each day since 1985, submit the following query:
  - 1. Article contains “uncertain” OR “uncertainty”, AND
  - 2. Article contains “economic” OR “economy”, AND
  - 3. Article contains “congress” OR “deficit” OR “federal reserve” OR “legislation” OR “regulation” OR “white house”

Measuring uncertainty in macroeconomy

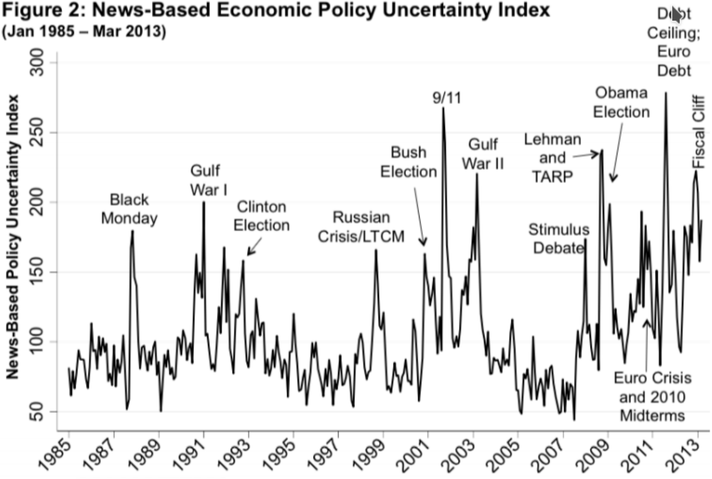
Baker, Bloom, and Davis

- Baker, Bloom, and Davis measure economic policy uncertainty using Boolean search of newspaper articles. (See `http://www.policyuncertainty.com/`).
- For each paper on each day since 1985, submit the following query:
  - 1. Article contains "uncertain" OR "uncertainty", AND
  - 2. Article contains "economic" OR "economy", AND
  - 3. Article contains "congress" OR "deficit" OR "federal reserve" OR "legislation" OR "regulation" OR "white house"
- Normalize resulting article counts by total newspaper articles that month.

Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis

Figure 2: News-Based Economic Policy Uncertainty Index
(Jan 1985 – Mar 2013)



LIWC

- LIWC (pronounced "Luke") stands for Linguistic Inquiry and Word Counts
  - 2300 words 70 lists of category-relevant words, e.g. "emotion", "cognition", "work", "family", "positive", "negative" etc.
  - Info and publications at `liwc.net`
  - Invented in 1980s, now in third version

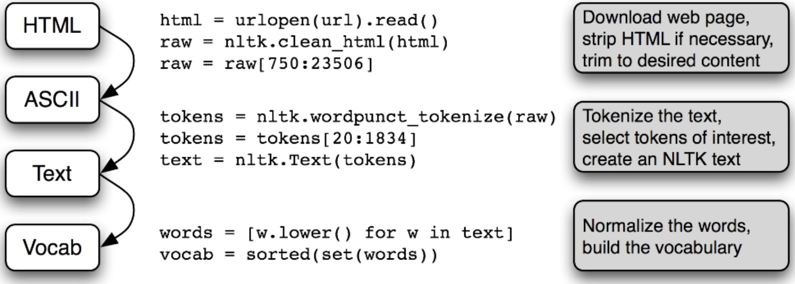
Emotion Lexicons

- 8 basic emotions, in four opposing pairs:
  - joy–sadness
  - anger–fear
  - trust–disgust
  - anticipation–surprise
- Mohammad and Turney (2011) code 10,000 words along the four dimensions (using Mturk)

Outline

Goals of Featurization

- The goal: produce features that are
  - **predictive** in the learning task
  - **interpretable** by human investigators
  - **tractable** enough to be easy to work with



Pre-processing

- Standard pre-processing steps:
  - drop capitalization, punctuation, numbers, stopwords (e.g. “the”, “such”)
  - remove word stems (e.g., “taxes” and “taxed” become “tax”)

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

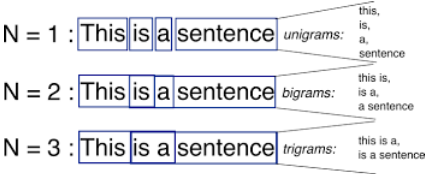
- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

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

- 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.

