

Lecture 6: Text Data, Image Data, and Foundation Models

Isaiah Hull^{1,2}

¹BI Norwegian Business School

²CogniFrame

November 28, 2023



Overview

Lecture 6: Overview

1. Text Data.
2. Image Applications.
3. Text Applications.

1. Text Data

Text, Images, and Foundation Models

Overview

- ▶ Based on Gentzkow et al. (2019) and Hull (2021).
 - ▶ <https://www.aeaweb.org/articles?id=10.1257/jel.20181020>
 - ▶ <https://link.springer.com/book/10.1007/978-1-4842-6373-0>

Introduction

Course Materials



<https://linktr.ee/mlcourse>

“Text as Data”

- Gentzkow, Taddy, and Kelly (2019)

Introduction

- ▶ Increased availability of text datasets.
- ▶ Many successful applications outside of economics and growing literature within economics.
- ▶ Inherently high-dimensional and requires different statistical methods.
- ▶ Paper provides overview of methods and literature.

Introduction

Overview

- ▶ Representing text as data
- ▶ Statistical methods
- ▶ Applications
- ▶ Pseudocode

Representing Text as Data

Representing Text as Data

- ▶ Must convert text to numerical format before applying statistical methods.
- ▶ Need a mapping from raw text, D , to numerical array, C .
- ▶ Typically want to use C to predict outcomes, V .
- ▶ Use \hat{V} in descriptive or causal analysis.

Representing Text as Data

- ▶ What is D ?
 - ▶ Document corpus: $\{D_0, \dots, D_{n-1}\}$.
- ▶ What is C ?
 - ▶ Language token counts.
 - ▶ E.g. word counts.
- ▶ What is V ?
 - ▶ Dependent variable.
 - ▶ E.g. policy outcome.

Representing Text as Data

- ▶ D_0 : Riksbank policy announcement (03/07/2019).

'Economic activity in Sweden remains strong and inflation is close to the target of 2 per cent. Uncertainty abroad has increased but new information since the monetary policy decision in April has not led to any major revisions of the forecasts overall. With continued support from monetary policy, the conditions for inflation to remain close to the target in the period ahead are considered good. The Executive Board has decided to hold the repo rate unchanged at -0.25 per cent. The forecast for the repo rate is also unchanged and indicates that it will be increased again towards the end of the year or at the beginning of next year. However, the risks surrounding developments abroad can have a bearing on the prospects for Sweden, which emphasises the importance of proceeding cautiously with monetary policy.'

- ▶ V_0 : -0.25 (05/09/2019).

Representing Text as Data

- ▶ D_1 : Riksbank policy announcement (25/04/2019).

'Activity in the Swedish economy is high and inflation is close to the target of 2 per cent. The strong economic activity in Sweden indicates that inflation will remain close to target going forward but recent outcomes suggest that inflationary pressures are slightly weaker than expected. The Executive Board has decided to hold the repo rate unchanged at -0.25 per cent and assesses that the rate will remain at this level for a somewhat longer period of time than was forecast in February. The repo rate is expected to be raised again towards the end of the year or at the beginning of next year and rate rises thereafter are expected to occur at a somewhat slower pace. The Executive Board has also decided that the Riksbank will purchase government bonds for a nominal value of SEK 45 billion from July 2019 to December 2020.'

- ▶ V_1 : -0.25 (03/07/2019).

Representing Text as Data

Feature Selection

1. Remove non-word elements.
2. Convert to lower case.
3. Remove stop words.
4. Remove rare words.
5. Stemming.
6. Lemmatization.

Representing Text as Data

Non-Word Elements: Regular Expression

```
>>> fun remove_non_words(doc):  
    text = regex('[^A-Za-z]+', ' ', doc)  
    lower = lower_case(text)  
    word_tokens = word_tokenize(lower)  
    return word_tokens
```

```
>>> words_d0 = remove_non_words(d0)  
>>> words_d1 = remove_non_words(d1)
```

Representing Text as Data

Stop Words: NLTK

'u'a', u'about', u'above', u'after', u'again', u'against', u'ain', u'all', u'am', u'an',
u'and', u'any', u'are', u'aren', u"aren't", u'as', u'at', u'be', u'because', ' ... ', u'from',
u'further', u'had', u'hadn', u"hadn't", u'has', u'hasn', u"hasn't", u'have', u'haven',
u"haven't", u'having', u'he', u'her', u'here', u'hers', ' ... ', u'what', u'when',
u'where', u'which', u'while', u'who', u'whom', u'why', u'will', u'with', u'won',
u"won't", u'wouldn', u"wouldn't", u'y', u'you', u"you'd", u"you'll", u"you're",
u"you've", u'your', u'yours', u'yourself', u'yourselves''

Representing Text as Data

Remove Stop Words

```
>>> fun remove_stop_words(words):  
    cleaned_words = []  
    for word in words:  
        if word not in stopwords:  
            cleaned_words.append(word)  
    return cleaned_words  
  
>>> words_d0 = remove_stop_words(words_d0)  
>>> words_d1 = remove_stop_words(words_d1)
```

Representing Text as Data

Data Cleaning: D_0

```
>>> print(words_d0)
```

```
['economic', 'activity', 'sweden', 'remains', 'strong', 'inflation', 'close', 'target', 'per', 'cent',  
'uncertainty', 'abroad', 'increased', 'new', 'information', 'since', 'monetary', 'policy', 'decision',  
'april', 'led', 'major', 'revisions', 'forecasts', 'overall', 'continued', 'support', 'monetary', 'policy',  
'conditions', 'inflation', 'remain', 'close', 'target', 'period', 'ahead', 'considered', 'good', 'executive',  
'board', 'decided', 'hold', 'repo', 'rate', 'unchanged', 'per', 'cent', 'forecast', 'repo', 'rate', 'also',  
'unchanged', 'indicates', 'increased', 'towards', 'end', 'year', 'beginning', 'next', 'year', 'however',  
'risks', 'surrounding', 'developments', 'abroad', 'bearing', 'prospects', 'sweden', 'emphasises',  
'importance', 'proceeding', 'cautiously', 'monetary', 'policy']
```

Representing Text as Data

Data Cleaning: D_1

```
>>> print(words_d1)
```

```
['activity', 'swedish', 'economy', 'high', 'inflation', 'close', 'target', 'per', 'cent', 'strong', 'economic',  
'activity', 'sweden', 'indicates', 'inflation', 'remain', 'close', 'target', 'going', 'forward', 'recent',  
'outcomes', 'suggest', 'inflationary', 'pressures', 'slightly', 'weaker', 'expected', 'executive', 'board',  
'decided', 'hold', 'repo', 'rate', 'unchanged', 'per', 'cent', 'assesses', 'rate', 'remain', 'level',  
'somewhat', 'longer', 'period', 'time', 'forecast', 'february', 'repo', 'rate', 'expected', 'raised',  
'towards', 'end', 'year', 'beginning', 'next', 'year', 'rate', 'rises', 'thereafter', 'expected', 'occur',  
'somewhat', 'slower', 'pace', 'executive', 'board', 'also', 'decided', 'riksbank', 'purchase',  
'government', 'bonds', 'nominal', 'value', 'sek', 'billion', 'july', 'december']
```

Representing Text as Data

(Optional) Stemming: D_0

```
>>> print(porter_stemmer(words_d0))
```

```
['u'econom', u'activ', 'sweden', u'remain', 'strong', u'inflat', 'close', 'target', 'per', 'cent',  
u'uncertainti', 'abroad', u'increas', 'new', u'inform', u'sinc', u'monetari', u'polici', u'decis', 'april',  
'led', 'major', u'revis', u'forecast', u'overal', u'continu', 'support', u'monetari', u'polici', u'condit',  
u'inflat', 'remain', 'close', 'target', 'period', 'ahead', u'consid', 'good', u'execut', 'board', u'decid',  
'hold', 'repo', 'rate', u'unchang', 'per', 'cent', 'forecast', 'repo', 'rate', 'also', u'unchang', u'indic',  
u'increas', u'toward', 'end', 'year', u'begin', 'next', 'year', u'howev', u'risk', u'surround', u'develop',  
'abroad', u'bear', u'prospect', 'sweden', u'emphasis', u'import', u'proceed', u'cautious', u'monetari',  
u'polici']
```

