Text As Data - Lecture 1

Use in economics and pre-processing

Machine Learning and Big Data

Vincent Bagilet

Introduction

Text as Data

- A plethora of text data
- Nowadays, we have both the technology and methods to study them massively

- Text data is **unstructured**:
  - Basically not sorted in a sort of “table-like” format
  - Info we want mixed with info we do not want
  - Need to throw away some info $\Rightarrow$ select what to keep

- Text data is very **high-dimensional** $\left(n_{\text{vocab}}^{n_{\text{words in doc}}}\right)$

- We often want to relate text data to **metadata**
  - *eg* who, when, on what topic?

Think About Your Own Question

- What type of text data? What source?
- How would you get the data?
- Which research question?
- How would you go about studying this?

Outline and Resources

Housekeeping

Assignment

- **Exercise** to do for the end of next week
- You can start today, even if we will not have covered everything
- Available on *Portail des études*

Material

- Available on *Portail des études* AND on the course website

Objectives

- **Why** is text data useful **in economics**?
- What is a typical **empirical workflow**?
- How to **concretely implement** these analyses in Python?
- What are **recent developments** in the field?
- Link this section to the rest of the class

Relation to the rest of the class

- More on the data analysis and big data side than on the ML one
- Here, focus on **pre-processing steps**, *ie* prepare the data to use it in:
  - A ML algorithm
  - An econometric analysis
- Can use the tools and algorithms you saw in the rest of the class on text data
  - Examples?

Outline

1. Introduction
2. Applications in economics
3. Workflow for analysis
4. Pre-processing
5. Representation
6. Dictionary based methods
7. Machine learning methods
8. Introduction to deep learning methods and LLMs

Economics Resources



ANNUAL REVIEWS

Annual Review of Economics

Text Algorithms in Economics

Elliott Ash[1] and Stephen Hansen[2,3]

[1]Center for Law and Economics, ETH Zurich, Zurich, Switzerland

[2]Department of Economics, University College London, London, United Kingdom; email: stephen.hansen@ucl.ac.uk

[3]Centre for Economic Policy Research, London, United Kingdom

ANNUAL REVIEWS CONNECT
www.annualreviews.org
- Download figures
- Navigate cited references
- Keyword search
- Explore related articles
- Share via email or social media

Annu. Rev. Econ. 2023. 15:659–88

First published as a Review in Advance on July 5, 2023

The *Annual Review of Economics* is online at economics.annualreviews.org

https://doi.org/10.1146/annurev-economics-082222-074352

Copyright © 2023 by the author(s). This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See credit lines of images or other third-party material in this article for license information.

JEL codes: C18, C45, C55

Keywords

text as data, topic models, word embeddings, large language models, transformer models

Abstract

This article provides an overview of the methods used for algorithmic text analysis in economics, with a focus on three key contributions. First, we introduce methods for representing documents as high-dimensional count vectors over vocabulary terms, for representing words as vectors, and for representing word sequences as embedding vectors. Second, we define four core empirical tasks that encompass most text-as-data research in economics and enumerate the various approaches that have been taken so far to accomplish these tasks. Finally, we flag limitations in the current literature, with a focus on the challenge of validating algorithmic output.

OPEN ACCESS

Downloaded from www.annualreviews.org. Guest (guest) IP: 176.171.138.79 On: Sun, 24 Nov 2024 18:55:38



Journal of Economic Literature 2019, 57(3), 535–574
https://doi.org/10.1257/jel.20181020

Text as Data†

Matthew Gentzkow, Bryan Kelly, and Matt Taddy*

An ever-increasing share of human interaction, communication, and culture is recorded as digital text. We provide an introduction to the use of text as an input to economic research. We discuss the features that make text different from other forms of data, offer a practical overview of relevant statistical methods, and survey a variety of applications. (*JEL* C38, C55, L82, Z13)

1. *Introduction*

New technologies have made available vast quantities of digital text, recording an ever-increasing share of human interaction, communication, and culture. For social scientists, the information encoded in text is a rich complement to the more structured kinds of data traditionally used in research, and recent years have seen an explosion of empirical economics research using text as data.