N-grams

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



- capture information and familiarity from local word order.
  - e.g. “estate tax” vs “death tax”

Filtering the Vocabulary

- N-grams will blow up your feature space: filtering out uninformative n-grams is necessary.
  - Google Developers recommend vocab size = $m =$20,000; I have gotten good performance from $m =$2,000.

Filtering the Vocabulary

- N-grams will blow up your feature space: filtering out uninformative n-grams is necessary.
  - Google Developers recommend vocab size $= m =$20,000; I have gotten good performance from $m =$2,000.

1. Drop phrases that appear in few documents, or in almost all documents, using tf-idf weights:

$$\text{tf-idf}(w) = (1 + \log(c_w)) \times \log(\frac{N}{d_w})$$

  - $c_w =$ count of phrase w in corpus, $N =$ number of documents, $d_w =$ number of documents where w appears.

Filtering the Vocabulary

- N-grams will blow up your feature space: filtering out uninformative n-grams is necessary.
  - Google Developers recommend vocab size = m =20,000; I have gotten good performance from m =2,000.

1. Drop phrases that appear in few documents, or in almost all documents, using tf-idf weights:

$$\text{tf-idf}(w) = (1 + \log(c_w)) \times \log(\frac{N}{d_w})$$

   - c_w= count of phrase w in corpus, N = number of documents, d_w = number of documents where w appears.

2. filter on parts of speech (keep nouns, adjectives, and verbs).

Filtering the Vocabulary

- N-grams will blow up your feature space: filtering out uninformative n-grams is necessary.
  - Google Developers recommend vocab size $= m =$20,000; I have gotten good performance from $m =$2,000.

1. Drop phrases that appear in few documents, or in almost all documents, using tf-idf weights:

$$\text{tf-idf}(w) = (1 + \log(c_w)) \times \log(\frac{N}{d_w})$$

   - c_w= count of phrase w in corpus, N = number of documents, d_w = number of documents where w appears.

2. filter on parts of speech (keep nouns, adjectives, and verbs).
3. filter on pointwise mutual information to get collocations (Ash JITE 2017, pg. 2)

Filtering the Vocabulary

- N-grams will blow up your feature space: filtering out uninformative n-grams is necessary.
  - Google Developers recommend vocab size = m =20,000; I have gotten good performance from m =2,000.

1. Drop phrases that appear in few documents, or in almost all documents, using tf-idf weights:

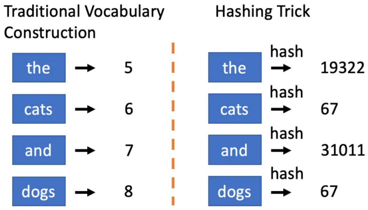
$$\text{tf-idf}(w) = (1 + \log(c_w)) \times \log(\frac{N}{d_w})$$

   - c_w= count of phrase w in corpus, N = number of documents, d_w = number of documents where w appears.

2. filter on parts of speech (keep nouns, adjectives, and verbs).
3. filter on pointwise mutual information to get collocations (Ash JITE 2017, pg. 2)
4. supervised feature selection: select phrases that are predictive of outcome.

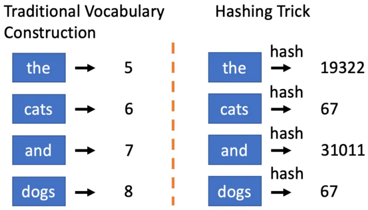
Hashing Vectorizer

- 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

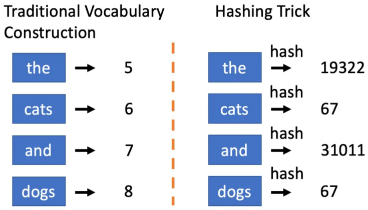
- A very different approach to tokenizing documents:
  - rather than make a one-to-one lookup for each n-gram, put n-grams through a hashing function that takes an arbitrary string and outputs an integer in some range (e.g. 1 to 10,000).