Representing Text as Data

(Optional) Stemming: D_1

```
>>> print(porter_stemmer(words_d1))
```

```
[u'activ', u'swedish', u'economi', 'high', u'inflat', 'close', 'target', 'per', 'cent', 'strong', u'econom',  
u'activ', 'sweden', u'indic', u'inflat', 'remain', 'close', 'target', u'go', 'forward', 'recent', u'outcom',  
'suggest', u'inflationari', u'pressur', u'slightli', 'weaker', u'expect', u'execut', 'board', u'decid',  
'hold', 'repo', 'rate', u'unchang', 'per', 'cent', u'assess', 'rate', 'remain', 'level', 'somewhat', 'longer',  
'period', 'time', 'forecast', u'februari', 'repo', 'rate', u'expect', u'rais', u'toward', 'end', 'year',  
u'begin', 'next', 'year', 'rate', u'rise', u'thereaft', u'expect', 'occur', 'somewhat', 'slower', 'pace',  
u'execut', 'board', 'also', u'decid', 'riksbank', u'purchas', u'govern', u'bond', u'nomin', u'valu', 'sek',  
'billion', u'juli', u'decemb']
```

Representing Text as Data

(Optional) Lemmatization and Rare Word Removal

- ▶ Lemmatization includes context and allows for inclusion of parts-of-speech (POS) tags.
 - ▶ Stemmer: Better → Better
 - ▶ Lemmatizer: Better → Good
- ▶ Identifying and dropping rare words reduces dimensionality of estimation problem.

Representing Text as Data

Feature Selection

$$\blacktriangleright [D_0, D_1] \rightarrow [C_0, C_1]$$

$$C = \begin{bmatrix} 1 & 2 & . & . & 2 & 0 \\ 4 & 1 & . & . & 0 & 1 \end{bmatrix}$$

Representing Text as Data

Term-Frequency Inverse-Document-Frequency (tf-idf)

► $tfidf_j = (\sum_i c_{ij}) * \log(n / \sum_i \mathbf{1}_{[c_{ij} > 0]})$

$$[5 * \log(2/2) \quad 3 * \log(2/2) \quad . \quad . \quad 2 * \log(2/1) \quad 1 * \log(2/1)]$$

- Remove words with low tf-idf scores.

Representing Text as Data

N-grams

- ▶ Used bag-of-words approach to construct C .
 - ▶ Sequence of words does not matter.
 - ▶ Only consider single words.
- ▶ Can construct C using N-grams.
 - ▶ Use sequence of N words.
 - ▶ Need a column for every pair of words.
 - ▶ Most research doesn't go beyond 3-grams.

Representing Text as Data

Bigrams: D_0

```
>>> bigrams_d0 = ngrams(words_d0, 2)
>>> print(bigrams_d0)
```

```
('monetary', 'policy'): 3, ('close', 'target'): 2, ('repo', 'rate'): 2, ('per', 'cent'): 2, ('indicates',  
'increased'): 1, ('considered', 'good'): 1, ('emphasises', 'importance'): 1, ('unchanged', 'indicates'): 1,  
('policy', 'conditions'): 1, ('sweden', 'emphasises'): 1, ...'
```

Representing Text as Data

Bigrams: D_1

```
>>> bigrams_d1 = ngrams(words_d1, 2)
```

```
>>> print(bigrams_d1)
```

```
('close', 'target'): 2, ('executive', 'board'): 2, ('per', 'cent'): 2, ('repo', 'rate'): 2, ('weaker', 'expected'):  
1, ('indicates', 'inflation'): 1, ('expected', 'occur'): 1, ('nominal', 'value'): 1, ('rate', 'expected'): 1,  
('rate', 'remain'): 1, ...'
```

Representing Text as Data

Trigrams: D_0

```
>>> trigrams_d0 = ngrams(words_d0, 3)
```

```
>>> print(trigrams_d0)
```

```
('information', 'since', 'monetary'): 1, ('decided', 'hold', 'repo'): 1, ('cent', 'forecast', 'repo'): 1,  
('surrounding', 'developments', 'abroad'): 1, ('new', 'information', 'since'): 1, ('executive', 'board',  
'decided'): 1, ('since', 'monetary', 'policy'): 1, ('next', 'year', 'however'): 1, ('year', 'however',  
'risks'): 1, ('cautiously', 'monetary', 'policy'): 1, ...'
```

Representing Text as Data

Trigrams: D_1

```
>>> trigrams_d1 = ngrams(words_d1, 3)
```

```
>>> print(trigrams_d1)
```

```
('decided', 'hold', 'repo'): 1, ('economy', 'high', 'inflation'): 1, ('purchase', 'government', 'bonds'): 1,  
('high', 'inflation', 'close'): 1, ('hold', 'repo', 'rate'): 1, ('board', 'also', 'decided'): 1, ('executive',  
'board', 'decided'): 1, ('remain', 'close', 'target'): 1, ('assesses', 'rate', 'remain'): 1, ('inflation',  
'remain', 'close'): 1, ...'
```

Representing Text as Data

Richer Representations

- ▶ Uncommon in economics and the social sciences.
- ▶ Most research uses bag-of-words approach.
- ▶ Construct n-grams based on meaning dependence.
- ▶ Work outside of economics often makes use of sequence data.

Statistical Methods

Statistical Methods

Workflow

1. Select Features: $D \rightarrow C$.
2. Split Sample: $(C, V) \rightarrow [(C^{train}, V^{train}), (C^{test}, V^{test})]$.
3. Train Model: $f(C^{train}) \rightarrow \hat{V}^{train}$.
4. Validate Model: $f(C^{test}) \rightarrow \hat{V}^{test}$

Statistical Methods

Dimensions

```
>>> print(dim(C))  
(n, p)  
>>> print(dim(V))  
(n, k)  
>>> print(dim( $C^{train}$ ))  
( $n^{train}$ , p)  
>>> print(dim( $V^{train}$ ))  
( $n^{train}$ , k)  
>>> print(dim( $C^{test}$ ))  
( $n^{test}$ , p)  
>>> print(dim( $V^{test}$ ))  
( $n^{test}$ , k)
```

Statistical Methods

Selecting $f(.)$

1. Dictionary-based methods.
2. Text regression.
3. Generative models.
4. Word embeddings.

Statistical Methods

Dictionary-Based Methods

- ▶ Do not use statistical inference.
- ▶ Most commonly-used method in social sciences.
- ▶ Researcher defines $f(\cdot)$ based on dictionary of words.
- ▶ Applying $f(\cdot)$ recovers latent variable, such as sentiment.

Statistical Methods

Dictionary-Based Methods

- ▶ Tetlock (2007): Bag-of-words. Measures latent “sentiment” in WSJ articles using counts of positive and negative words.
- ▶ Baker et al. (2016): Intersection of terms about “policy,” “economics,” and “uncertainty” to construct EPU index in newspaper articles.

Popular Economics Dictionaries

- ▶ Loughran & McDonald (2011): Dictionaries based on 10-K financial filings. Includes negative, positive, uncertain, litigious, and other sentiments.
- ▶ Apel & Blix Grimaldi (2014): Construct net hawkishness index for analyzing central bank communication. Pair nouns with adjectives to reduce noise.

Statistical Methods

Dictionary-Based Methods: Example

$$\text{Positivity: } \frac{2}{135}, \text{ Negativity: } \frac{2}{135}, \text{ Net Positivity: } 0 = \frac{2}{135} - \frac{2}{135}$$

Economic activity in Sweden remains **strong** and inflation is close to the target of 2 per cent. **Uncertainty** abroad has increased but new information since the monetary policy decision in April has not led to any major revisions of the forecasts overall. With continued support from monetary policy, the conditions for inflation to remain close to the target in the period ahead are considered **good**. The Executive Board has decided to hold the repo rate unchanged at -0.25 per cent. The forecast for the repo rate is also unchanged and indicates that it will be increased again towards the end of the year or at the beginning of next year. However, the **risks** surrounding developments abroad can have a bearing on the prospects for Sweden, which emphasises the importance of proceeding cautiously with monetary policy.