To take just a few examples: In finance, text from financial news, social media, and company filings is used to predict asset price movements and study the causal impact of new information. In macroeconomics, text is used to forecast variation in inflation and unemployment, and estimate the effects of policy uncertainty. In media economics, text from news and social media is used to study the drivers and effects of political slant. In industrial organization and marketing, text from advertisements and product reviews is used to study the drivers of consumer decision making. In political economy, text from politicians' speeches is used to study the dynamics of political agendas and debate.

The most important way that text differs from the kinds of data often used in economics is that text is inherently high dimensional. Suppose that we have a sample of documents, each of which is w words long, and suppose that each word is drawn from a vocabulary of p possible words. Then the unique representation of these documents has dimension p^w. A sample of thirty-word Twitter messages that use only the one thousand most common words in the English language, for example, has roughly as many dimensions as there are atoms in the universe.

A consequence is that the statistical methods used to analyze text are closely related to those used to analyze high-dimensional data in other domains, such as machine learning and computational biology. Some methods, such as lasso and other penalized regressions, are applied to text more or less exactly as they are in other settings. Other methods, such as topic models and multinomial inverse regression, are close cousins of more general

* Gentzkow: Stanford University. Kelly: Yale University and AQR Capital Management. Taddy: University of Chicago Booth School of Business.

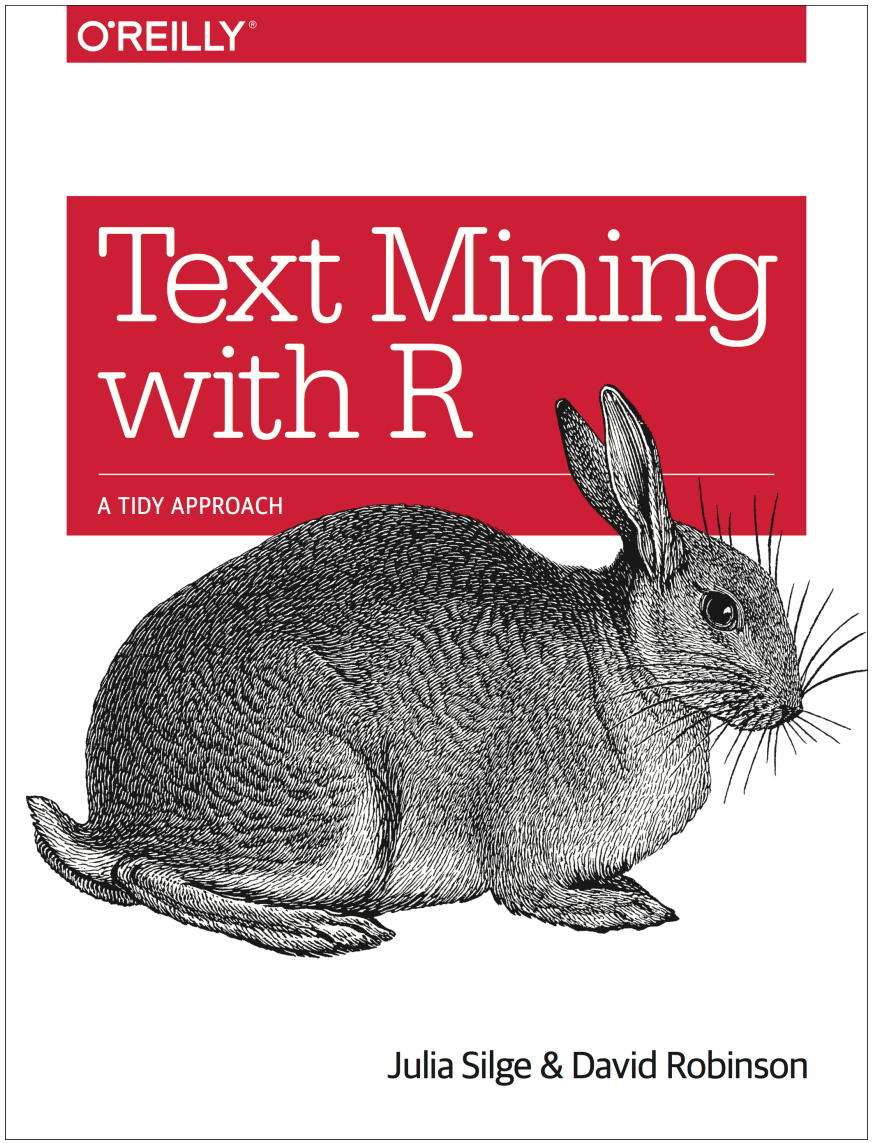
† Go to https://doi.org/10.1257/jel.20181020 to visit the article page and view author disclosure statement(s).