- 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.

Parts of speech

- Parts of speech (POS) tags provide useful word categories corresponding to their functions in sentences:
  - **Content**: noun (NN), verb (VB), adjective (JJ), adverb (RB)
  - **Function**: determinant (DT), preposition (IN), conjunction (CC), pronoun (PR).

Parts of speech

- Parts of speech (POS) tags provide useful word categories corresponding to their functions in sentences:
  - **Content**: noun (NN), verb (VB), adjective (JJ), adverb (RB)
  - **Function**: determinant (DT), preposition (IN), conjunction (CC), pronoun (PR).
- 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.

A decent baseline for featurization

A decent baseline for featurization

- Tag parts of speech: keep nouns, verbs, and adjectives.
- Drop stopwords, capitalization, punctuation.
- Run snowball stemmer to drop word endings.

A decent baseline for featurization

- Tag parts of speech: keep nouns, verbs, and adjectives.
- Drop stopwords, capitalization, punctuation.
- Run snowball stemmer to drop word endings.
- Make bigrams from the tokens.
- Take top 10,000 bigrams based on tf-idf weight.

A decent baseline for featurization

- Tag parts of speech: keep nouns, verbs, and adjectives.
- Drop stopwords, capitalization, punctuation.
- Run snowball stemmer to drop word endings.
- Make bigrams from the tokens.
- Take top 10,000 bigrams based on tf-idf weight.
- Represent documents as tf-idf frequencies over these bigrams.

Application: What Drives Media Slant?

Gentzkow and Shapiro (2010)

Application: What Drives Media Slant?

Gentzkow and Shapiro (2010)

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

Application: What Drives Media Slant?

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

Application: What Drives Media Slant?

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:

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

  - 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)

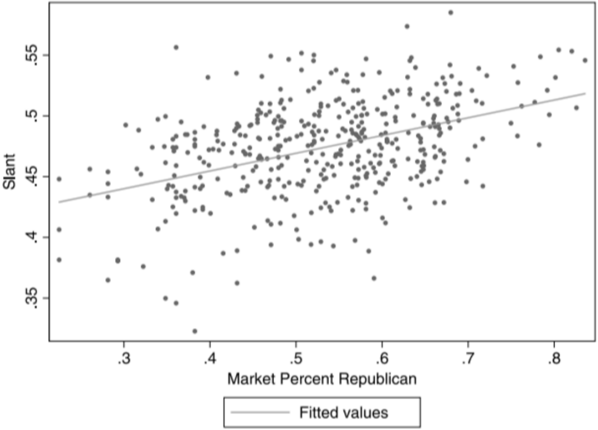
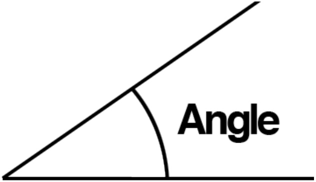


FIGURE 4.—Newspaper slant and consumer ideology. The newspaper slant index against Bush's share of the two-party vote in 2004 in the newspaper's market is shown.

Outline

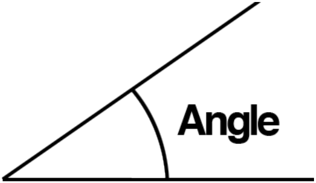
Cosine Similarity



$$\text{cos\_sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

where v_1 and v_2 are vectors, representing documents (e.g., IDF-weighted frequencies).

Cosine Similarity

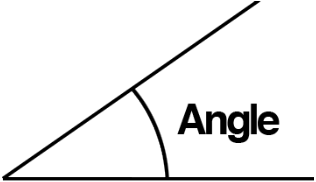


$$\text{cos\_sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

where v_1 and v_2 are vectors, representing documents (e.g., IDF-weighted frequencies).

- each document is a non-negative vector in an m-space ($m =$ size of dictionary):
  - closer vectors form smaller angles: $\cos(0) = +1$ means identical documents.
  - furthest vectors are orthogonal: $\cos(\pi/2) = 0$ means no words in common.

