Introduction to Text Mining and Natural Language Processing

Session 2: Project Design and Getting the Text

Hannes Mueller

January 2026

Barcelona School of Economics

Orientation

Course Overview

Part 1: Project Design and Getting the Text (Session 2)

Part 2: Text Mining Basics (Sessions 3 and 4)

Part 3: Dimensionality Reduction (Sessions 5 to 7)

In-class assignment in session 6

Part 4: Supervised Learning with Text (Sessions 8 and 9)

In-class assignment in session 9

Write me your term-paper ideas throughout.

Term paper presentations: 16th of March (feedback!)

This session

Course Overview

Project design

Getting the text

- documents (first notebook)
- scraping (TA session)
- API (not discussed)

Commercial break!!!!

Thinking about a Ph.D.?

Typically I am approached by several people wanting to do a Ph.D.

There are very few high quality programs that offer a mix of ML and economics/social sciences. But now we have the IDEA program:

- the M.Sc. has been shortened to 1 year
- it will be "a la carte", i.e. allow you to specialize
- allows for a bigger applied/ML leg
- Top-knotch Ph.D. program after: 1 DS student already in the Ph.D., 1 joining soon, 3 considering M.Sc.
- Christopher Rauh is director
- several other young faculty are joining the campus (ML yay!)

Link: `https://www.uabidea.eu/`

Project Design

Terminology

Documents D: unit of analysis (article or a paragraph or a tweet...)

Features, metadata X, y: date, name of the author, sentiment, topic, etc. (outside or generated from text)

Features, metadata sometimes used for supervision y.

Many methods exploit document boundaries - makes these boundaries a sort of X or y.

It is **crucial** to keep track of what metadata you need and where this should come from.

Index (Meta)Data

A crucial set of metadata X is what we often use as indexes.

Imagine we abuse notation and we write the corpus as a collection of articles $d \in 1, ..., D$.

This index d can then be used to write something like X_d or y_d for features extracted from d or labels related to d.

There are often units i or dates t associated with d.Think of these as $i(d)$ or $t(d)$, i.e. this metadata is a function of the document.

First step is to build functions that generate columns in a dataframe so that it looks as follows:

Table with Index (Meta)Data

date (t)	individual (i)	text (D)
2025-01-01	User001	"New Year's resolutions include ..."
2025-01-02	User001	"Analyzing trends in text data ..."
2025-01-03	User001	"Topic modeling helps uncover ..."
...	...	...
2025-01-01	User002	"Exploring the relationship ..."
2025-01-03	User002	"Text mining is often used for ..."
...	...	...
2025-01-03	User003	"Stemming is a way to unify text ..."
...	...	...

Using Index (Meta)Data to Aggregate

The question is then often how to aggregate up so that we get feature representations X_{it} that can be related to y_{it}.

Go back to previous table to think about missing values there...

Overview of the Text Pipeline

- Data collection - getting the D
- Pre-processing - making the D easier to use
- Analysis - set D in a relationship to some X or y through clustering, classification or exploration
- Communication - visualizations, regression tables, ROC curves...

Text Pipeline Design

- Before you start to design a pipeline - ask why?
  - Exploring a corpus
  - Explaining a corpus
  - Classification of documents into classes
  - Generating data for another task
- Purpose determines what metadata and method you want.
- It even determines of what you call a document D.
- Best explained with an example...

Example: sentiment analysis

- very common
- goal: track *sentiment expressed by someone regarding something*
- deconstruct this:
  - sentiment expressed by whom (news outlets, voter groups)?
  - sentiment regarding what (candidates, companies, products, concepts)?
  - other relevant dimensions (characteristics, time, space)?

Example: Tetlock (2007)

Goal: show that sentiment of the market affects stock market returns

Returns equation of the VAR used is:

$$Dow_t = \alpha_1 + \beta_1 \cdot L5(Dow_t) + \gamma_1 \cdot L5(BdNws_t) + \delta_1 \cdot L5(Vlm_t) + \lambda_1 \cdot Exog_{t-1} + \epsilon_{1t}$$

where:

- Dow_t is the Dow returns at time t.
- $L5(X_t)$ represents the five lags of variable X at time t.
- $BdNws_t$ is the media pessimism factor.
- Vlm_t is the detrended log of daily volume on the NYSE.
- $Exog_{t-1}$ represents exogenous controls in model.
- ϵ_{1t} is the error term.

Example: sentiment analysis

- For the regression Tetlock needed $BdNws_t$.

Example: sentiment analysis

- For the regression Tetlock needed $BdNws_t$.
- Uses *Wall Street Journal's (WSJ's) "Abreast of the Market" column* to measure sentiment.

Example: sentiment analysis

- For the regression Tetlock needed $BdNws_t$.
- Uses *Wall Street Journal's (WSJ's) "Abreast of the Market" column* to measure sentiment.
- We will talk about text mining method later in class.

Example: sentiment analysis

- For the regression Tetlock needed $BdNws_t$.
- Uses *Wall Street Journal's (WSJ's) "Abreast of the Market" column* to measure sentiment.
- We will talk about text mining method later in class.
- For now, note the implicit metadata usage by using a specific column as D_t to then extract sentiment features X_t.

Example: sentiment analysis

- For the regression Tetlock needed $BdNws_t$.
- Uses *Wall Street Journal's (WSJ's) "Abreast of the Market" column* to measure sentiment.
- We will talk about text mining method later in class.
- For now, note the implicit metadata usage by using a specific column as D_t to then extract sentiment features X_t.
- "Abreast of the Market" implies that D_t talks about the stock market.

