Text As Data - Lecture 2

Analysis

Machine Learning and Big Data

Vincent Bagilet

2025-12-03

Summary from last class

NLP in Economics

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

Overal Approach

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

Pre-processing

- Capitalization
- Punctuation
- Stop words
- Stemming and lemmatizing

Transforming your data

From text to numbers

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

Occurrences and counts

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

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

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

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

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

Matrices

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

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

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

Tf-idf implementation

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

assemblee_2018 = pd.read_csv("../../../data/assemblee_2018.csv")

assemblee_small = assemblee_2018.head(20)
assemblee_small = assemblee_small[['speaker_name', 'date', 'text']]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(assemblee_small['text'])
X
```

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

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

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

Dimensionality reduction

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

Usefulness

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

Application of similarity analysis

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

The paper in one line

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

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

On your own data set

On your own dataset

What similarity question could you ask on you own data?

Dictionary-based methods

Overview

- Methods that use **predefined lists** of words or phrases for analysis
- Applications:
  - Count (co-)occurrences of certain words/dictionaries
  - Compute metrics based on values associated to each gram (*eg* sentiment score)
  - Classify text into categories defined by dictionaries
- Match your text data with the dictionary, typically with RegEx

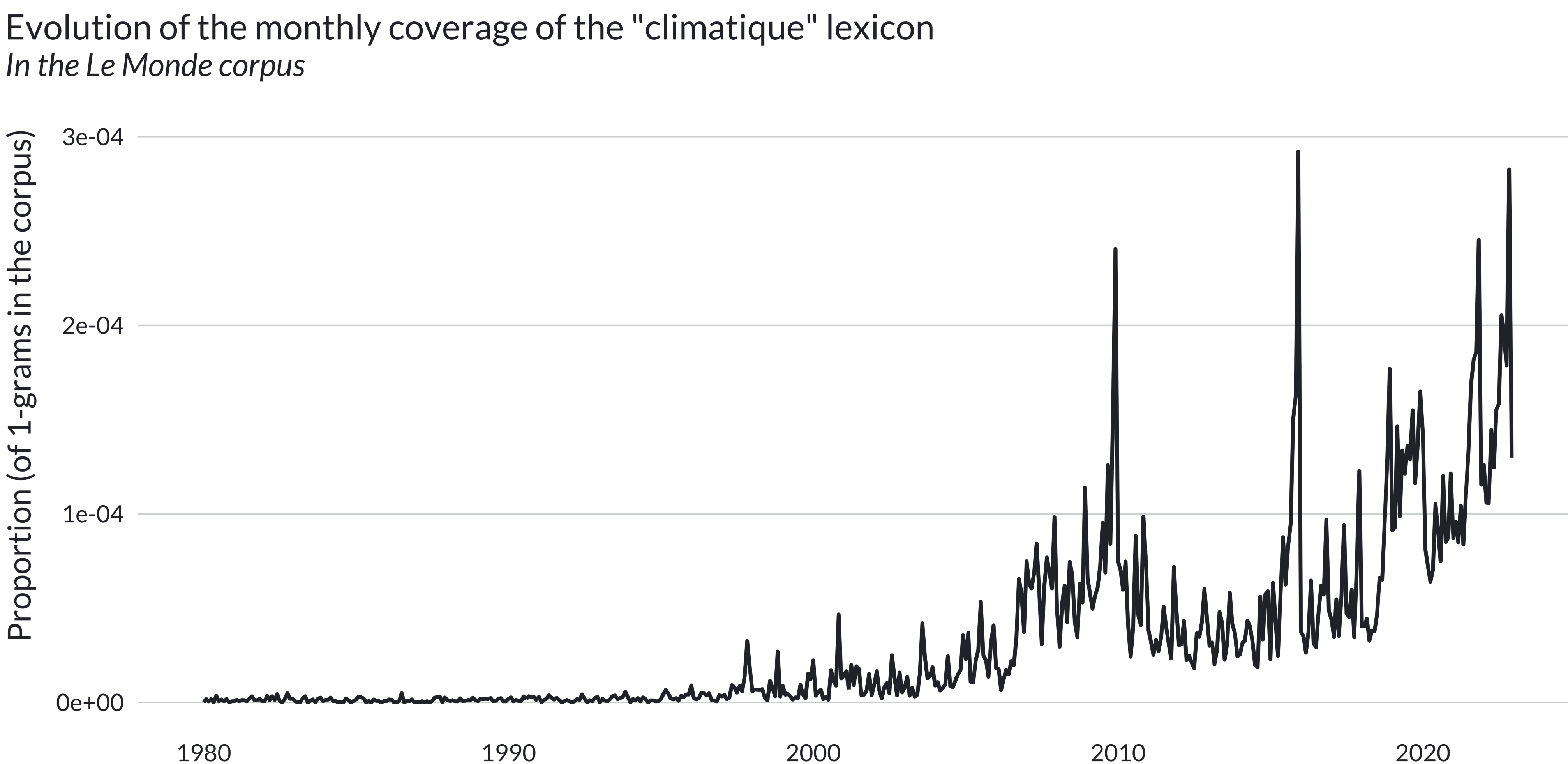
Occurrences

- Tools that allow some similar analyses?
  - Google Trends
  - Google ngrams
  - Gallicagram API and wrapper R, `rallicagram`

On your own dataset

- What question could you ask with this?
- When would you use this as the end-goal of your analysis?
- What information do you loose?

Example


Evolution of the monthly coverage of the "climatique" lexicon
In the Le Monde corpus
Proportion (of 1-grams in the corpus)
3e-04
2e-04
1e-04
0e+00
1980
1990
2000
2010
2020

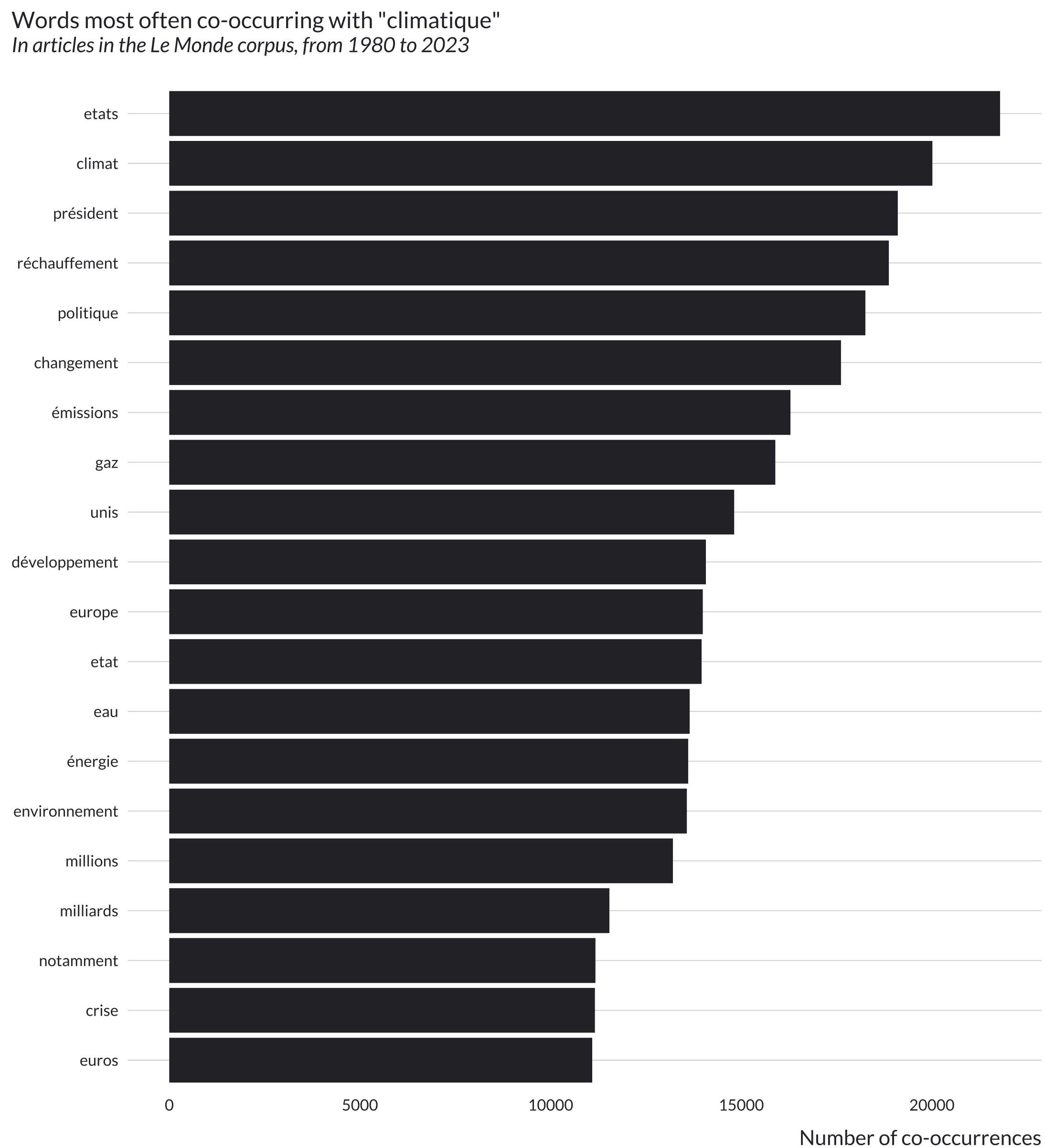
Co-occurrences

- Terms that appear together. Where/at which level?
  - In a document
  - In a sentence
  - In a n-gram

On your own dataset

- What question could you ask with occurrences?
- When would you use occurrences as the end-goal of your analysis?
- What information do you loose?

Example of co-occurrences



Building a dictonary

- Defining your dictionary:
  - Using external sources
  - Using domain expertise (pre-existing topic specific dictionaries)
  - Choosing terms depending on how well they predict human-annotation
