
BSE Summer School 2024
BSE
Barcelona School of Economics
SUMMER SCHOOL

Harnessing Language Models: Your Path to NLP Expert

Hannes Mueller (IAE (CSIC), BSE, CEPR)

July 2024

BSE Summer School

Introduction

Course overview

1. Overview of NLP methods
2. Language Modelling
3. Transformers
4. BERT training and applications
5. Prompt Engineering
6. Fine Tuning LLMs, Multi-Agent Pipelines

Overview of NLP methods

- Language and Cognition
- Strings and RegEx
- Text Mining Basics
  - Document-term matrix
  - Pre-processing
  - Vectorization
  - Tf-idf

Language and Cognition

Language and Cognition

Text is a written representation of language. The importance of language is hard to overstate:

- Wittgenstein argued that the world we see is defined and given meaning by the words we choose.
- The *language of thought theory* in philosophy believes that mental representation has a linguistic structure. Thoughts are "sentences in the head", meaning they take place within a mental language.
- Stephen Pinker hypothesizes an internal language that houses mental representations of concepts such as the meaning of words and sentences.

Language and Cognition

The Sapir–Whorf hypothesis in linguistics states that the grammatical structure of a mother language influences the way we perceive the world. This is contested by Firestone and Scholl (2016).

Boroditsky (2011) discusses some striking examples (spatial orientation, perception of time, memory).

Chen (2013, QJE) shows that the languages that grammatically associate the future and the present, foster future-oriented behavior.

Take-away

What does this mean for text mining? Some thoughts:

- Text mining is very powerful for understanding humans as language *fine-tunes* how we think.
- Therefore, with NLP we are re-constructing a latent structure which is both driven by and driving the way we think.
- **Your human prior and judgement is your most important tool in text mining applications.**
- You can see this in how people typically show performance by showing language output.
    - Turing-test - using language *like a human* is best possible performance.

Introduction to Strings and RegEx

What Are Strings?

- Strings are sequences of characters used to store text data.
- Enclosed in quotes: single (' '), double (" "), or triple ("' "' or """ """).
- Used for storing and manipulating textual information.

Characteristics of Python Strings

- **Immutability:** Once created, strings cannot be changed.
- **Indexing:** Access individual characters using indexes.
  - First character: `string[0]`
  - Last character: `string[-1]`

Useful String Operations

- **Concatenation:** Combining strings.
  - Example: `'Hello' + ' ' + 'World'`
- **Length:** Finding the number of characters.
  - Example: `len('Hello')` outputs 5
- **Slicing:** Extracting parts of the string.
  - Example: `'Hello'[1:4]` outputs 'ell'

More Useful String Operations

- **Query for content:** Operators work as in lists
  - Example: `"w" in string` gives `True` if fulfilled
- **Lowercasing:** Set all letters lower case.
  - Example: `string="WHAT you want"` and `string.lower()` outputs `what you want`
- **Find position:** Find the first position of an occurrence
  - Example: `string.find("want")` outputs 9

Key method: split()

- The `split()` method in Python is used to split a string into a list.
- By default, it splits the string by whitespace but can split using any other delimiters when specified.
- Syntax: `string.split(separator, maxsplit)`

Example:

```
text = "Hello, welcome to the world of Python"

words = text.split()

print(words)
```

This will output:

['Hello,', 'welcome', 'to', 'the', 'world', 'of', 'Python']

What is RegEx?

- Regular Expressions (RegEx) are sequences of characters that form a search pattern.
- Used for pattern matching, searching, and string manipulation.
- Fundamental in text processing and data extraction.
- There are a lot of special characters and functions. Best seen "in action" but no time.
- We will briefly introduce concepts so that you know what is going on when you see it.

Basic Syntax of RegEx

- **Literals:** Match exact characters. For example, `abc` matches "abc".
- **Metacharacters:** Symbols with special meanings. For example, `.` matches any character, `^` matches the start of a string.
- **Quantifiers:** Specify the number of occurrences. For example, `*` for 0 or more, `+` for 1 or more.

Groups and Capturing in RegEx

- Use parentheses to create groups in patterns.
- Groups can extract parts of the string for further processing.
- **Example 1:**
  - Pattern: `(\d{3})-(\d{2})-(\d{4})`
  - Matches: "123-45-6789" (e.g., a U.S. Social Security Number)
  - Captures three groups: "123", "45", "6789"
- **Example 2:**
  - Pattern: `\d{2}/\d{2}/\d{4}`
  - Matches: "12/02/2023"
  - Captures the three elements of the date.

Example:

Challenge: extract the email addresses out of the following text.

```
text = """
Hello,

My email is john.doe@gmail.com and my friend's email is
jane.doe@yahoo.com. We both like to communicate.

Best regards,
John
"""
```

Solution and Explanation

```
import re

email_regex = r'[\w\._-]+@[\w]+\.[\w]+\b'

emails = re.findall(email_regex, text)
```

This pattern will match email addresses that contain one or more word characters, period, and hyphen characters the @ and then one or more word chars, a period (.) followed by more word characters. It will only match the email address if it is followed by a word boundary (b). This will ensure that the pattern does not match the extra period at the end of the email address.