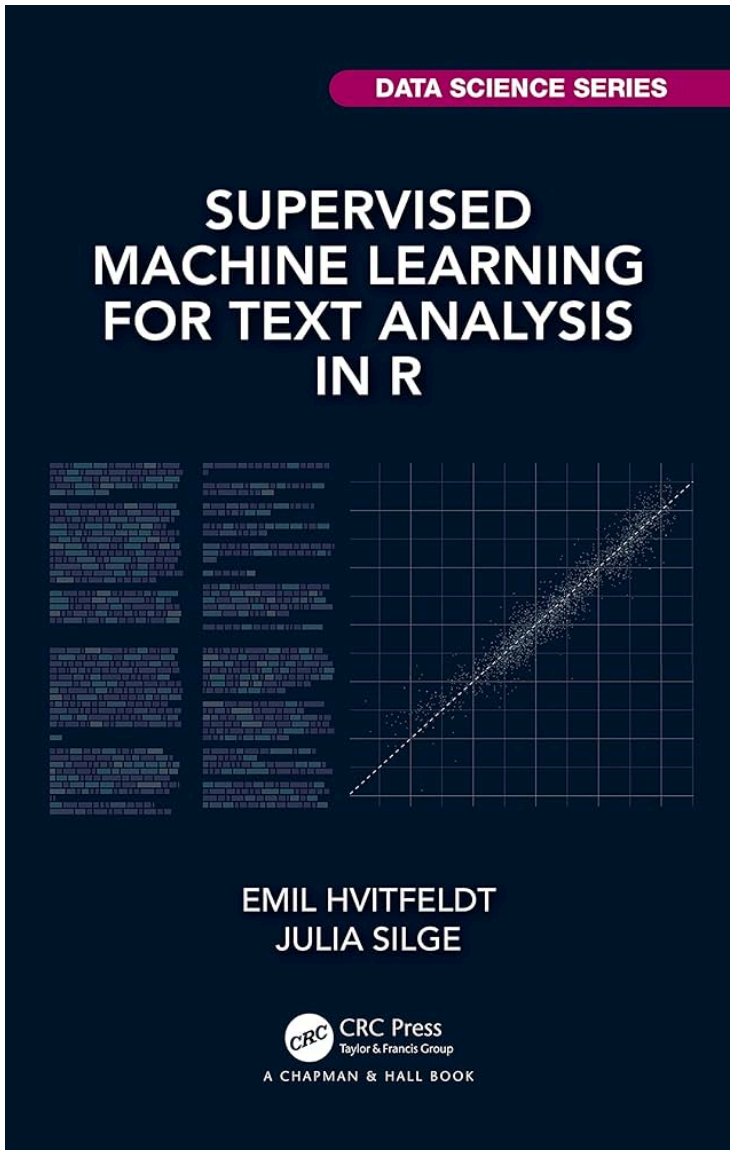
Elliott Ash's Text as Data class

Python Resources

- spaCy 101: tutorial for `spaCy`, a Python library for NLP
- Natural Language Processing with Python: general Python book on NLP (with the `nltk` library)
- Companion Python notebooks to Ash and Hansen (2023)

R Resources


O'REILLY
Text Mining with R
A TIDY APPROACH
Julia Silge & David Robinson


DATA SCIENCE SERIES
SUPERVISED MACHINE LEARNING FOR TEXT ANALYSIS IN R
EMIL HVITFELDT
JULIA SILGE
CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Applications in Economics

Measuring Document Similarity

- **What?**
  - Make pairwise document comparisons
  - Forming clusters of related documents
- **Examples**
  - Cagé, Hervé, and Viaud (2020) use the distance between online news articles and social media posts to group items into common stories
  - Biasi and Ma (2022) build a measure of **syllabi**'s distance from **frontier knowledge** (academic articles) and then relate this metric to socio-economic variables
- **How?**
  - Represent documents in some sort of vector form and then compute cosine similarity
  - There are different ways of defining vectors