- Use of **pre-existing dictionaries**:
  - AFINN, NRC, bing for sentiment analysis
  - WordNet (synonyms, antonyms, etc)
  - LIWC (Linguistic Inquiry and Word Counts): words sorted into categories (*eg* anger, familly, etc)
- **Expanding** dictionaries:
  - With words that are close by in an embedding space
  - Use LLMs

Pros and Cons

- **Advantages**:
  - Interpretable
  - Straightforward to implement (once dictonary built)
  - No training data
- **Limitations**:
  - May miss some terms, etc
  - Cannot take polysemy into account
  - May not be scalable

Application of a dictionary method

- Hassan et al. (2019): *Firm-Level Political Risk: Measurement and Effects*

> **The paper in one line**
>
> - "Build a measure of political risk faced by individual US firms: the share of their quarterly earnings conference calls devoted to political risk"

- Approach to quantify political risk faced by a firm:
    1. **Identify political terms**: bigrams that are in political science textbooks but not in general financial texts
    2. Count the number of co-occurrences of these bigrams with a dictionary for risk and uncertainty
    3. Compute the share of the earning calls dedicated to political risks

Sentiment analysis

- **Goal**:
  - Get to the "**tone**" dimension of a document (positive, negative, neutral)
- Sentiment score for a document: average of scores of all grams

word	value
clean	2
cleaner	2
withdrawal	-3
petrified	-2
rejecting	-1
dear	2
inspiration	2
embittered	-2

Sentiment analysis

- How would you implement it on your data set? Which question?

- Implementation with TextBlob (that upweights adjectives for instance)