Text Mining Basics

Applications of text mining with NLP

- Sentiment Analysis: Determine sentiment in text for consumer opinions.
- Text Classification: Categorize text into predefined labels.
- Information Retrieval: Retrieve relevant information using search engines.
- Summarization/feature extraction: Create concise summaries/features of longer texts.
- Question Answering Systems: Provide specific answers to human queries.
- Language Translation: Convert text between languages.
- Chatbots and Virtual Assistants: Understand and respond to human language.

Text as vectors

- In essence text mining is converting text into vectors
- The art of NLP is to transform high-dimensional/complex data into manageable vectors.
- This means we need to capture semantic meaning of words, sentences, documents in a vector space.

Bag of words vs. Embeddings

It helps to divide NLP methods into two categories

- Bag of words model: each document is a vector of counts of terms (Hannes)
- Embeddings: each word, sentence, paragraph... is represented by a vector (Arnault)

All modern methods use the second model. But in practice we often still use the much simpler bag-of-words model. Depends a lot of application but don't use LLMs blindly!

Document Term Matrix

Introduction to Pre-processing

- Pre-processing is often done "before" we start to apply another method.
- But, in a way this is not really a separate step to the analysis.
- Very often the analysis is really determined in the collection and pre-processing stages.
- Before you get the data you need to have some idea of why you are mining.

Definitions

- A single observation in a textual database is called a *document*, $d \in \{1, ..., D\}$.

Definitions

- A single observation in a textual database is called a *document*, $d \in \{1, ..., D\}$.
- The set of documents that make up the dataset is called a *corpus* D.

Definitions

- A single observation in a textual database is called a *document*, $d \in \{1, ..., D\}$.
- The set of documents that make up the dataset is called a *corpus* D.
- We often have covariates associated with each document that are sometimes called *metadata*, X or y.

Definitions

- A single observation in a textual database is called a *document*, $d \in \{1, ..., D\}$.
- The set of documents that make up the dataset is called a *corpus* D.
- We often have covariates associated with each document that are sometimes called *metadata*, X or y.
- When processing the first step is called *tokenization*. Tokens include punctuation and can, depending on the method, even be parts of words.

Definitions

- A single observation in a textual database is called a *document*, $d \in \{1, ..., D\}$.
- The set of documents that make up the dataset is called a *corpus* D.
- We often have covariates associated with each document that are sometimes called *metadata*, X or y.
- When processing the first step is called *tokenization*. Tokens include punctuation and can, depending on the method, even be parts of words.
- The vocabulary V is spanned by terms v.

Our Workhorse: Bag of Words Model

- Bag of words model means we lose grammar.
- Order of words does not matter. *Documents*, d, can then be represented by a vector of counts of *terms*, v.
- Example of one document:
  - *The president of the United States is Joe Biden.*
  - Document vector, d after transformation in unigrams:

the	*president*	*of*	*united*	*states*	*is*	*joe*	*biden*
2	1	1	1	1	1	1	1

- Note difference between terms and *words* in this model.

The Document Term Matrix

A document-term matrix (DTM) is a mathematical representation of text data.

- rows represent documents
- columns represent terms
- a cells of the matrix contains the frequency or the weight of each term in each document
- call these counts $x_{d,v}$

Structure of DTM

	v_1	v_2	v_3	...	v_V
d_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	...	$x_{1,V}$
d_2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	...	$x_{2,V}$
d_3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	...	$x_{3,V}$
d_4	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	...	$x_{4,V}$
...	...	...	...	...	...
d_D	$x_{D,1}$	$x_{D,2}$	$x_{D,3}$	...	$x_{D,V}$

Where:

- d_1 is the first document
- v_1 is the first term (e.g. *beach*)

Example of Document-term Matrix

Having an example matrix will be useful. Imagine you have three documents. In our example a document is a sentence.

	the	*president*	*of*	*united*	*states*	*is*	*joe*	*biden*
d_1	2	1	1	1	1	1	1	1
d_2	0	1	0	0	0	1	1	1
d_3	1	0	0	1	1	0	0	0

In the example $D = 3$ and $V = 8$

N-grams

- Combinations of more than one word in a row are called N-grams. For example, bigrams, trigrams...
- Common bigrams in the example above will be *united_states* and *joe_biden*

Pre-processing

Pre-processing Intro

Steps to create a DTM:

- Tokenization: Splitting text into tokens.
- Normalization: Doing things to the text to reduce dimensionality
- After normalization we build the DTM:
  - Counting: Counting the frequency of each term in each document.
  - Dictionary: Build the dictionary of all terms.
  - Organizing: Structuring the data into the matrix.

The last three steps are essential - they are done by a package called the *vectorizer*.