Concept Detection

- **What?**
  - Detect the presence of a concept
- **Example**
  - Djourelova, Durante, and Martin (2024) interpretable news topics using CorEx
- **How?**
  - Dictionary methods / pattern matching
  - Topic models to identify latent concepts
  - Distance between documents and dictionaries
  - Supervised (human classification) learning problem
  - BERT/GPT type of models

Relation Between Concepts

- **What?**
  - How are concepts related?
- **How?**
  - Co-occurrence of dictionaries
  - Word Embedding Association Test (WEAT):
    - Picks two terms at each end of the spectrum (eg rich and poor) and compute the cosine similarity of each term of interest with these “extreme” terms
- **Examples**
  - Ash, Chen, and Ornaghi (2024) on gender attitudes of individual US judges
  - Kozlowski, Taddy, and Evans (2019) locate words on meaningful dimension, *eg*, rich/poor (this has a feel of Bourdieu’s *social space*)

Associating Text with Metadata

- **Impute** an outcome of interest to other documents
- Supervised learning
- *eg* have some data on party label but not for everyone
- **Example**
  - Gentzkow and Shapiro (2010) build a model for party label using US Congressional speeches. Then use it to predict political bias of newspapers

Workflow for Analysis

How are text data structured?

- Text data is a sequence of characters called **documents**
- A **corpus** is a collection of documents
- The unit of analysis (the “document”) depends on the question:
  - Fine enough to fit relevant metadata variation
  - Not unnecessarily fine to reduce dimensionality

From text to a usable format

- Text data is very highly dimensional
- Need to transforming data into a **useful representation** for modeling
- Encode text data into **numeric arrays**:
  - Tokenization and word counts
  - Document-Term Matrix
  - Latent Semantic Allocation
  - Word embeddings
- Both for computational reasons and to have objects we can manipulate and do computations on

Steps of Text Analysis

- Gentzkow, Kelly, and Taddy (2019) summarize a text analysis in three steps:
  1. Represent raw text D as a numerical array C
  2. Map C to predicted values $\widehat{V}$ of unknown outcomes V
  3. Use $\widehat{V}$ in subsequent descriptive or causal analysis

Workflow, rephrased

1. Collecting data
2. Pre-processing (stemming, lemming, etc)
3. Data transformation (from pre-processed text to numeric arrays)
4. Analysis and modelisation
5. Validation
6. Use in econometric analyses

Type of Analyses

- **Dictionary Methods**: rely on pre-defined lexicons and info associated with specific keywords
- **Rule-Based Methods**: use predefined rules and patterns (*eg* RegEx)
- **Machine Learning Methods**: methods leveraging lightweight ML models
  - Supervised ML
  - Unsupervised ML
- **Deep Learning Methods**: methods leveraging neural networks
- Off-the-shelf **Large Language Models** (LLM) based methods



There is an important **interpretability-flexibility trade-off**

Gathering data

Common Text Data Sources

- **Political economy**: news articles, parliamentary debates, speeches, social network posts, party manifestos, press releases, etc
- **Economic history**: correspondence, institutional documents, books, etc
- **Labor and Industrial Organisation (IO)**: job adds, product descriptions, etc
- **Finance and macro**: earnings conference calls, central bank speeches, etc

Where to find data

- Digital archives of **news** (Factiva, ProQuest, Europress, etc)
  - Expensive. Some universities have subscriptions.
- Companies **APIs** (*eg* Twitter[1], New York Times)
  - In general there are **wrappers** to directly access the data from Python or R
- Data sets of occurrence of keywords (Google Trends, Google Ngram, Gallicagram)
- **Online** ⇒ scrapping
- **Printed texts**: digitalize and OCR
- Curated lists of text data: here, here, here, and there

Overview of web scraping

- The overall idea is to **write algorithms** that:
  1. **Browse** a website and **download** relevant `html` pages
     - Main challenge: identify the relevant pages
  2. Transform the `html` pages into a **data frame** containing text data
     - Typical structure: date column, title column, author column, text column, etc
     - Main challenge: identify the right tags
- Not limited to text data: you can retrieve anything that is online

HTML pages

- HTML: *HyperText Markup Language*
- Structure of an example html page

```
<html>
<head>
  <title>Page title</title>
</head>
<body>
  <h1 id='first'>A heading</h1>
  <p>Some text &amp;amp; <b>some bold text.</b></p>
  <img src='myimg.png' width='100' height='100'>
</body>
```