Cosine Similarity



$$\text{cos\_sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

where v_1 and v_2 are vectors, representing documents (e.g., IDF-weighted frequencies).

- each document is a non-negative vector in an m-space ($m =$ size of dictionary):
  - closer vectors form smaller angles: $\cos(0) = +1$ means identical documents.
  - furthest vectors are orthogonal: $\cos(\pi/2) = 0$ means no words in common.
- For n documents, this gives $n \times (n-1)$ similarities.

Text analysis of patent innovation

Kelly, Papanikolau, Seru, and Taddy (2018)

"Measuring technological innovation over the very long run"

- Data:
  - 9 million patents since 1840, from U.S. Patent Office and Google Scholar Patents.
  - date, inventor, backward citations
  - text (abstract, claims, and description)

Text analysis of patent innovation

Kelly, Papanikolau, Seru, and Taddy (2018)

"Measuring technological innovation over the very long run"

- Data:
  - 9 million patents since 1840, from U.S. Patent Office and Google Scholar Patents.
  - date, inventor, backward citations
  - text (abstract, claims, and description)
- Text pre-processing:
  - drop HTML markup, punctuation, numbers, capitalization, and stopwords.
  - remove terms that appear in less than 20 patents.
  - 1.6 million words in vocabulary.

Measuring Innovation

Kelly, Papanikolau, Seru, and Taddy (2018)

- Backward IDF weighting of word w in patent i:

$$\text{BIDF}(w, i) = \frac{\#\text{ of patents prior to } i}{\log\left(1 + \#\text{ patents prior to } i \text{ that include } w\right)}$$

  - down-weights words that appeared frequently before a patent.

Measuring Innovation

Kelly, Papanikolau, Seru, and Taddy (2018)

- Backward IDF weighting of word w in patent i:

$$\text{BIDF}(w,i) = \frac{\#\text{ of patents prior to } i}{\log\left(1 + \#\text{ patents prior to } i \text{ that include } w\right)}$$

  - down-weights words that appeared frequently before a patent.
- For each patent i:
  - compute cosine similarity ρ_{ij} to all future patents j, using BIDF of i.

Measuring Innovation

Kelly, Papanikolau, Seru, and Taddy (2018)

- Backward IDF weighting of word w in patent i:

$$\text{BIDF}(w,i) = \frac{\#\text{ of patents prior to } i}{\log\left(1 + \#\text{ patents prior to } i \text{ that include } w\right)}$$

  - down-weights words that appeared frequently before a patent.
- For each patent i:
  - compute cosine similarity ρ_{ij} to all future patents j, using BIDF of i.
- 9m$\times$9m similarity matrix = 30TB of data.
  - enforce sparsity by setting similarity $< .05$ to zero (93.4% of pairs).

Novelty, Impact, and Quality

Kelly, Papanikolau, Seru, and Taddy (2018)

- "Novelty" is defined by dissimilarity (negative similarity) to previous patents:

$$\text{Novelty}_j = -\sum_{i \in B(j)} \rho_{ij}$$

  where $B(j)$ is the set of previous patents (in, e.g., last 20 years).

Novelty, Impact, and Quality

Kelly, Papanikolau, Seru, and Taddy (2018)

- "Novelty" is defined by dissimilarity (negative similarity) to previous patents:

$$\text{Novelty}_j = -\sum_{i \in B(j)} \rho_{ij}$$

  where $B(j)$ is the set of previous patents (in, e.g., last 20 years).

- "Impact" is defined as similarity to subsequent patents:

$$\text{Impact}_i = \sum_{j \in F(i)} \rho_{ij}$$

  where $F(i)$ is the set of future patents (in, e.g., next 100 years).

Novelty, Impact, and Quality

Kelly, Papanikolau, Seru, and Taddy (2018)

- "Novelty" is defined by dissimilarity (negative similarity) to previous patents:

$$\text{Novelty}_j = -\sum_{i \in B(j)} \rho_{ij}$$

where $B(j)$ is the set of previous patents (in, e.g., last 20 years).