Text Regression

1. Penalized linear models.
2. Dimension reduction.
3. Nonlinear models.
4. Bayesian models.

Penalized Linear Models

- ▶ Simple and fast to estimate.
- ▶ Familiar for economists.
- ▶ Near-frontier performance for many problems.

Penalized Linear Model

$$\eta_i = \alpha + \mathbf{c}_i' \boldsymbol{\beta} \quad (1)$$

$$\min \left\{ l(\alpha, \boldsymbol{\beta}) + n\lambda \sum_{j=1}^p \kappa_j(|\beta_j|) \right\} \quad (2)$$

OLS

$$l(\alpha, \beta) = \sum_i (v_i - \eta_i)^2 \quad (3)$$

Binomial Logit

$$l(\alpha, \beta) = - \sum_i [\eta_i v_i - \log(1 + e^{\eta_i})] \quad (4)$$

Statistical Methods

Parameters

- ▶ λ = overall penalty magnitude
- ▶ κ_j = cost functions that penalize β_j
- ▶ L_1 penalty popular

Cost Functions

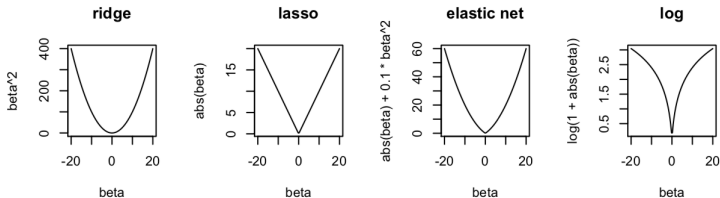


Figure 1: From left to right, L_2 costs (ridge, [Hoerl and Kennard 1970](#)), L_1 (lasso, [Tibshirani 1996](#)), the “elastic net” mixture of L_1 and L_2 ([Zou and Hastie 2005](#)), and the log penalty ([Candes et al. 2008](#)).

L_1 Cost Function

$$\min \left\{ l(\alpha, \beta) + n\lambda \sum_{j=1}^p \omega_j |\beta_j| \right\} \quad (5)$$

- ▶ ω_j proportional to covariate standard deviation
 - ▶ Manning et al. (2008): “rare feature upweighting”

Statistical Methods

λ Selection

- ▶ No agreement in literature
- ▶ Out-of-sample fit (e.g. MSE)
- ▶ K-fold cross-validation fit
- ▶ AIC or BIC

Dimension Reduction

- ▶ Replace regressors with linear combinations
 - ▶ Principal components regression (PCR)
 - ▶ Partial least squares (PLS)

Principal Components Analysis

- ▶ PCA yields low-rank representation that approximates C well.

```
>>> print(dim( $\Gamma$ ))
```

```
(n, K)
```

```
>>> print(dim(B))
```

```
(p, K)
```

```
>>> print(dim(C))
```

```
(n, p)
```


Principal Components Regression

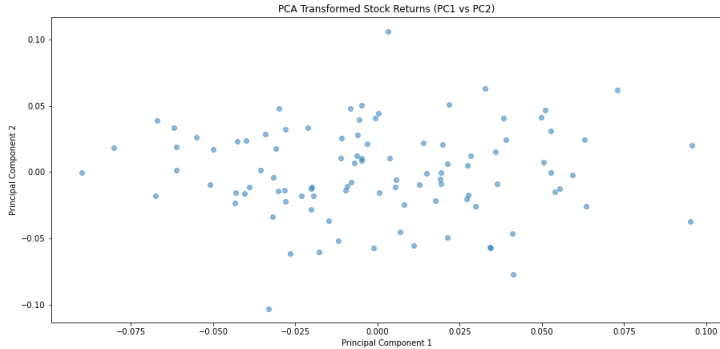
- ▶ Γ captures K common components or factors.
- ▶ B describes strength of associations between words and factors.
- ▶ K components are used in a standard regression.

PCA Example

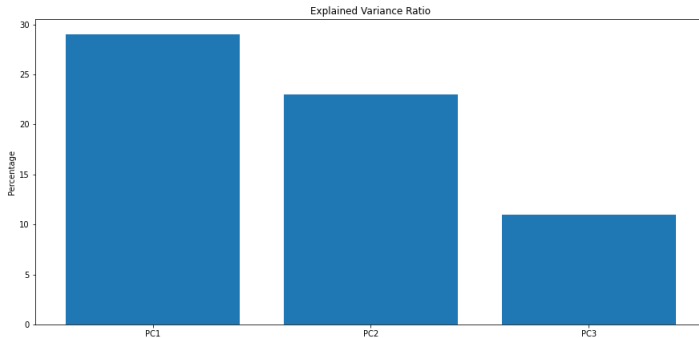


Statistical Methods

PCA Example



PCA Example



Partial Linear Regression

- ▶ PCA fails to account for variation in V .
- ▶ PLR explicitly allows for this.

Partial Linear Regression

1. Estimate univariate covariance between v_i and c_{ij} .
2. Construct single aggregate predictor.

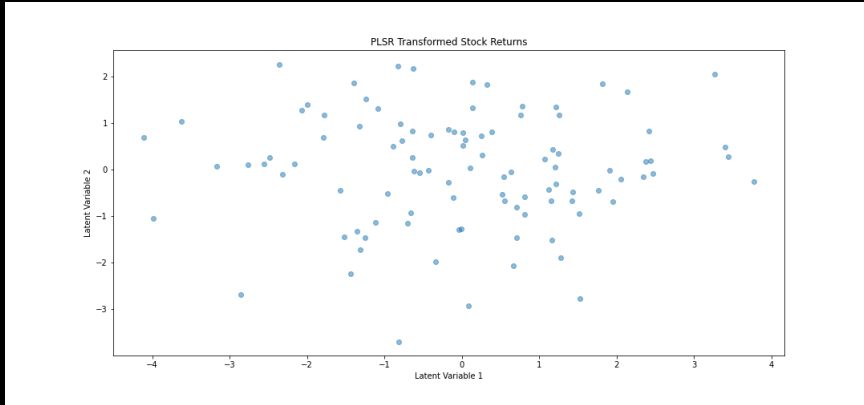
$$\hat{v}_i = \sum_j \phi_j c_{ij} / \sum_j \phi_j \quad (6)$$

3. For $K > 1$ case, repeat with orthogonalization.

PLR Example



PLR Example



Nonlinear Text Regression

- ▶ GLM: Allow for nonlinear transformations of C .
- ▶ SVM: Used for classification problems.
- ▶ Regression Trees: Allow for nonlinearity and higher-order interactions.
- ▶ Deep Learning: Perform feature discovery and estimate nonlinear associations.

Generative Models

- ▶ Want to know $p(c_i|v_i)$, rather than $p(v_i|c_i)$.
- ▶ Unsupervised Models: No direct observations on v_i .
- ▶ Supervised Models: The attributes, v_i , are observed during training.

Topic Model (LDA)

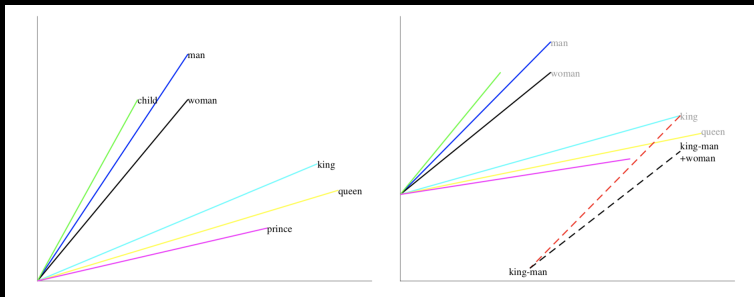
- ▶ Unsupervised generative model.
- ▶ Each topic is a probability vector over possible tokens.
- ▶ Can allow for a mix of topics or restrict to single topic.

Statistical Methods

Word Embeddings

- ▶ Move from token counts, C , to alternative representation of text.
- ▶ Words as points in vector space.
- ▶ Allows for clusters of words to become co-located in space.
- ▶ Example: {king, queen, prince, man, woman, child}

Word Embeddings



Statistical Methods

Word Embeddings

Dimension	king	queen	prince	man	woman	child
Royalty	0.99	0.99	0.95	0.01	0.02	0.01
Masculinity	0.94	0.06	0.02	0.99	0.02	0.49
Age	0.73	0.81	0.15	0.61	0.68	0.09
...						