- An IMDb example

Webscraping concretely

- There are libraries for web scraping with useful **documentation**:
  - Beautiful Soup (Python)
  - rvest (R)
- Basic pages are **static** and easy to scrape
- More complex to scrape **dynamic** pages
  - Use Selenium
  - It opens a browser and can be used to automate tasks

A few webscraping tips

Tips

- Before scraping, look at whether someone did not already scraped what you want to scrape (*eg* for IMDb)
- To identify relevant pages use the site **sitemap** (website.com/sitemap.xml)
    - Example
- Use **ChatGPT** or Claude for scraping
- Use SelectorGadget or tools from the developer panel of your browser
- Save html pages before processing them
- Read the `robots.txt` page to learn what you can and cannot do

Optical Character Recognition (OCR)

- Some data is either not digitized or digitized but in text/PDF format
- Use software to transform it to text data
- Tesseract and its Python and R wrappers
- When PDFs are digitally made, you can use a text extraction software to directly retrieve the text from the metadata

Pre-processing

Goal

- Going from raw text to something usable in analysis
- ↘ the dimensionality by ↘ the size of the vocabulary
- Identify some words as the same
- Some information is useful, other less:
  - **What to keep?**
  - **What format?**

Tokens

- **Tokens**: a meaningful unit of text, such as a word, a character
- In NLP, often split data into tokens
- Example from the French parliament: *"Ne fumez pas la moquette"*
- Words: *{Ne, fumez, pas, la, moquette}*
- n-grams (here 2-grams): *{Ne fumez, fumez pas, pas la, la moquette}*
- Sentences

```
1  import nltk
2
3  # Run only once
4  nltk.download('punkt')
5  nltk.download('punkt_tab')
6
7  sentence = "Ne fumez pas la moquette"
8  tokens = nltk.word_tokenize(sentence)
9
10 tokens
```

```
['Ne', 'fumez', 'pas', 'la', 'moquette']
```

Pre-processing Choices

- Pre-processing choices often affects the results:





What To Keep?

- Not all data is useful
- Sometimes, may remove useful information: *eg* “happy” vs “not happy”
- Remove capitalization?
- Punctuation?
- Numbers?
- Stopwords?
    - Words that carry little information
    - How do you define this set?

Capitalization and punctuation

- Often uninformative
- But sometimes important:
  - Sentence splitting
  - Part-of-speech tagging
  - Named entity recognition
  - Text generation
- Option: keep caps not at the beginning of a sentence

```
1 tokens_lower = [word.lower() for word in tokens]
2 print(tokens_lower)
```

```
['ne', 'fumez', 'pas', 'la', 'moquette']
```

Stop words

- We have seen that might create noise but may also carry meaning
- How to define stopwords?
- Can use existing lists

```
stopwords = set(nltk.corpus.stopwords.words('french'))

print(stopwords)
```

```
{'ta', 'eusse', 'lui', 'une', 'ou', 'sont', 'soyez', 'avez', 'nos', 'vos', 'mes', 'n', 'as', 'étés', 'te', 'es', 'eurent',
'fusse', 'la', 'serons', 'était', 'aurons', 'aurait', 'étées', 'êtes', 'ne', 'eussiez', 'ses', 'se', 'eux', 'aurai', 'ayons',
'eut', 'eusses', 'que', 'eue', 'ayants', 'aviez', 'eu', 'le', 'serais', 'même', 'avions', 'eût', 'fûmes', 'avec', 'seras',
'soit', 'aies', 'étaient', 'eussent', 'les', 'auriez', 'étais', 'fussent', 'ayez', 'avons', 'par', 'aura', 'avaient', 'm',
'vous', 'qu', 'ma', 'fusses', 'avais', 'fus', 'votre', 'du', 'pour', 'ayant', 'dans', 'l', 'suis', 'ayante', 'ont', 'aient',
'nous', 'sa', 'ton', 'je', 't', 'd', 'étiez', 'étions', 'me', 'pas', 'aurions', 'notre', 'étée', 'serions', 'aurais',
'soyons', 'auraient', 'qui', 'serai', 'eus', 'furent', 'c', 'ai', 'tes', 'eues', 'été', 'ils', 'en', 'fûtes', 'son', 'fut',
'ait', 'de', 'moi', 'y', 'serez', 'seront', 'il', 'seriez', 'et', 'serait', 'j', 's', 'fussiez', 'soient', 'des', 'elle',
'eûtes', 'on', 'aie', 'est', 'auras', 'tu', 'étantes', 'étant', 'aux', 'mon', 'au', 'étante', 'fussions', 'sois', 'sera',
'aurez', 'avait', 'sommes', 'auront', 'toi', 'sur', 'eûmes', 'étants', 'seraient', 'leur', 'un', 'ces', 'ayantes', 'mais',
'ce', 'à', 'fût', 'eussions'}
```

```
tokens_filtered = [word for word in tokens_lower if word not in stopwords]
print(tokens_filtered)
```

```
['fumez', 'moquette']
```