Discuss with your neighbor(s)

Think of D, X and y, their indexes and their sources.

- You want to track what consumers think of specific products but you are worried some consumers are more negative than others.
- You want to track political polarization over time in a parliament using the attitude of different parties towards each other.

Getting the Text

Getting the Text

1. From Files: Reading directly from text files, PDFs, or other document formats.
2. Web Scraping: Extracting data from websites.
3. APIs: Accessing data provided by various online services.
4. OCR: Converting images or scanned documents into editable and searchable text.

Getting the Text through APIs

- APIs (Application Programming Interfaces) provide a structured way to request and retrieve data.
- Common APIs that provide text data:
  - Twitter API used to be great but now only Reddit is available (thank you Elon).
  - Google Books API - For accessing a wide variety of books data.
  - New York Times API - Provides access to various types of news and information.
- APIs usually return data in JSON or XML format, which can be easily parsed and analyzed.

Getting Text from Files

- Maneuvering and opening files:
  - `os` - Provides a way of using operating system-dependent functionality like maneuvering, reading or writing to a file.
  - `open` - Built-in method for reading and writing files.
  - often we can get metadata X, y at the file level through file names
- Getting the documents D and additional metadata X, y:
  - `RegEx`: we will see a lot of this today
  - `pandas`: storing D and X, y together

Spain text corpus

We will practice this now.

Open zip file first.

Getting the Text through Webscraping

- Python packages for webscraping:
  - `Requests` - Allows you to send HTTP/1.1 requests extremely easily.
  - `BeautifulSoup` - Easy to use tool for parsing HTML and XML documents.
  - `Selenium` - Allows your python code to interact with webpages - key for cracking hard nuts.
  - `Scrapy` - Advanced package - we will not discuss this in class.

Intro to scraping

We will now try to scrape the UN Security Council resolutions.

`https://www.un.org/securitycouncil/content/resolutions-0`

A paragraph in a UN resolution is D. What is the metadata X, to get if you want to build indexes to analyze the effect of resolutions on violence in a country?

Step 1: Inspection

Go to: `https://www.un.org/securitycouncil/content/resolutions-adopted-security-council-1946`

Open developer tools:

- Mac: Cmd+Alt+I
- Windows/Linux: Ctrl+Shift+I

Step 2: get URL to send us something

```
import requests
from bs4 import BeautifulSoup
import os

root = "https://www.un.org/securitycouncil/content/"
first_year = "resolutions-adopted-security-council-1946"

URL = root + first_year
print(URL)
headers = {'User-Agent': "Mozilla/5.0 (Windows NT 10.0;..."

response = requests.get(URL, headers=headers)
```

Step 3: enter data structure

```
#seeing it as a table with links and text
soup = BeautifulSoup(response.text, 'html.parser')

table = soup.find('table')
table_body = table.find('tbody')

#the row command identifies them through <tr>
rows = table_body.find_all('tr')
for row in rows:
    print(row)
    cols = row.find_all('td')
    print(cols)
```

TA Session: Hotel price analysis

Event shocks, hotel prices, and market segmentation

We will study how a large (future) event in Spanish cities affect hotel prices on Booking.

Core idea: scrape prices and hotel descriptions for (i) treated cities and (ii) one comparison city each, for two different weeks: event week (treated period) and non-event week (control period).

Dataset you will generate in the TA (one row per hotel $\times$ date): city, hotel, date, price, TreatCity (dummy), TreatPeriod (dummy), text

Where "text" is the hotel description and we will keep coming back to this as one a corpus.

DiD study of the effect on events on hotel prices

Fix notation: let p_{ht} be the price of hotel h in city $c(h)$ on date (or period) t.

Imagine we have only one event for now and one treatment and one control city. Call the city where the event takes place the $TreatCity_{c(h)}$ and the week at which the event took place $TreatPeriod_t$.

We can then write a simple difference-in-difference equation:

$$\begin{aligned} p_{ht} = \alpha + \theta_1 \,\mathrm{TreatCity}_{c(h)} + \theta_2 \,\mathrm{TreatPeriod}_t \\ + \tau \left(\mathrm{TreatCity}_{c(h)} \times \mathrm{TreatPeriod}_t\right) + \varepsilon_{ht}. \end{aligned} \tag{1}$$

where τ is the DiD estimate of the event effect on prices. The change in the treated city during the event week.

Call X_h the text-derived features (constructed from `text`).

Selection into the market

The set of observed hotels can change between weeks. Then the estimated price effect mixes: (i) true within-hotel price changes and (ii) composition changes (different hotels being listed). How we can separate this out using the text data? What would be the best way econometrically?

Advice on specification

Write the equation with more than one event. Strictly speaking we now have $TreatPeriod_{t(c(h))}$ but this is overkill. The basic, best specification with more cities can be written with fixed effects, i.e. like this.

$$\begin{aligned} p_{ht} = \alpha + \gamma_{c(h)} + \delta_t \\ + \tau \left(\text{TreatCity}_{c(h)} \times \text{TreatPeriod}_t \right) + \varepsilon_{ht}. \end{aligned} \tag{2}$$

How would you introduce the text feature X_h?

Heterogeneous effects by market segment

But perhaps there is also heterogeneity in the price elasticity of demand in different market segments.

Let s_h be a market-segment proxy for hotel h (constructed from the scraped `text` data).

How would you modify the equation above to test for different price elasticity in demand?