Word Embeddings

- ▶ $\text{queen} = \text{king} - \text{man} + \text{woman}$
- ▶ $\text{prince} = \text{king} - \text{man} + \text{child}$

Pretrained Word Embeddings

- ▶ Pretrained word embeddings are sufficient for many problems.
 - ▶ Word2Vec (Mikolov et al., 2013).
 - ▶ Global Vector for Word Representation (Pennington et al., 2014).

Statistical Methods

Selecting a Method

- ▶ Dictionary-Based Methods: Prior information strong and reliable. Information in data is weak.
- ▶ Text Regression: Good for predicting attribute when large amount of labeled data is available.
- ▶ Generative Models: Multiple attributes of interest and wish to resolve or control for interdependencies.

Applications

Applications

Authorship

- ▶ Stock and Trebbi (2003): Who wrote the appendix of “The Tariff on Animal and Vegetable Oils”?
- ▶ Appendix introduces instrumental variables regression in a “succinct and insightful” way.
- ▶ Rest of the book is a “painfully detailed treatise...”
- ▶ Some have speculated that the author’s son, Sewall, wrote the appendix.
- ▶ Manski (1998): Sewall wrote it.
- ▶ Angrist and Krueger (2001): Philip wrote it.

Applications

Authorship

- ▶ Stock and Trebbi (2003).
 - ▶ Extract four principal components.
 - ▶ Perform regressions with authors as the dependent variable.
- ▶ $C^{train} = 45$ documents with known authorship.
- ▶ $C^{test} = 8$ blocks of text from appendix with unknown authorship.

Applications

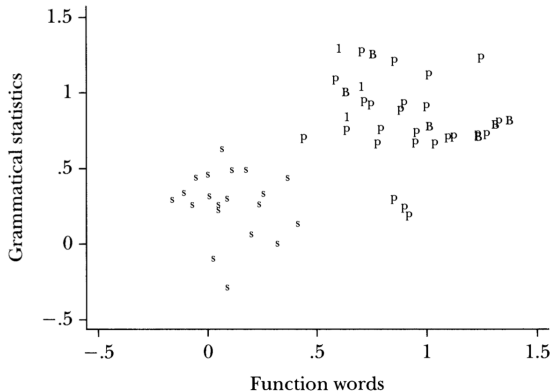
Authorship

s = block undisputedly written by Sewall Wright

p = block undisputedly written by Philip G. Wright

1 = block from chapter 1, *The Tariff on Animal and Vegetable Oils*

B = block from Appendix B, *The Tariff on Animal and Vegetable Oils*



Applications

Stock Prices

- ▶ Tetlock (2007): Uses dictionary-based method to measure media sentiment. Pessimism score forecasts one-day-ahead DJI returns.
- ▶ Wisniewski and Lambe (2013): Negative media attention Granger-causes bank stock returns during Great Recession.
- ▶ Jegadeesh and Wu (2013): Use token counts from annual reports to predict stock returns.

Applications

Stock Prices

- ▶ Manela and Moreira (2015): Use nonlinear text regression to predict news-implied market volatility.
- ▶ Bandiera et al. (2017): Use unsupervised ML (LDA) to uncover topics in CEO diaries. Show that CEO-firm match important for performance.

Applications

Central Bank Communication

- ▶ Lucca and Trebbi (2009): Predict Treasury yields with net hawkishness index.
- ▶ Born et al. (2014): Measure impact of central bank sentiment on stock market returns and volatility.
- ▶ Hansen et al. (2017): Examine how central bank transparency affects debates. Use topic modeling on 149 FOMC meeting transcripts.

Applications

Nowcasting

- ▶ Choi and Varian (2012): Predict macroeconomic variables using Google search results.
- ▶ Saiz and Simonsohn (2013): Measure corruption using search engine queries.

Applications

Policy uncertainty

- ▶ Baker et al. (2016): Measure economic policy uncertainty using text data from newspapers.

Conclusions

- ▶ Substantial growth in availability of text-based datasets.
- ▶ Text data is inherently high-dimensional and, consequently, requires a different set of models than economists typically use.
- ▶ Most work in economics has focused on dictionary-based methods.
- ▶ Outside of economics, penalized regression methods, deep learning, generative models, and dimensionality reduction techniques dominate.

Introduction to Scraping and NLP

Google Colab: Scraping



Overview

Scraping

1. GET requests
2. HTML parsing
3. Scraping ethics

Introduction

How does a browser work?¹

- ▶ Alice has a website.
- ▶ Bob wants to view Alice's site.
- ▶ Bob's computer creates a header and body.
 - ▶ Header = MAC address + Alice's IP address.
 - ▶ Body = GET request for index.html.
- ▶ Header and body bundled as packet and sent to Alice via intermediary servers.
- ▶ Alice's server locates the file and returns it to Bob.

¹See Mitchell (2015) for additional detail.

Introduction

How does a browser work?

- ▶ A browser isn't needed for anything Bob did.
- ▶ We can do this with Python.

Introduction

How does a browser work?

#example1.py

\$user python

```
>>> from urllib.request import urlopen  
>>> url = "http://www.math.unm.edu/  
writingHTML/tut/index.html"  
>>> html = urlopen(url)  
>>> print(html.read())
```

Introduction

How does a browser work?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
<html>
```

```
<head>
```

```
<title>Writing HTML</title>
```

```
<META name="..."> </head>
```

```
<body bgcolor="#FFFFFF">
```

```
...
```

```
<p>
```

We suggest that you proceed through the lessons in order, but at any time you can return to the index to jump to a different lesson. Within each lesson you can compare your work to a sample file for that lesson. ...

```
<p>
```

```
...
```

```
</body>
```

```
</html>
```

Scraping

BeautifulSoup

- ▶ Python package for HTML and XML parsing.
- ▶ Generates parse tree from extracted code.
- ▶ Fixes HTML errors, layouts, and indenting.

Scraping

BeautifulSoup

► Broken HTML

```
<ul class=country>  
  <li>Area  
  <li>Population  
</ul>
```

Scraping

BeautifulSoup

#example2.py

\$user pip install bs4

\$user python

```
>>> from bs4 import BeautifulSoup
```

```
>>> broken_html = "<ul class=country><li>Area<li>Population</ul>"
```

```
>>> soup = BeautifulSoup(broken_html)
```

```
>>> fixed_html = soup.prettify()
```

```
>>> print(fixed_html)
```

Scraping

BeautifulSoup

► Fixed HTML

```
<ul class="country">  
  <li>  
    Area  
  </li>  
  <li>  
    Population  
  </li>  
</ul>
```

Scraping

BeautifulSoup

#example3.py

\$user python

```
>>> from urllib.request import urlopen
```

```
>>> from bs4 import BeautifulSoup
```

```
>>> url = "https://en.wikipedia.org/  
wiki/Richard_Thaler"
```

```
>>> html = urlopen(url)
```

```
>>> soup = BeautifulSoup(html.read())
```

```
>>> print(soup.h1)
```

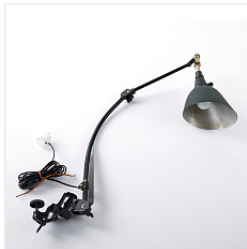
```
<h1 class="firstHeading" id="firstHeading" lang="en">Richard Thaler</h1>
```

```
>>> print(soup.h1.text)
```

Richard Thaler

BeautifulSoup

All auctions



492993 CURT FISCHER, bordslampa, "114", Midgard, 1930-tal, ...
Estimate: SEK 5 000 SEK
🕒 < 1 minute left, 2017-10-15 kl 14:42
Slakthusgatan 22, Stockholm
Bid: SEK 3 000 SEK



492971 BÖRGE MOGENSEN, stolar, 6 st, ek, tygklädsel, Karl ...
Estimate: SEK 3 000 SEK
🕒 4 minutes left, 2017-10-15 kl 14:45
Magasin 5, Stockholm



492917 CHANEL, sjal mönstrad med broscher på brun ullbotten med ...
Estimate: SEK 2 000 SEK
🕒 7 minutes left, 2017-10-15 kl 14:48
Magasin 5, Stockholm
Bid: SEK 1 800 SEK

<https://online.auktionsverket.se>

Scraping

BeautifulSoup

#example5.py

\$user python