- Which list? How to define it?

- Build one
  - By hand, starting from existing ones
  - Remove words that appear the most frequently, *eg* by Inverse Document Frequency (idf):

$$idf(term) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{document containing term}}}\right)$$

- *idf* decreases the weight of commonly used words and increases that of words that are rarely used in the corpus
- *tf-idf*: multiplying the term frequency (*tf*) and its *idf*

Stemming/Lemmatization

- Find the stem of the word:
  - Ruled based
  - Produces grams that are not actual words

```
stemmer = nltk.stem.snowball.SnowballStemmer(language="french")
tokens_stemmed = [stemmer.stem(word) for word in tokens]
print(tokens_stemmed)
```

```
['ne', 'fum', 'pas', 'la', 'moquet']
```

- Lemmatization:
  - Similar but with semantic rules
  - Produces actual words but sentences that do not mean anything

```
import spacy
nlp = spacy.load("fr_core_news_sm")
doc = nlp(sentence)
lemmas = [token.lemma_ for token in doc]

print(lemmas)
```

```
['ne', 'fumer', 'pas', 'le', 'moquette']
```

Transforming your data

From text to numbers

- Transform data to be able to use it in algorithms
- Main data structures for documents in NLP:
  - As raw text
  - As a sequence of tokens for each document
  - As a vector, stored in a matrix (document-term matrix, embedding matrix)

Occurrences and counts

- Raw **count** of occurrences of each term by document
- **Term Frequency (tf)** of word w in document d (weight by the length of the document):

$$tf_{wd} = \frac{\text{Number of occurences of w in d}}{\text{Total number of tokens in d}}$$

- Account for the **specificity** of the term for the document: **tf-idf** (multiplying the term frequency (*tf*) and its *idf*)

$$idf(term) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

- *idf* decreases the weight of commonly used words and increases that of words that are rarely used in the corpus
- Can also use other transformations: dummy indicator of presence of a word in a document, log counts, etc

Matrices

- Then create **Document-Term Matrices** that represent each document and each word in the vocabulary (*ie* in the corpus)

	economy	policy	growth
Doc1	3	1	0
Doc2	0	2	4
Doc3	1	0	1
Doc4	2	3	1

- These are **bag-of-words** (BoW) approaches:
  - Put all the term in a document together, regarding of their order (and count them)
  - Loose information about order between terms

Tf-idf implementation

```
1  import pandas as pd
2  from sklearn.feature_extraction.text import TfidfVectorizer
3
4  assemblee_2018 = pd.read_csv("../../data/assemblee_2018.csv")
5
6  assemblee_small = assemblee_2018.head(20)
7  assemblee_small = assemblee_small[['speaker_name', 'date', 'text']]
8
9  vectorizer = TfidfVectorizer()
10 X = vectorizer.fit_transform(assemblee_small['text'])
11 X
```

```
<20x254 sparse matrix of type '<class 'numpy.float64'>'
    with 400 stored elements in Compressed Sparse Row format>
```

```
1  pd.DataFrame(data = X.toarray()).style.set_table_attributes('style="display:block; max-height:300px; overflow-y:auto;"')
```

	0	1	2	3	4	5	6	7
0	0.000000	0.000000	0.204050	0.000000	0.204050	0.204050	0.204050	0.000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
3	0.000000	0.234636	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
5	0.220997	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

Dimensionality reduction

- Massive matrices ($n_{docs} \times n_{vocab}$)
- Even if **sparse** (contains mostly zeros), it might take up a **lot of RAM** to run algorithms on these
- Plus, we often do not care about the words in a document but about the **underlying meaning**
- Want vectors to be able to capture meaning
- Reduce the dimensionality of the matrices to capture this meaning and lower computing load:
  - Apply Principal Component Analysis (PCA) to the matrix: called **Latent Semantic Analysis**
  - Latent Dirichlet Allocation (**LDA**)
  - **Word embeddings**: many approaches. We will discuss them a bit later.