Normalization

- Before we start the counting process we typically normalize.
- Steps we follow:
  - lemmatizing
  - remove punctuation
  - unify, e.g. convert to lower case
  - remove stopwords
  - stemming

Stemming/Lemmatizing

- Stemming: Deterministic algorithm for removing suffixes. Porter stemmer is popular.
  - In the Porter algorithm, argue, argued, argues, arguing, and argus reduce to the stem argu. Sometimes equivalence between tokens is misleading: ‘university’ and ‘universe’ stemmed to same form.
- Lemmatizing: Tag each token with its part of speech, then look up each (word, POS) pair in a dictionary to find linguistic root. E.g. ‘saw’ tagged as verb would be converted to ‘see’, ‘saw’ tagged as noun left unchanged.

Example: NYT - March 29, 1991. Libya

The exiled Prince Idris of Libya has said he will take control of a dissident Libyan paramilitary force that was originally trained by American intelligence advisers, and he has promised to order it into combat against Col. Muammar el-Qaddafi, the Libyan leader. The United States' two-year effort to destabilize Colonel Qaddafi ended in failure in December, when a Libyan-supplied guerrilla force came to power in Chad, where the original 600 commandos were based. The new Chad Government asked the United States to fly the Libyan dissidents out of the country, beginning a journey that has taken them to Nigeria, Zaire and finally Kenya. So far, no country has agreed to take them permanently. The 400 remaining commandos, who have been disarmed, were originally members of the Libyan Army captured by Chad in border fighting in 1988. They volunteered for the force as a way of escaping P.O.W. camps. "Having received pledges of allegiance from leaders of the force, Prince Idris has stepped in to assume responsibility for the troops' welfare," said a statement released in Rome by the royalist Libyan government in exile. It was overthrown in 1969.

Example: NYT - March 29, 1991. Libya (Stopwords)

the exiled prince idris of libya has said he will take control of a dissident libyan paramilitary force that was originally trained by american intelligence advisers, and he has promised to order it into combat against col. muammar el-qaddafi, the libyan leader. the united states' two-year effort to destabilize colonel qaddafi ended in failure in december, when a libyan-supplied guerrilla force came to power in chad, where the original 600 commandos were based. the new chad government asked the united states to fly the libyan dissidents out of the country, beginning a journey that has taken them to nigeria, zaire and finally kenya. so far, no country has agreed to take them permanently. the 400 remaining commandos, who have been disarmed, were originally members of the libyan army captured by chad in border fighting in 1988. they volunteered for the force as a way of escaping p.o.w. camps. "having received pledges of allegiance from leaders of the force, prince idris has stepped in to assume responsibility for the troops' welfare," said a statement released in rome by the royalist libyan government in exile. it was overthrown in 1969.

Example: NYT - March 29, 1991. Libya

exiled prince idris libya control dissident libyan paramilitary force originally trained american intelligence advisers, promised order combat col. muammar el-qaddafi, libyan leader. united states' two-year effort destabilize colonel qaddafi ended failure december, libyan-supplied guerrilla force came power chad, original 600 commandos based. new chad government asked united states fly libyan dissidents country, beginning journey taken nigeria, zaire finally kenya. far, country agreed permanently. 400 remaining commandos, disarmed, originally members libyan army captured chad border fighting 1988. volunteered force way escaping p.o.w. camps. "having received pledges allegiance leaders force, prince idris stepped assume responsibility troops' welfare," statement released rome royalist libyan government exile. overthrown 1969.

Example: NYT - March 29, 1991. Libya (Stemming)

exiled prince idris libya control dissident libyan paramilitary force originally trained american intelligence advisers, promised order combat col. muammar el-qaddafi, libyan leader. unit state two-year effort destabilize colonel qaddafi ended failure december, libyan-supplied guerrilla forces came power chad, origin 600 commando based. new chad government asked united states fly libyan dissidents country, beginning journey taken nigeria, zaire finally kenya. far, country agreed permanently. 400 remain commandos, disarmed, originally members libyan army captured chad border fighting 1988. volunteered force way escaping p.o.w. camps. "having received pledges allegiance leader force, prince idris steped assume responsibility troop's welfare," statement released rome royalist libyan governmeant exile. overthrown 1969.

Example: NYT - March 29, 1991. Libya

exil princ idri libya control dissid libyan paramilitari forc origin train american intellig advisers, promis order combat col. muammar el-qaddafi, libyan leader. unit state two-year effort destabil colonel qaddafi end failur december, libyan-suppli guerrilla forc came power chad, origin 600 commando based. new chad govern ask unit state fli libyan dissid country, begin journey taken nigeria, zair final kenya. far, countri agre permanently. 400 remain commandos, disarmed, origin member libyan armi captur chad border fight 1988. volunt forc way escap p.o.w. camps. "have receiv pledg allegi leader force, princ idri step assum respons troop welfare," statement releas rome royalist libyan govern exile. overthrown 1969.

Result of Pre-processing

- After pre-processing, each document is a finite list of terms (unigrams, bigrams...).
- Each document is represented by a vector of length V
- The $D \times V$ matrix D of all such counts is the document term matrix.
- Keep this in mind for the practice session.

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$
  - Keep this in mind also when aggregating documents!

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$
  - Keep this in mind also when aggregating documents!
- 2) some terms that are very common across the corpus.

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$
  - Keep this in mind also when aggregating documents!
- 2) some terms that are very common across the corpus.
- 3) sometimes we want to weigh specific content stronger

Some Thoughts about Term Frequencies

We can get a useful representation of a document through term frequencies. But there can be problems.

- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$
  - Keep this in mind also when aggregating documents!
- 2) some terms that are very common across the corpus.
- 3) sometimes we want to weigh specific content stronger
- We discuss tf-idf for 2) and dictionaries for 3).

Re-weighing vectors: tf-idf

Zipf's Law

- Zipf's Law is an empirical regularity for most natural languages:
  - Given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.
- Means that a few terms will have very large counts, many terms have small counts.

How to Re-weigh Counts

- Imagine you want to get to a vector representation of a corpus that controls for this.
- **Idea: penalize term counts $x_{d,v}$ of terms that appear in a lot of documents.**
- Documents are then characterized by the more *unique* terms.
- We will divide the term-frequency (tf) by the document-frequency (df).
- Result is the $tf - idf$

Document Frequency

- Let $x_{d,v}$ be the count of the term v in document d.
- We can express this count in the standard way as an absolute count $x_{d,v}$ or as

$$tf_{d,v} = \begin{cases} 0 & \textit{if } x_{d,v} = 0 \\ \log(x_{d,v}) + 1 & \textit{otherwise} \end{cases}$$

  which is also called the term frequency.

Inverse Document Frequency

- Now re-weigh terms that appear in a lot of documents.
- Sometimes this is even used in pre-processing text.
- $idf_v = \log(D/df_v)$ where
  - df_v is the number of documents that contain the term v
  - D is the number of documents
- Higher weight to words in fewer documents.
- Log dampens effect of weighting.
- A *smooth* alternative is $idf_v = \log\left(\frac{D}{df_v}\right) + 1$

Now Build the Tf-idf Score

- tf-idf score is situated at the document (d) - term (v) level.

$$tf\text{-}idf_{d,v} = \begin{cases} 0 & if\ x_{d,v} = 0 \\ [1 + \log(x_{d,v})] * \log\left(D/df_v\right) & otherwise \end{cases}$$

- sklearn does instead

  $tf\text{-}idf_{d,v} = x_{d,v} * (\log\left[(1 + D)\,/\,(1 + df_v)\right] + 1)$

Excursion: similarity

Similarity

- Call the vector representation of a document, d, $\vec{V}_d$, where the entries of the vector can be the term frequencies $x_{d,v}$, something based on a dictionary, or the $tfidf_{d,v}$.
- Imagine you want to measure whether two documents are similar using this vector.
- One way is to calculate the euclidian distance between two documents 1 and 2 is:

$$d_{1,2} = \sqrt{\sum_v (x_{d1,v} - x_{d2,v})^2}$$

- Why can this backfire badly?

Cosine similarity

Instead, what we typically do is to calculate the cosine similarity.

$$cs_{1,2} = \frac{\vec{V}_1 \cdot \vec{V}_2}{\left|\vec{V}_1\right| \left|\vec{V}_2\right|}$$

Think of this as a two-step procedure. First, if we normalize vectors by their length

$$\vec{v}_d = \frac{\vec{V}_d}{\left|\vec{V}_d\right|}$$

so they have length 1 and then we look at the dot product.

$$cs_{1,2} = \vec{v}_1 \cdot \vec{v}_2$$

Why the dot product?

- You might ask yourself - is the dot product a reasonable measure of similarity/distance?
- It tends to give a lot of weight to large values.
- Important coinciding counts will give a lot of weight to large entries in the vectors $\vec{V}$.
- Smaller entries will receive less attention.
- But there is an important fact "behind the scenes" that makes the dot product surprisingly reasonable as a similarity measure.

The dot product and Euclidean distance

Apart of the other anwers, there is indeed a relation between Euclidean distance $d(X, Y)$ and the inner product $\langle X, Y \rangle$ if the vectors $X, Y \in \mathbb{R}^N$ are normalized, that is $\langle X, X \rangle = \langle Y, Y \rangle = 1$, in particular

$$\frac{d(X, Y)^2}{2} = 1 - \langle X, Y \rangle.$$

This can be shown as

$$d(X, Y)^2 = \langle X - Y, X - Y \rangle = \langle X, X \rangle + \langle Y, Y \rangle - 2\langle X, Y \rangle = 2(1 - \langle X, Y \rangle)$$

Share Cite Follow

edited Dec 18, 2019 at 8:10

answered Nov 2, 2018 at 15:48

Pavel Prochazka

341 ■ 2 ▲ 7

Dictionary Methods

Dictionary methods

- Remember that the basic problem we are trying to solve is that the number of terms, V, is relatively high.
- Challenge in using text data for decision making is to therefore to reduce its dimensionality down from V.
- Dictionary methods are typically used in two cases:
  - 1) Human has a strong prior on what to use, i.e. specific keywords.
  - 2) Human knows what "kind" of text they want to capture, i.e. text which talks about finance.
- 1) is trivial. Key difficulty is how to come up with a dictionary for 2). We will do mostly literature discussion today.