```
>>> from urllib.request import urlopen
>>> from bs4 import BeautifulSoup
>>> url = "https://online.auktionsverket.com/"
>>> html = urlopen(url)
>>> soup = BeautifulSoup(html.read())
```

Scraping

BeautifulSoup

#example5.py (continued)

```
>>> soup.title
<title>Stockholms Auktionsverk Online</title>
>>> soup.title.name
'title'
>>> soup.title.string
u'Stockholms Auktionsverk Online'
>>> soup.title.parent.name
'head'
>>> soup.p
<p>Invitation to consign</p>
```

Scraping

BeautifulSoup

#example5.py (continued)

```
>>> soup.a
```

```
<a href="/"><div style="padding-left:0.7em; padding-right:0.4em; float:left;">Home</div></a>
```

```
>>> soup.find_all('a')
```

```
[<a href="/"><div style="padding-left:0.7em; padding-right:0.4em; float:left;">Home</div></a>, <a href="http://auktionsverket.com/news/"><div style="padding-left:0.25em; padding-right:0.4em; float:left; margin-left:0.4em;">Right now</div></a>, ...
```

Scraping

BeautifulSoup

#example5.py (continued)

```
>>> links = soup.find_all('a')
```

```
>>> for link in links[:5]:
```

```
    print(link)
```

```
<a href="/"><div style="padding-left:0.7em; padding-right: ...
```

```
<a href="http://auktionsverket.com/news/"> ...
```

```
<a href="http://auktionsverket.com/buy/kop-online...
```

```
<a href="http://auktionsverket.com/salja/">...
```

```
<a href="http://auktionsverket.com/sell/valuations/">...
```

Introduction

Scraping ethics

- ▶ Check robots.txt file before scraping.
 - ▶ Lists suggested restrictions on automated interaction.
 - ▶ Good Internet ethics.
 - ▶ Lowers probability of an IP ban.
 - ▶ Protocol: <http://www.robotstxt.org>
- ▶ Robots.txt examples:
 - ▶ <https://www.ecb.europa.eu/robots.txt>
 - ▶ <http://www.bcb.gov.br/robots.txt>

Introduction

Scraping ethics

User-agent: *

Sitemap: <https://www.ecb.europa.eu/sitemap.xml>

Disallow: /*_content.bg.html\$

Disallow: /*_content.cs.html\$

Disallow: /*_content.da.html\$

Disallow: /*_content.de.html\$

Disallow: /*_content.el.html\$

...

Disallow: /ecb/10ann/shared/movies/

Disallow: /ecb/educational/pricestab/shared/movie/

Disallow: /ecb/educational/shared/movies/

...

Crawl-delay: 5

Introduction

Scraping ethics

- ▶ **User-agent: *** → Rules apply to all users.
- ▶ **Sitemap: ...** → Sitemap located here.
- ▶ **Disallow: ...** → Don't scrape this URL.
- ▶ **Crawl-delay: 5** → Use a 5-second delay between requests.

Introduction

Scraping ethics

- ▶ **User-agent: Offline Explorer/1.9**
Disallow: /
 - ▶ → User-agent: Offline Explorer/1.9 shouldn't scrape any pages.
- ▶ **User-agent: htdig/3.1.4 (****@bcb.gov.br)**
Disallow:
 - ▶ → User-agent: htdig/3.1.4 (****@bcb.gov.br) is unrestricted.
- ▶ **User-agent: ***
Disallow: css/*
 - ▶ → All other user-agents are restricted.

Natural Language Processing

Overview

Natural Language Processing

1. Text cleaning
2. Text classification
3. Sentiment analysis
4. Topic models

Text cleaning

Installation

```
#example6.py
```

```
$user pip install nltk
```

```
$user python
```

```
>>> import nltk
```

```
>>> nltk.download('all')
```

```
>>> from nltk.corpus import reuters
```

```
>>> print(reuters.categories())
```

Processing and understanding text

Tokenization

[u'acq', u'alum', u'barley', u'bop', u'carcass', u'castor-oil', u'cocoa', u'coconut',
u'coconut-oil', u'coffee', u'copper', u'copra-cake', u'corn', u'cotton', u'cotton-oil',
u'cpi', u'cpu', u'crude', u'dfl', u'dlr', u'dmk', u'earn', u'fuel', u'gas', u'gnp',
u'gold', u'grain', u'groundnut', u'groundnut-oil', u'heat', u'hog', ... , u'reserves',
u'retail', u'rice', u'rubber', u'rye', u'ship', u'silver', u'sorghum', u'soy-meal',
u'soy-oil', u'soybean', u'strategic-metal', u'sugar', u'sun-meal', u'sun-oil',
u'sunseed', u'tea', u'tin', u'trade', u'veg-oil', u'wheat', u'wpi', u'yen', u'zinc']

Text cleaning

Tokenization

```
#example6.py
```

```
>>> print(reuters.fileids(['gold']))
```

Processing and understanding text

Tokenization

[u'test/14842', u'test/15411', u'test/15471', u'test/15481', u'test/15803',
u'test/15811', u'test/16009', u'test/16149', u'test/16162', u'test/16212',
u'test/16248', u'test/16286', u'test/16589', u'test/16604', u'test/17457',
u'test/17622', u'test/17632', u'test/17654', ..., u'training/8068', u'training/8331',
u'training/8757', u'training/8854', u'training/8857', u'training/8877',
u'training/8948', u'training/9104', u'training/9190', u'training/9453',
u'training/9489', u'training/9799', u'training/9853', u'training/9866']

Text cleaning

Tokenization

#example6.py (continued)

```
>>> gold = reuters.raw(fileids='training/9799')
```

```
>>> print len(gold)
```

```
1315
```

```
>>> print(gold[:50])
```

```
'GERMAN ANALYSTS SEE GOLD RISING IN 2ND HALF 1987'
```

Text cleaning

Tokenization

#example6.py (continued)

```
>>> goldSentences = nltk.sent_tokenize(gold)
```

```
>>> print(goldSentences[4])
```


Processing and understanding text

Tokenization

Despite current strong interest in gold mine stocks, many investors still want to buy physical gold, Witte said.

Text cleaning

Tokenization

#example6.py (continued)

```
>>> goldWords = nltk.word_tokenize(gold)
```

```
>>> print(len(goldWords))
```

```
241
```

```
>>> print(goldWords[150:160])
```

Processing and understanding text

Tokenization

[u'stocks',
u'may',
u'also',
u'wane',
u'if',
u'stock',
u'exchange',
u'rallies',
u'under',
u'way']

Text cleaning

Tokenization

#example6.py

```
>>> goldWords = [w.lower() for w in goldWords]
>>> fdist = nltk.FreqDist(goldWords)
>>> print('Rise :' + str(fdist['rise']))
>>> print('Fall :' + str(fdist['fall']))
```

Rise: 3

Fall: 1

Cleaning Raw Text

Speech

September 26, 2017

Inflation, Uncertainty, and Monetary Policy

Chair Janet L. Yellen

At the "Prospects for Growth: Reassessing the Fundamentals" 59th Annual Meeting of the National Association for Business Economics, Cleveland, Ohio

Share ➡

I would like to thank the National Association for Business Economics for inviting me to speak today and for the vital role the association plays in fostering debate on important economic policy questions.

Today I will discuss uncertainty and monetary policy, particularly as it relates to recent inflation developments. Because changes in interest rates influence economic activity and inflation with a substantial lag, the Federal Open Market Committee (FOMC) sets monetary policy with an eye to its effects on the outlook for the economy. But the outlook is subject to considerable uncertainty from multiple sources, and dealing with these uncertainties is an important feature of policymaking.

Text cleaning


Cleaning Raw Text


Model-Based Decomposition of ECI Hourly Compensation Growth

	ECI growth	Contributions of:			
		Expected inflation	Trend productivity	Slack	Other
2002-07	3.28	2.09	1.44	-0.08	-0.17
2008-09	1.80	2.16	1.18	-0.97	-0.56
2010-11	2.08	2.14	0.82	-1.09	0.22
2012-13	1.92	2.12	0.44	-0.67	0.02
2014-15	2.05	2.01	0.15	-0.20	0.09
2016-17:Q2	2.31	1.99	0.01	0.03	0.28

Note: ECI growth is reported as average percent changes at an annual rate for the periods shown; contributions are expressed in percentage points. The contribution of the model's constant term is included in the contribution for trend productivity. Contributions may not sum to total growth because of rounding.

References

Aaronson, Daniel, Luojia Hu, Arian Seifoddini, and Daniel G. Sullivan (2015). "Changing Labor Force Composition and the Natural Rate of Unemployment ", "Chicago Fed Letter 338. Chicago: Federal Reserve Bank of Chicago, May.

Aaronson, Stephanie, Tomaz Cajner, Bruce Fallick, Felix Galbis-Reig, Christopher Smith, and William Wascher (2014). "Labor Force Participation: Recent Developments and Future Prospects ", "Brookings Papers on Economic Activity, Fall, pp. 197-255.

Text cleaning

Cleaning Raw Text

#example7.py

\$user python