Usefulness

- After transformation, documents are represented as sequences of tokens or as vectors
- Can now use these representations **for our intended tasks**
- Compute document similarity, for instance by calculating cosine between two document-vectors (**cosine-similarity**)
- Identify presence of concepts:
  - Count occurrences of words or dictionaries, build RegEx, train a ML algorithm, etc
  - Identify clusters of documents (*ie* of vectors)
  - Some of theses tasks **only involve matrix products**

Application of similarity analysis

- Bertrand et al. (2021): *Hall of Mirrors: Corporate Philanthropy and Strategic Advocacy*

> **The paper in one line**
>
> - Show that when nonprofits receive donations from firms, they tend to comment more on the same US federal regulatory rules but also adopt a closer type of comments

- Approach to similarity analysis:
  1. Collapse the documents to organization-rule-year-level observations
  2. Apply **LSA** with tf-idf weighting to the matrix
  3. Compute **cosine similarity** between documents
  4. **Regress** similarity measure on dummy for donation (and FEs and controls)
- **Compare** LSA to a Doc2vec and LDA on a similar task: predicting if two documents come from the same docket (documents pertaining to a same narrow topic)

On your own data set

On your own dataset

What similarity question could you ask on you own data?

Summary

NLP in Economics

- Measuring document similarity
- Concept detection
- Relation between concepts
- Associating text with metadata

Overal Approach

- **Get** text data (ready-made, scrap, OCR, etc)
- **Pre-process** the data
- **Transform** data into useful format (a numeric array)
- **Run** analysis. Several methods:
  - Dictionary-based
  - Rule-based
  - Machine Learning
  - Deep Learning
- Use output in an **econometric** analysis

References

Ash, Elliott, Daniel L. Chen, and Arianna Ornaghi. 2024. "Gender Attitudes in the Judiciary: Evidence from US Circuit Courts." *American Economic Journal: Applied Economics* 16 (1): 314–50. https://doi.org/10.1257/app.20210435.

Ash, Elliott, and Stephen Hansen. 2023. "Text Algorithms in Economics." *Annual Review of Economics* 15 (1): 659–88. https://doi.org/10.1146/annurev-economics-082222-074352.

Bertrand, Marianne, Matilde Bombardini, Raymond Fisman, Brad Hackinen, and Francesco Trebbi. 2021. "Hall of Mirrors: Corporate Philanthropy and Strategic Advocacy." *The Quarterly Journal of Economics* 136 (4): 2413–65. https://doi.org/10.1093/qje/qjab023.

Biasi, Barbara, and Song Ma. 2022. "The Education-Innovation Gap." Working {{Paper}}. Working Paper Series. National Bureau of Economic Research. https://doi.org/10.3386/w29853.

Cagé, Julia, Nicolas Hervé, and Marie-Luce Viaud. 2020. "The Production of Information in an Online World." *The Review of Economic Studies* 87 (5): 2126–64. https://doi.org/10.1093/restud/rdz061.

Djourelova, Milena, Ruben Durante, and Gregory J Martin. 2024. "The Impact of Online Competition on Local Newspapers: Evidence from the Introduction of Craigslist." *The Review of Economic Studies*, May, rdae049. https://doi.org/10.1093/restud/rdae049.

Gentzkow, Matthew, Bryan Kelly, and Matt Taddy. 2019. "Text as Data." *Journal of Economic Literature* 57 (3): 535–74. https://doi.org/10.1257/jel.20181020.

Gentzkow, Matthew, and Jesse M Shapiro. 2010. "What Drives Media Slant? Evidence From U.S. Daily Newspapers." *Econometrica* 78 (1): 35–71. https://doi.org/10.3982/ECTA7195.

Kozlowski, Austin C., Matt Taddy, and James A. Evans. 2019. "The Geometry of Culture: Analyzing the Meanings of Class Through Word Embeddings." *American Sociological Review*, September. https://doi.org/10.1177/0003122419877135.