Dictionary based method

- Dictionary is a list of terms. Call this set of terms $\mathfrak{D}$.
- Boolean search provides a count of the number of times specific terms appear in a document, $x_{d,v}$.
- Important advantage: often you can query a database that you don't own to give you $x_{d,v}$
- In most methods, this is then aggregated to deliver some sort of score or index at document level (which is then typically further aggregated).
- In applications both the dictionaries vary widely and the ways to score documents vary.

Aggregation methods

- Aggregation is as important as the dictionaries themselves.
- Simple sum: score d with $x_d = \sum_{v \in \mathfrak{D}} x_{d,v}$
- Why do we typically not use (aggregates of) these raw counts?
- Normalized sum: score d with $s_d = \sum_{v \in \mathfrak{D}} x_{d,v} / \sum_v x_{d,v}$.
- Indicator: score d with $I\left(\sum_{v \in \mathfrak{D}} x_{d,v} > 0\right)$
- Interaction: score d with $\prod_{v \in \mathfrak{D}} I\left(x_{d,v} > 0\right)$

Examples of dictionary based methods I

- Tetlock (2007): Wall Street Journal articles are used with dictionaries to generate measures of "positive" and "negative" sentiment.
- Gentzkow and Shapiro (2010): Use party labels on speeches to build dictionary with $\chi^2 - stats$
- Chadefaux (2014): uses dictionary of words to forecast internal armed conflict
- Baker et al (2016): use counts of specific words to generate measure of economic uncertainty.

Examples of Dictionary Based Methods II

- Hassan et al (2019): build a dictionary from textbooks - terms which are in politics textbook but *not* in a finance textbook
- Böhme, Gröger and Stöhr (2020): predict migration intensions using dictionaries entered in Google trends
- Lewandowsky, Jetter and Ecker (2021) Trump's diversionary tweets. A simple dictionary using priors (CJI)
- Garcia-Uribe et al (2022): build a dictionary of divisive political issues from discussions in history books
- Modern event coders like GDELT and ICEWS use huge dictionaries of actors, sentiment etc.

Example 1: Tetlock (2007)

Sentiment Analysis

- Uses *Wall Street Journal's (WSJ's) "Abreast of the Market" column* to measure sentiment.
- Sentiment analysis is GI categories from the Harvard psychosocial dictionary.
- Sentiment analysis is really easy as packages (textblob) come with pre-defined dictionaries.

Example: self-made dictionary with NLTK

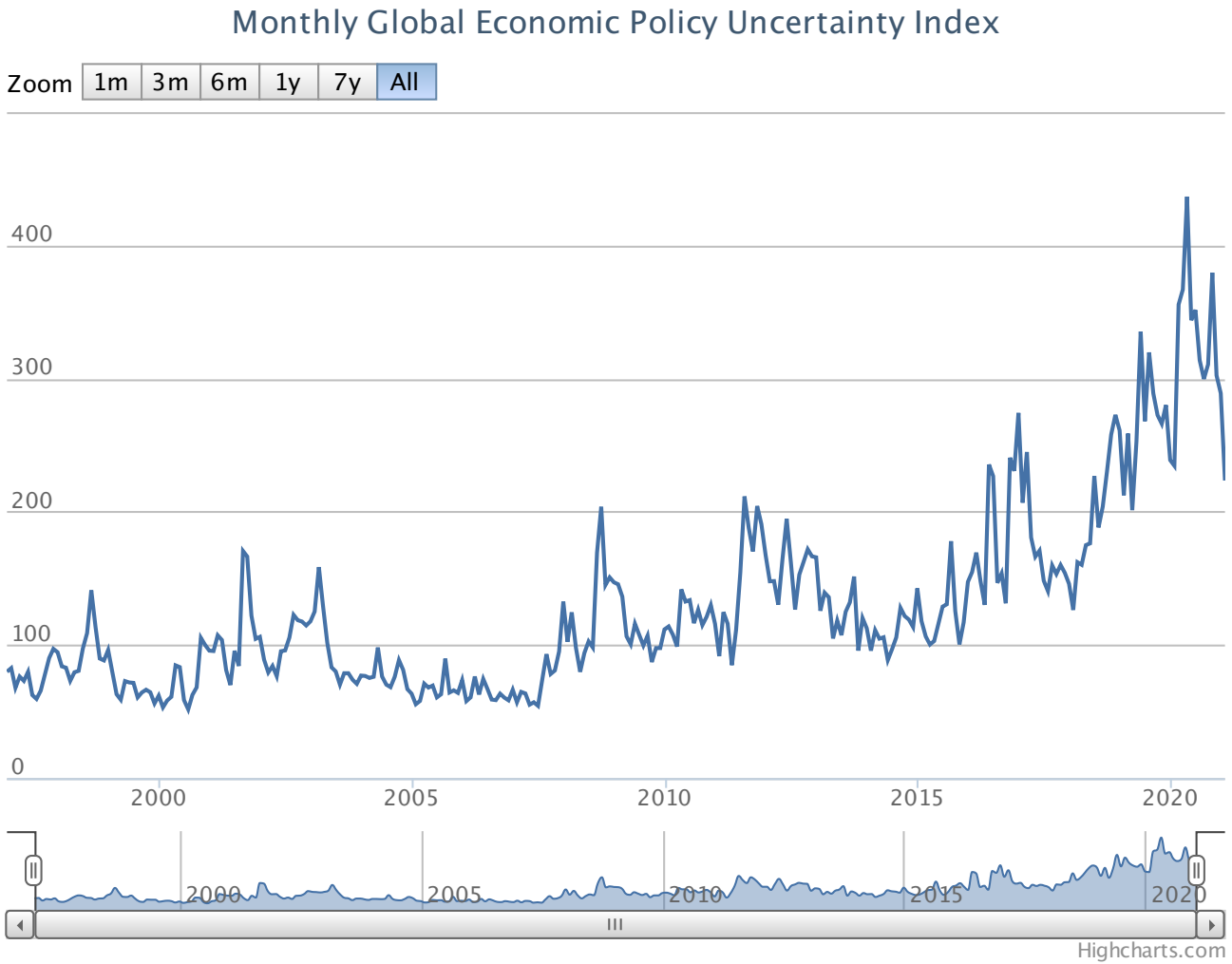
```
import nltk
from nltk.tokenize import word_tokenize

# Sentiment dictionary:
dict = {'happy': 1, 'sad': -1, 'amazing': 2, 'terrible': -2}

text = "It is a beautiful day but I am sad"
tokens = word_tokenize(text)

score = sum(sentiment_dict.get(word, 0) for word in tokens)

print(f"Sentiment score: {score}")
```