```
>>> from urllib.request import urlopen
>>> from bs4 import BeautifulSoup
>>> import nltk
>>> url = "https://www.federalreserve.gov/
newsevents/speech/yellen20170926a.htm"
>>> html = urlopen(url)
>>> soup = BeautifulSoup(html.read())
```

Text cleaning

Cleaning Raw Text

#example7.py (continued)

```
>>> paragraphs = soup.find_all('p')
```

```
>>> paragraphs = [p.text for p in paragraphs]
```


Text cleaning

Cleaning Raw Text

#example7.py (continued)

```
>>> len(paragraphs)
```

```
154
```

```
>>> speech = ' '.join(paragraphs)
```

```
>>> speech = speech.split('References')[0]
```

Text cleaning

Cleaning Raw Text

#example7.py (continued)

```
>>> wordTokens = nltk.word_tokenize(speech)
>>> wordTokens = [w.lower() for w in wordTokens]
>>> speechLength = len(wordTokens)
>>> fdist = nltk.FreqDist(wordTokens)
>>> print(fdist['inflation'])
```

120

Text cleaning

Cleaning Raw Text

#example7.py (continued)

```
>>> print(speechLength)  
5960
```

```
>>> inflationCount = fdist['inflation']  
>>> print(100.0*inflationCount/speechLength)  
2.0134228187919465
```

Text cleaning

Cleaning Raw Text

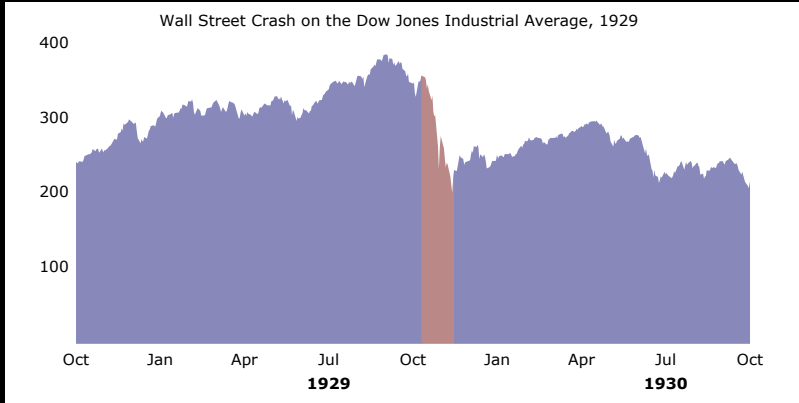
#example7.py (continued)

```
>>> from nltk.corpus import stopwords
>>> stops = stopwords.words('english')
>>> wordTokens = [word for word in wordTokens if word not in stops]
>>> fdist = nltk.FreqDist(wordTokens)
>>> print(100.0*fdist['inflation']/len(wordTokens))

3.078501795792714
```

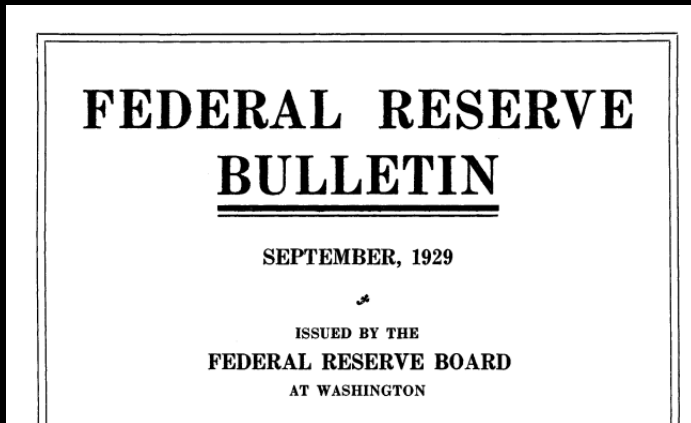
Text cleaning

Cleaning Raw Text



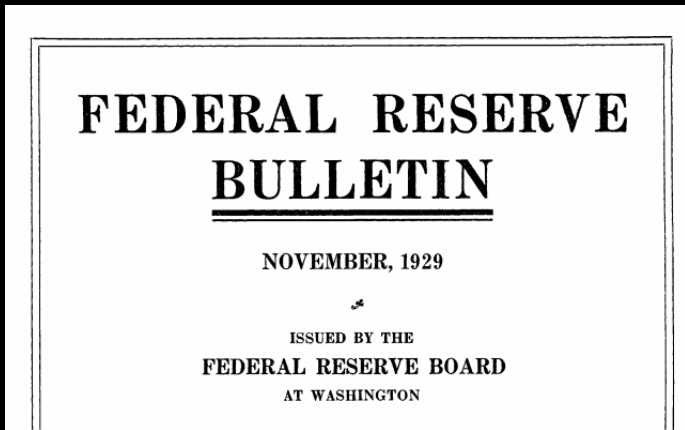
https://en.wikipedia.org/wiki/Wall_Street_Crash_of_1929

Cleaning Raw Text



Text cleaning

Cleaning Raw Text



Text cleaning

Cleaning Raw Text

#example8.py

\$user pip install PyPDF3

\$user python

```
>>> from urllib.request import urlopen
```

```
>>> import nltk
```

```
>>> from nltk.corpus import stopwords
```

```
>>> from nltk.stem.porter import PorterStemmer
```

```
>>> from PyPDF3 import PdfFileReader
```

```
>>> import re
```


Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> save_dir = '../Downloads/'
>>> url_09_29 = "https://fraser.stlouisfed.org/files/
docs/publications/FRB/1920s/frb_091929.pdf"
>>> url_11_29 = "https://fraser.stlouisfed.org/files/
docs/publications/FRB/1920s/frb_111929.pdf"
>>> content_09_29 = urlopen(url_09_29).read()
>>> content_11_29 = urlopen(url_11_29).read()
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> with open(save_dir+'frb_0929.pdf', 'wb') as f:  
    f.write(content_09_29)
```

```
>>> with open(save_dir+'frb_1129.pdf', 'wb') as f:  
    f.write(content_11_29)
```

```
>>> pdf_09_29 = PdfFileReader(  
open(save_dir+'frb_0929.pdf', 'rb'))
```

```
>>> pdf_11_29 = PdfFileReader(  
open(save_dir+'rb_1129.pdf'), 'rb'))
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> pdf_09_29_text = [pdf_09_29.getPage(j).  
extractText() for j in range(pdf_09_29.numPages)]  
>>> pdf_11_29_text = [pdf_11_29.getPage(j).  
extractText() for j in range(pdf_11_29.numPages)]  
>>> pdf_09_29_text = ' '.join(pdf_09_29_text)  
>>> pdf_11_29_text = ' '.join(pdf_11_29_text)
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> def tokenize(text):
    text = re.sub('[^A-Za-z]+', ' ', text)
    wordTokens = nltk.word_tokenize(text)
    wordTokens = [token.lower() for token in
wordTokens if len(token)>1]
    stops = set(stopwords.words("english"))
    wordTokens = [token for token in
wordTokens if token not in stops]
    return wordTokens

>>> tokens_09_29 = tokenize(pdf_09_29_text)
>>> tokens_11_29 = tokenize(pdf_11_29_text)
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> porter_stemmer = PorterStemmer()
>>> stems_09_29 = [porter_stemmer.stem(token) for token in tokens_09_29]
>>> stems_11_29 = [porter_stemmer.stem(token) for token in tokens_11_29]
>>> fdist_09_29 = nltk.FreqDist(stems_09_29)
>>> fdist_11_29 = nltk.FreqDist(stems_11_29)
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> print(fdist_09_29.most_common(5))
```