- "Impact" is defined as similarity to subsequent patents:

$$\text{Impact}_i = \sum_{j \in F(i)} \rho_{ij}$$

where $F(i)$ is the set of future patents (in, e.g., next 100 years).

- A patent has high **quality** if it is **novel** and **impactful**:

$$\log \text{Quality}_k = \log \text{Impact}_k + \log \text{Novelty}_k$$

Validation

Kelly, Papanikolau, Seru, and Taddy (2018)

- For pairs with higher ρ_{ij}, patent j more likely to cite patent i.

Validation

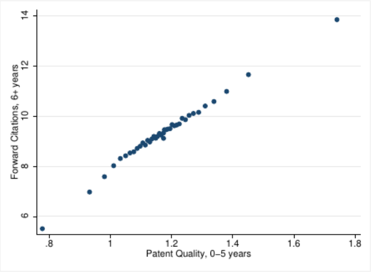
Kelly, Papanikolau, Seru, and Taddy (2018)

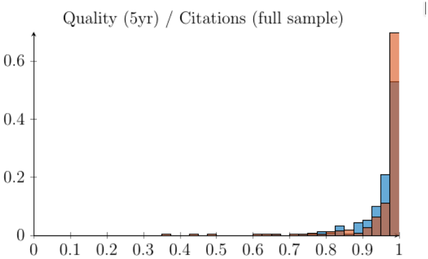
- For pairs with higher ρ_{ij}, patent j more likely to cite patent i.
- Within technology class (assigned by patent office), similarity is higher than across class.

Validation

Kelly, Papanikolau, Seru, and Taddy (2018)

- For pairs with higher ρ_{ij}, patent j more likely to cite patent i.
- Within technology class (assigned by patent office), similarity is higher than across class.
- Higher quality patents get more cites:





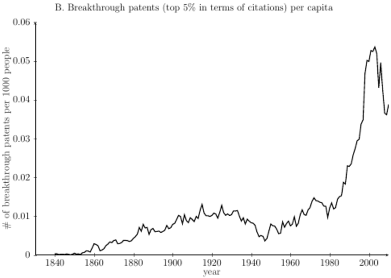
Most Innovative Firms

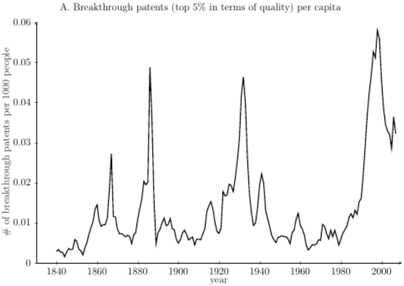
Kelly, Papanikolau, Seru, and Taddy (2018)

Assignee	First Year	# Breakthroughs
General Electric	1872	3,457
Westinghouse Electric Co.	1889	1,762
Eastman Kodak Co.	1890	2,244
Western Electric Co.	1899	1,222
AT&T (includes Bell Labs)	1899	5,645
Standard Oil Co.	1900	1,212
Dow Chemical Co.	1902	1,235
Du Pont	1905	3,353
International Business Machines	1908	14,913
American Cyanamid Co.	1909	690
Universal Oil Products Co.	1919	590
RCA	1920	3,222
Monsanto Company (inc. Monsanto Chemicals)	1921	902
Honeywell International, inc.	1928	872
General Aniline & Film Corp.	1929	1,181
Massachusetts Institute of Technology	1935	504
Philips	1939	1145
Texas Instruments	1960	2,088
Xerox	1961	2,198
Applied Materials	1971	510
Digital Equipment	1971	1,101
Hewlett-Packard Co.	1971	2,661
Intel	1971	2,629
Motorola, inc.	1971	4,129
Regents of the University of California	1971	823
United States Navy	1945	791
NCR	1973	737
Advanced Micro Devices	1974	1,195
Apple Computer	1978	864