Example 2: Baker, Bloom and Davis (2016)

Combining dictionaries

- Baker, Bloom, and Davis (2016) use dictionaries to produce the data behind `http://www.policyuncertainty.com/`
- BBD are interested in measuring economic policy uncertainty.
- Uncertainty about policies might be a key driver of economic activity.

EPU


Monthly Global Economic Policy Uncertainty Index
Zoom
1m
3m
6m
1y
7y
All
400
300
200
100
0
2000
2005
2010
2015
2020
Highcharts.com

Research Goal

- Capture uncertainty about
  - who will make economic policy decisions
  - what economic policy actions will be undertaken and when
  - the economic effects of policy actions (or inaction)
- Including uncertainties related to the economic ramifications of "noneconomic" policy matters, for example, military actions

Method Overview

- BBD create an index based on Boolean searches of newspaper articles from major newspapers.
- For each paper they submit the following queries (separately):
  - 1. (E) Article contains "economic" OR "economy"
  - 2. (P) Article contains "congress" OR "deficit" OR "federal reserve" OR "legislation" OR "regulation" OR "white house"
  - 3. (U) Article contains "uncertain" OR "uncertainty"
- **How would you combine these indicators?**

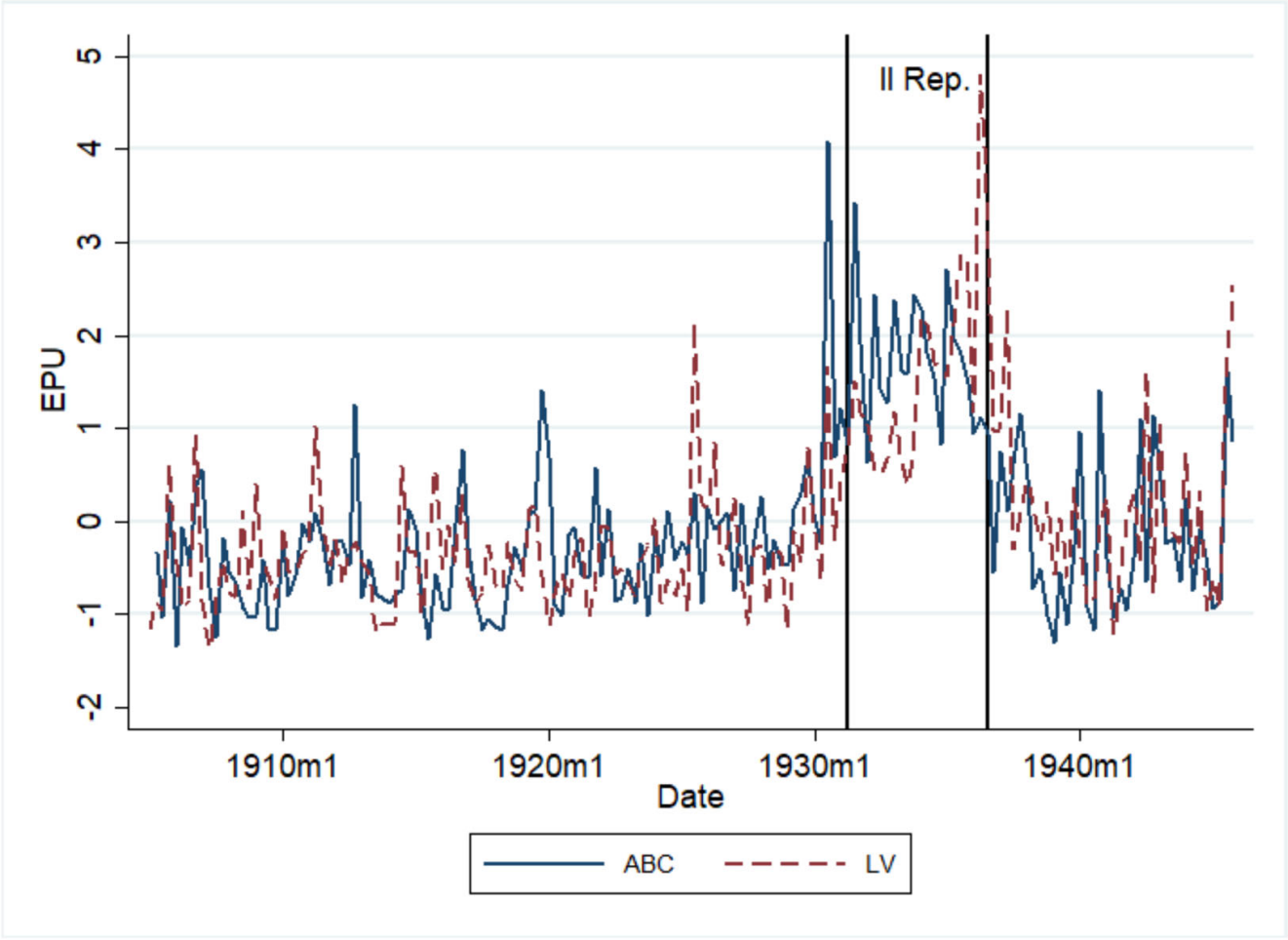
Example 3: Garcia-Uribe et al (2024)