Text cleaning

Cleaning Raw Text

1. (u'bank', 573)

2. (u'aug', 315)

3. (u'reserv', 258)

4. (u'feder', 161)

5. (u'juli', 130)]

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> print(fdist_11_29.most_common(5))
```


Processing and understanding text

Cleaning Raw Text

1. (u'oct', 415)
2. (u'bank', 305)
3. (u'reserv', 184)
4. (u'feder', 149)
5. (u'index', 106)]

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> more_stops = ['june','juli','aug','sep','sept',  
'oct','nov','octob','feder','reserv','nation',  
'cent','new','year','month','per','total']
```

```
>>> fdist_09_29 = nltk.FreqDist([stem for stem in stems_09_29 if stem not in  
more_stops])
```

```
>>> fdist_11_29 = nltk.FreqDist([stem for stem in stems_11_29 if stem not in  
more_stops])
```

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> print(fdist_09_29.most_common(5))
```

Processing and understanding text

Cleaning Raw Text

1. (u'bank', 573)

2. (u'rate', 125)

3. (u'loan', 120)

4. (u'gold', 110)

5. (u'foreign', 107)

Text cleaning

Cleaning Raw Text

#example8.py (continued)

```
>>> print(fdist_11_29.most_common(5))
```

Text cleaning

Cleaning Raw Text

1. (u'bank', 305)
2. (u'index', 106)
3. (u'industri', 101)
4. (u'loan', 63)
5. (u'employ', 55)

Text Classification

Text classification

Overview

- ▶ Unsupervised Learning
 - ▶ No target.
 - ▶ Clustering.
 - ▶ Pattern detection.
- ▶ Supervised Learning
 - ▶ Continuous target → regression analysis.
 - ▶ Discrete target → classification.

Text classification

Classifying Articles

#example8.py

\$user python

>>> import nltk

>>> from nltk.corpus import reuters, stopwords

>>> import re

Text classification

Classifying Articles

#example8.py (continued)

```
>>> import numpy as np
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> from sklearn.naive_bayes import GaussianNB
>>> from sklearn.metrics import confusion_matrix
>>> print(reuters.categories())
```

Text classification

Classifying Articles

[u'acq', u'alum', u'barley', u'bop', u'carcass', u'castor-oil', u'cocoa', u'coconut',
u'coconut-oil', u'coffee', u'copper', u'copra-cake', u'corn', u'cotton', u'cotton-oil',
u'cpi', u'cpu', u'crude', u'dfl', u'dlr', u'dmk', u'earn', u'fuel', u'gas', u'gnp',
u'gold', u'grain', u'groundnut', u'groundnut-oil', u'heat', u'hog', ... , u'reserves',
u'retail', u'rice', u'rubber', u'rye', u'ship', u'silver', u'sorghum', u'soy-meal',
u'soy-oil', u'soybean', u'strategic-metal', u'sugar', u'sun-meal', u'sun-oil',
u'sunseed', u'tea', u'tin', u'trade', u'veg-oil', u'wheat', u'wpi', u'yen', u'zinc']

Text classification

Classifying Articles

#example8.py (continued)

```
>>> print(reuters.fileids(['corn']))
```

Text classification

Classifying Articles

[u'test/14832', u'test/14858', u'test/15033', u'test/15043', u'test/15106',
u'test/15287', u'test/15341', u'test/15618', u'test/15648', u'test/15676',
u'test/15686', u'test/15720', u'test/15845', u'test/15856', u'test/15860',
u'test/15863', u'test/15871',...,u'training/8535', u'training/855', u'training/8759',
u'training/8941', u'training/8983', u'training/8993', u'training/9058',
u'training/9093', u'training/9094', u'training/934', u'training/9470',
u'training/9521', u'training/9667', u'training/97', u'training/9865',
u'training/9958', u'training/9989']

Text classification

Classifying Articles

#example8.py (continued)

```
>>> print(reuters.fileids(['wheat']))
```

Text classification

Classifying Articles

[u'test/14841', u'test/15043', u'test/15097', u'test/15132', u'test/15271',
u'test/15273', u'test/15341', u'test/15388', u'test/15472', u'test/15500',
u'test/15567', u'test/15572', u'test/15582',..., u'training/8179', u'training/8273',
u'training/8413', u'training/8535', u'training/856', u'training/8604',
u'training/874', u'training/8759', u'training/8993', u'training/9021',
u'training/9095', u'training/97', u'training/9773', u'training/9782',
u'training/9793', u'training/9865']

Text classification

Classifying Articles

#example8.py (continued)

```
>>> corn = reuters.fileids(['corn'])
>>> wheat = reuters.fileids(['wheat'])
>>> common = set(corn).intersection(wheat)
>>> corn = [id for id in corn if id not in common]
>>> wheat = [id for id in wheat if id not in common]
```


Text classification

Classifying Articles

#example8.py (continued)

```
>>> train_corn_ids = [train for train in corn if train.find('train')>-1]
>>> test_corn_ids = [test for test in corn if test.find('test')>-1]
>>> train_wheat_ids = [train for train in wheat if train.find('train')>-1]
>>> test_wheat_ids = [test for test in wheat if test.find('test')>-1]
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> train_corn_target = []  
>>> test_corn_target = []  
>>> train_wheat_target = []  
>>> test_wheat_target = []  
>>> train_corn = []  
>>> test_corn = []  
>>> train_wheat = []  
>>> test_wheat = []
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> def load_train_data():  
    train = []  
    train_target = []  
    for id in train_corn_ids:  
        train_corn_target.append(0)  
        train_corn.append(reuters.raw(id))  
    for id in train_wheat_ids:  
        train_wheat_target.append(1)  
        train_wheat.append(reuters.raw(id))  
    train = train_corn + train_wheat  
    train_target = train_corn_target +  
    train_wheat_target  
    return train, train_target
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> def load_test_data():  
    test = []  
    test_target = []  
    for id in test_corn_ids:  
        test_corn_target.append(0)  
        test_corn.append(reuters.raw(id))  
    for id in test_wheat_ids:  
        test_wheat_target.append(1)  
        test_wheat.append(reuters.raw(id))  
    test = test_corn + test_wheat  
    test_target = test_corn_target +  
    test_wheat_target  
    return test, test_target
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> train, train_target = load_train_data()
```

```
>>> test, test_target = load_test_data()
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> def preprocess_text(text):
    text = re.sub('[^A-Za-z]+', '', text)
    wordTokens = nltk.word_tokenize(text)
    wordTokens = [token.lower() for token in
wordTokens if len(token)>1]
    stops = stopwords.words("english")
    wordTokens = [token for token in wordTokens
if token not in stops]
    cleanedText = ' '.join(wordTokens)
    return cleanedText
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> train = [preprocess_text(doc) for doc in train]
```

```
>>> test = [preprocess_text(doc) for doc in test]
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> counts = vectorizer.fit_transform(np.hstack([train,test])).toarray()
>>> train_counts = counts[:len(train_counts),:]
>>> test_counts = counts[len(train_counts):,:]
```


Text classification

Classifying Articles

#example8.py (continued)

```
>>> nbm = GaussianNB()
>>> nbm.fit(train_counts, train_target)
>>> train_pred = nbm.predict(train_counts)
>>> test_pred = nbm.predict(test_counts)
>>> confusion_matrix(train_target, train_pred)
```

Text classification

Classifying Articles

	corn	wheat
corn	122	0
wheat	3	150

Text classification

Classifying Articles

#example8.py (continued)

```
>>> confusion_matrix(test_target, test_pred)
```

Text classification

Pre-Processing Data

	corn	wheat
corn	22	12
wheat	6	43

Text classification

Classifying Articles

#example8.py (continued)