Breakthrough patents: citations vs quality

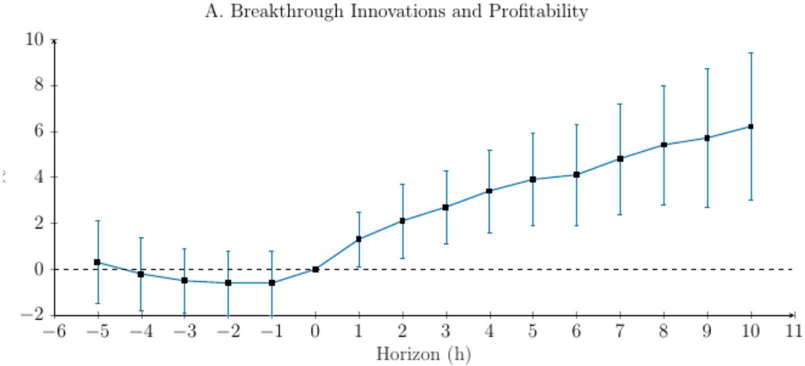
Kelly, Papanikolau, Seru, and Taddy (2018)





Breakthrough patents and firm profits

Kelly, Papanikolau, Seru, and Taddy (2018)



Outline

A baseline for machine learning using text

1. Take tf-idf-weighted POS-filtered bigrams (from above) as inputs X.

A baseline for machine learning using text

1. Take tf-idf-weighted POS-filtered bigrams (from above) as inputs X.
2. Train a machine learning model predict outcome y:
   2.1 For classification, regularized logistic regression (or gradient boosted classifier).
   2.2 For regression, use elastic net (or gradient boosted regressor).

A baseline for machine learning using text

1. Take tf-idf-weighted POS-filtered bigrams (from above) as inputs X.
2. Train a machine learning model predict outcome y:
   2.1 For classification, regularized logistic regression (or gradient boosted classifier).
   2.2 For regression, use elastic net (or gradient boosted regressor).
3. Use cross-validation grid search in training set to select model hyperparameters.

A baseline for machine learning using text

1. Take tf-idf-weighted POS-filtered bigrams (from above) as inputs X.
2. Train a machine learning model predict outcome y:
   2.1 For classification, regularized logistic regression (or gradient boosted classifier).
   2.2 For regression, use elastic net (or gradient boosted regressor).
3. Use cross-validation grid search in training set to select model hyperparameters.
4. Evaluate model in held-out test set:
   4.1 For classification, use F1 score and confusion matrix.
   4.2 For regression, use R squared and calibration plot.

Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," ***Political Analysis*** **(2018).**

Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," *Political Analysis* (2018).

- Machine Learning Problem:
  - Corpus $D = 3.5$M U.K. parliament speeches, 1935-2013.

Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," *Political Analysis* (2018).

- Machine Learning Problem:
  - Corpus $D = 3.5$M U.K. parliament speeches, 1935-2013.
  - Label $Y =$ party of speaker (Conservative or Labour)

Application: Predicting Political Party from Text

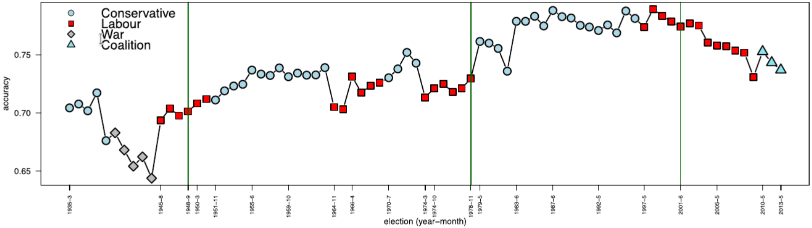
Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," *Political Analysis* (2018).

- Machine Learning Problem:
  - Corpus $D = 3.5$M U.K. parliament speeches, 1935-2013.
  - Label $Y =$ party of speaker (Conservative or Labour)

- Analysis:
  - In years that classifier is more accurate, speech is more polarized.

Polarization in U.K. Parliament

Peterson and Spirling (*Political Analysis* 2018)



- Accuracy of party prediction over time.