Use Tf-idf to construct dictionary

- Garcia-Uribe et al (2022) Economic Uncertainty and Divisive Politics: Evidence from the "dos Españas"
- Construct the EPU index for Spain in 1905-1945.
- Historic data does not provide article split: we simulate this.
- Find shift upward before the civil war broke out.
- Question: did this correlate with political tensions?

EPU in Spain before the Civil War

Figure 1: EPU Index for Spain: 1905-1945



Note: The EPU index is calculated using the procedure described in Appendix B. Quarterly data used. Sample period: 1905–1945.

Building the Dictionary

- We use "supervision" through the work that historians have done.

Building the Dictionary

- We use "supervision" through the work that historians have done.
- Idea: exploit accounts of pre-civil war period talk about four divisive issues - socioeconomic conflict, regional separatism, the power of the military, the role of the church/religious education

Building the Dictionary

- We use "supervision" through the work that historians have done.
- Idea: exploit accounts of pre-civil war period talk about four divisive issues - socioeconomic conflict, regional separatism, the power of the military, the role of the church/religious education
- We copy the text of the description of these issues into four different documents and then calculate the tf-idf on the terms in the four documents:

Building the Dictionary

- We use "supervision" through the work that historians have done.
- Idea: exploit accounts of pre-civil war period talk about four divisive issues - socioeconomic conflict, regional separatism, the power of the military, the role of the church/religious education
- We copy the text of the description of these issues into four different documents and then calculate the tf-idf on the terms in the four documents:
    - **What will happen?**

Building the Dictionary

- We use "supervision" through the work that historians have done.
- Idea: exploit accounts of pre-civil war period talk about four divisive issues - socioeconomic conflict, regional separatism, the power of the military, the role of the church/religious education
- We copy the text of the description of these issues into four different documents and then calculate the tf-idf on the terms in the four documents:
    - **What will happen?**
- We then simply take the top terms as our dictionaries for the four issues.