```
>>> from keras.models import Sequential  
>>> from keras.layers import Dense, Dropout  
>>> from keras.utils import to_categorical
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> model = Sequential()
>>> model.add(Dense(32, activation='relu'))
>>> model.add(Dropout(0.30))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dropout(0.30))
>>> model.add(Dense(2, activation='softmax'))
```

Text classification

Classifying Articles

#example8.py (continued)

```
>>> model.compile(optimizer='adam',  
                  loss='binary_crossentropy',  
                  metrics=['acc'])  
  
>>> history = model.fit(train_counts,  
                        train_target,  
                        epochs=20, batch_size=32, shuffle=True,  
                        validation_split=0.20)
```

Text classification

Classifying Articles

Epoch 1/20

7/7 [=====] - 0s 16ms/step - loss: 0.7160 - acc: 0.4876

Epoch 2/20

7/7 [=====] - 0s 3ms/step - loss: 0.6330 - acc: 0.7215

Epoch 3/20

7/7 [=====] - 0s 3ms/step - loss: 0.5936 - acc: 0.8153

...

Epoch 20/20

7/7 [=====] - 0s 3ms/step - loss: 0.0426 - acc: 1.0000

Text classification

Classifying Articles

#example8.py (continued)

```
>>> confusion_matrix(test_target,  
                      np.argmax(test_pred,  
                                axis=1))
```

Text classification

Classifying Articles

	corn	wheat
corn	29	5
wheat	3	46

Sentiment Analysis

Sentiment Analysis

Overview

1. Scrape ECB speech.
2. Pre-process and tokenize text.
3. Apply sentiment model.

Sentiment Analysis

Dictionary-Based Methods: Example

$$\text{Positivity: } \frac{2}{135}, \text{ Negativity: } \frac{2}{135}, \text{ Net Positivity: } 0 = \frac{2}{135} - \frac{2}{135}$$

Economic activity in Sweden remains **strong** and inflation is close to the target of 2 per cent. **Uncertainty** abroad has increased but new information since the monetary policy decision in April has not led to any major revisions of the forecasts overall. With continued support from monetary policy, the conditions for inflation to remain close to the target in the period ahead are considered **good**. The Executive Board has decided to hold the repo rate unchanged at -0.25 per cent. The forecast for the repo rate is also unchanged and indicates that it will be increased again towards the end of the year or at the beginning of next year. However, the **risks** surrounding developments abroad can have a bearing on the prospects for Sweden, which emphasises the importance of proceeding cautiously with monetary policy.

Sentiment Analysis

Importing Modules

#example9.py

\$pip install pysentiment2

\$user python

```
>>> from urllib.request import urlopen
```

```
>>> from bs4 import BeautifulSoup
```

```
>>> import numpy as np
```

```
>>> import pysentiment2 as ps
```

```
>>> from sklearn.feature_extraction.text import CountVectorizer
```

Sentiment Analysis

Scrape ECB Speeches

#example9.py (continued)

```
>>> sp210429 = 'https://www.ecb.europa.eu/  
press/key/date/2021/html/ecb.sp210429~  
3f8606edca.en.html'
```

```
>>> sp210429 = BeautifulSoup(urlopen(sp210429).read())
```

Sentiment Analysis

Extract Paragraphs

#example9.py (continued)

```
>>> sp210429 = [p.text for p in sp210429.find_all('p')]
```

```
>>> print(sp210429[15])
```


Sentiment Analysis

Extract Paragraphs

Recently, a Task Force on Climate-related Financial Risks that operates under the Basel Committee on Banking Supervision looked into the effects of physical and transition risks on banks. ... The existing ones will do. However, the task force also concluded that we still need an enhanced toolbox that can better measure climate risks. This is why the ECB is encouraging banks to enhance both the quantity and the quality of their climate-related disclosures, and to become more transparent about their overall exposures to climate-related risks.

Sentiment Analysis

Tokenize Text and Compute Sentiment

#example9.py (continued)

```
>>> lm = ps.LM()
```

```
>>> tokens = [lm.tokenize(p) for p in paragraphs]
```

```
>>> sentiment = [lm.get_score(p)['Polarity'] for p in tokens]
```

```
>>> print(paragraphs[10])
```

```
>>> print(sentiment[10])
```

Sentiment Analysis

Sentiment Example

Net Positivity: -0.40

Transition risks, on the other hand, include the financial losses that can directly or indirectly result from the process of adjusting to a low-carbon and more environmentally sustainable economy. This adjustment could be triggered by legislation, such as carbon pricing or the banning of carbon-intensive activities. ... And preliminary results from our ongoing climate stress test show that without further policy action, companies will face substantially rising costs from extreme weather events. This will greatly raise their probability of default.

Topic Modeling

Topic Modeling

Data Preparation

#example9.py (continued)

```
>>> from sklearn.decomposition import LatentDirichletAllocation
>>> vectorizer = CountVectorizer()
>>> tfreq = vectorizer.fit_transform([' '.join(token) for token in tokens])
>>> feature_names = vectorizer.get_feature_names()
```

Topic Modeling

Model Training

#example9.py (continued)

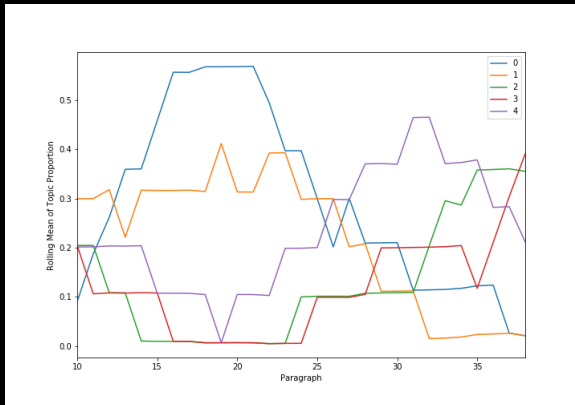
```
>>> lda = LatentDirichletAllocation(  
        n_components=5)
```

```
>>> props = lda.fit_transform(tfreq)
```

```
>>> wordDist = lda.components_
```

Topic Modeling

Topic Evolution



Topic Modeling

Model Training

#example9.py (continued)

```
>>> for i in range(5):  
    topics.append([feature_names[name] for name  
                  in wordDist[i].argsort()[-3:][:1]])  
  
>>> print(topic)
```


Topic Modeling

Topic Descriptions

```
[[ 'risk', 'climat', 'on'],  
 [ 'emiss', 'need', 'reduct'],  
 [ 'risk', 'assess', 'climat'],  
 [ 'climat', 'risk', 'loss'],  
 [ 'risk', 'climat', 'relat']]
```

2. Image Applications

Introduction to TensorFlow

Colab Tutorial

- ▶ Image Data Applications

3. Text Applications

Dynamic embedded topic models

Example

Google Colab: DETM



Transformer Models

Transformer Models

Challenges in S2S Modeling

- ▶ Dense neural networks require fixed-length inputs and outputs.
- ▶ Not suitable for S2S tasks like machine translation that require variable input and output lengths.
- ▶ LSTMs offer a solution to handle variable length sequences.

Transformer Models

LSTMs and Encoder-Decoder Architecture

- ▶ LSTMs treat input data as sequences, allowing for more parsimonious parameterization.
- ▶ Encoder-decoder architecture: Encoder maps input symbols to a latent vector, decoder maps latent vector to output symbols.
- ▶ Output sequences have variable lengths, model trained to output an end-of-sentence (EOS) token.

Transformer Models

Advancements in S2S Modeling

- ▶ LSTM-based models provided initial breakthroughs in high-quality machine translation and NLP tasks.
- ▶ Attention mechanism and Transformer model revolutionized sequence modeling in S2S contexts and NLP.
- ▶ Foundation for modern NLP tools in various applications.

Transformer Models

Embedding and Attention Mechanism

- ▶ Convert sequence of words to a sequence of embedding vectors.
- ▶ Attention mechanism determines related symbols without considering temporal ordering.
- ▶ Produces sequence of contextualized embeddings.
- ▶ Effective for various NLP tasks, including machine translation.

Transformer Model

Multi-headed Attention

- ▶ Innovation of the Transformer model.
- ▶ Attends to each subspace separately and in parallel.

Google Colab: Transformer Models