Resulting Dictionary

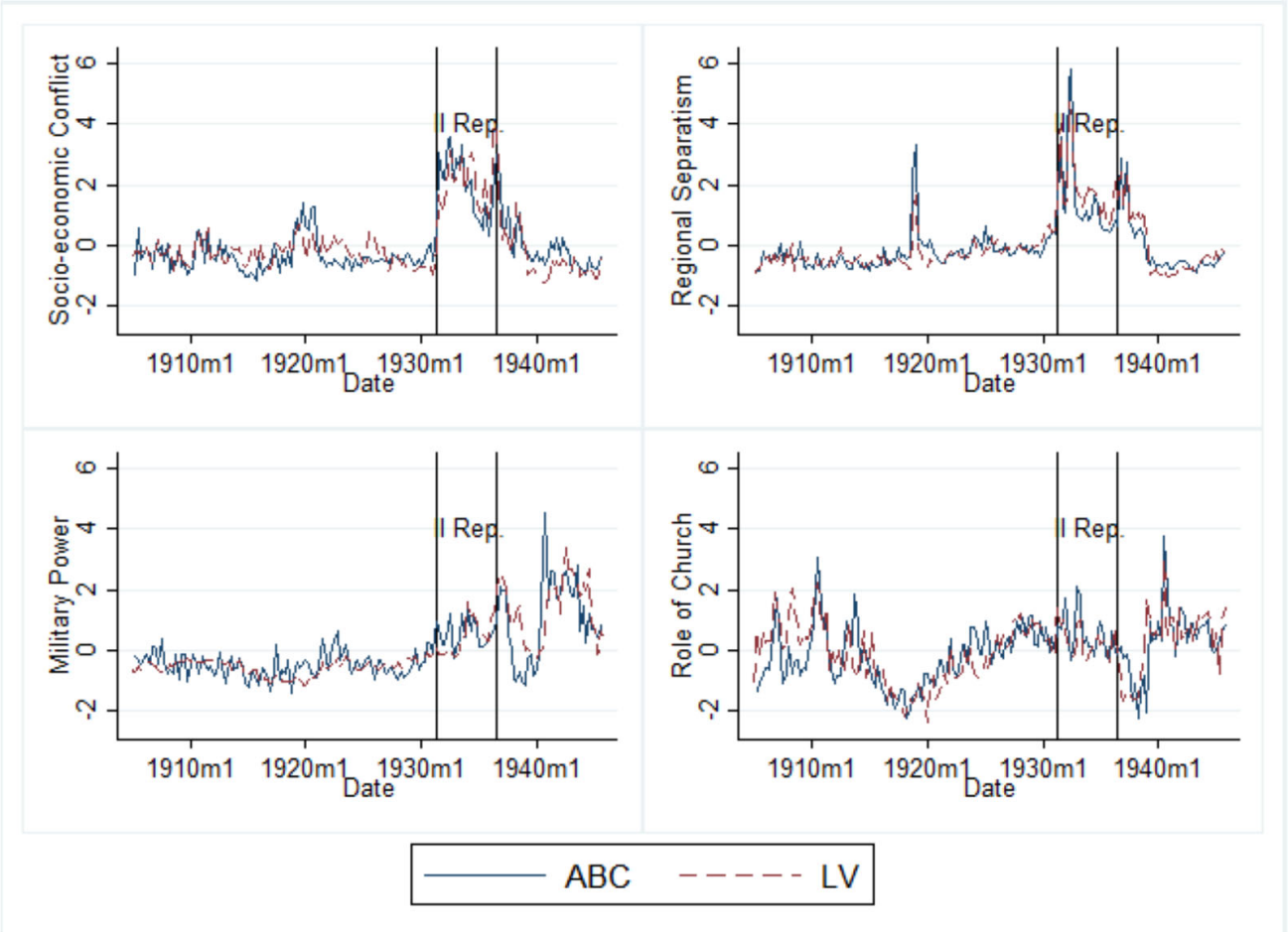
Table 1: Dictionaries of Four Divisive Issues

Socio-Economic Conflict	Regional Separatism	Role of Church	Power of Military
tierras	**estatuto**	iglesia	militar
trabajo	cataluña	**católicos**	ejército
reforma agraria	proyecto	enseñanza	oficiales
reforma	**vasco**	poltica	militares
agraria	catalana	**católica**	guerra
campesinos	**proyecto estatuto**	órdenes	generales
casas	**autonomía**	entonces	**reforma militar**
jurados mixtos	**federal**	**cardenal**	**ascensos**
jurados	**integral**	parte	**orden público**
mixtos	macià	hizo	civil
viejas	**estatuto cataluña**	**conventos**	orden
casas viejas	**república catalana**	segura	reforma
grandes	catalán	madrid	público
largo	barcelona	**iglesia católica**	guardia
largo caballero	izquierda	españoles	decreto
caballero	**catalanes**	**religiosos**	parte
extremadura	consejo	**religiosas**	**mantuvo**
huelgas	**referéndum**	**edificios**	fuerzas
instituto	generalidad	civil	cuerpo
social	navarra	régimen	servicio
contratos	vascos	española	**retiro**
jornaleros	mayoría	intelectuales	seis
propietarios	aprobado	intelectual	seis meses
fincas	nuevo	creía	**armadas**
parte	noviembre	**católico**	**fuerzas armadas**
salarios	regiones	**pastoral**	militar manuel
contratos trabajo	país vasco	órdenes religiosas	**profesional**
hectáreas	votos	marañón	**jurisdicción militar**
obreros	diputados	maestros	**oficialidad**
ministro trabajo	francesc	colegios	armas

Note: The words under each issue are the 30 initial words from the tf-idf model. The bold-faced words are the ones finally used for the indices after removing common and period-specific words. See Appendix C for details. See Table A2 for an English translation.

Resulting Dictionary

Figure 2: Four Divisive Issues



ote: The four indices are calculated with a tf-idf model. See Appendix C for details. Quarterly data used. Sample period: 905–1945.

Recap

- We now have a good idea of vector representations of documents.
- We have seen three types:
  - Absolute and relative term frequency counts
  - Tf-idf counts
  - Dictionary-based weights
- Important: it helps a lot to think about this carefully in your application. Try your own version!
- Arnault will move to vector representations of terms.

Practice session preparation

In the practice session we will:

- Open a corpus D and explore it
  - corpus data
  - labels y that indicate type of article (4 classes)
- Go through pre-processing the corpus
- Talk about different document-term matrixes
- Implement the method in Garcia-Uribe et al (2024)
  - sum document vectors d if they are from same class
  - calculate tf-idf score across 4 classes to generate dictionary
- Please prepare by going through preparation notebook before.