```
from textblob import TextBlob

sentence_sentiment =  "Beautiful is better than ugly."
sent = TextBlob(sentence_sentiment)
print(sent.sentiment.polarity)
```

```
0.21666666666666667
```

- Can also implement non-dictionary based sentiment analyses (*eg* with transformers)

> ⚠ **Warning**
>
> Such analyses (and many other NLP analyses) can be biased!
>
> *eg* “Let’s go get Italian VS Mexican food”

Application of a sentiment analysis

- Almond, Du, and Papp (2022): *Favourability towards natural gas relates to funding source of university energy centres*

The paper in one line

- Use lexicon and rule-based sentiment analysis to show that fossil-funded energy centres from universities are more favourable in their reports towards natural gas than towards renewable energy

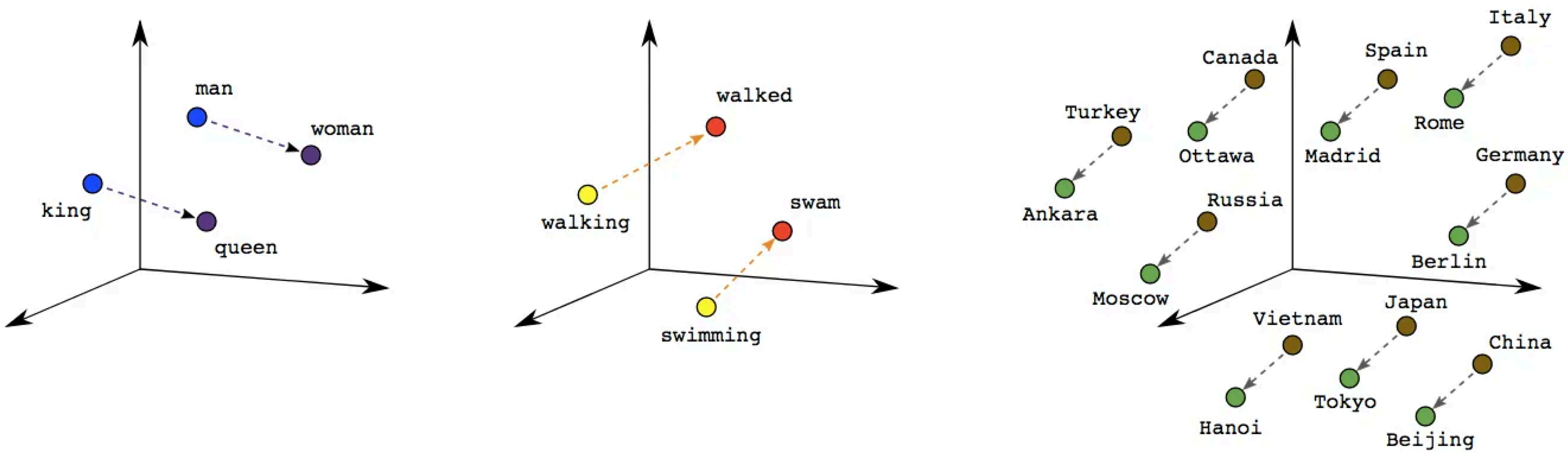
- Approach to sentiment analysis:
  1. Use Vader dictionary
  2. Regress the score of each sentence on a dummy for whether the sentence includes keywords related a fuels type with report FEs

Word Embeddings

General idea

- Project documents into a **vector space**
- Building **dense** matrices, *ie* non-sparse
- Uses the **context**
    - Words that often occur jointly will be close in the embedding space
    - *eg* we probably want *pen* and *pencil* to be relatively close in the space
- Captures **relations** between words
- "Classic" embeddings do not take polysemy into account: collapse each word to one vector
- We are going to talk about more complex, deep learning based, embeddings later in this session

Visual intuition



Male-Female Verb Tense Country-Capital

Pre-trained embeddings

- **Ready to use**, capture relationships between words in common language
- Trained on large corpora (Common Crawl, Wikipedia)
- Different **algorithms**: FastText (Facebook), word2vec (Google), GloVe (Stanford)
- Computation **methods**:
  - CBOW (Continuous Bag Of Words), Skip-gram
  - Basically, compute words that co-occur in a k-words window

Train your own embedding

- Will capture the relationships between words **in your corpus**
- Take **specificities** of your corpus into account
- **Need a lot of text data** to train properly train your own embeddings
- Use the algorithms (FastText, word2vec, GloVe) and methods (CBOW, skipgram) described above to train you embedding
- Specify:
  - Desired dimension of the embedding space
  - Number of words in the training window

Applications

- Find **nearest-neighbors** of words
  - Can be used to build dictionaries
  - Describe relationships between words
- Identify **clusters** of words
  - Can be used for topic modeling (k-nearest neighbors in the embedding space)
- Identify specific **dimension**
  - *eg* difference between “man” and “woman” will give you some sort of “gender” direction (see next example)

On your own dataset

What type of analysis could you do with embeddings?

Application of WEAT

- Ash, Chen, and Ornaghi (2024): *Gender Attitudes in the Judiciary: Evidence from US Circuit Courts*

> **The paper in one line**
>
> - Use judge-specific embedding to determine judges gender attitudes and show that it correlates with gendered-biased behavior

- Approach to compute gender attitudes:
  1. Train judge-specific embeddings (wih GloVe algorithm)
  2. Identify a gender dimension (vector) in the space by taking the difference between words annotated as “male” or “female” in the LIWC Dictionary
  3. Same for a stereotypical career-family vector
  4. Compute the cosine similarity between these vectors = gender attitudes

Machine Learning

Supervised ML

- **Classification** model: predict a class label or group membership
- **Regression** model: predict a numeric or continuous value
- We learned how to transform our data from **unstructured to structured** format
- We can now **apply what you learn in the rest of the class** to this text data

  $\longrightarrow$ will not go into more details here

On your own dataset

- Examples of applications?
- How to build validation sets?

Unsupervised ML

- Use **unlabeled data** ≠ many of the algorithms discussed in class
- Do not try to fit to make the model predict a "ground truth"
- For text data, unsupervised ML is mostly used for **topic models** :
    - They infer latent topics in the corpus
    - **Supervised**: define topics, write algorithms to predict which category they belong to
    - **Semi-supervised**: give anchor-words (*eg* CorEx)
    - **Unsupervised**: let the algorithm discover topics by itself (*eg* LDA)

Latent Dirichlet Allocation (LDA)

- Every document is a mixture of topics
  - *eg* doc 1 is 20% topic A, 80% topic B
- Every topic is a mixture of words
  - *eg* topic A is composed of words “banana”, “apple”, etc
- LDA estimate both at the same time, it is probabilistic ML
- Only one parameter to specify: the number of topics (k)

Example LDA

```python
from sklearn.decomposition import LatentDirichletAllocation
from wordcloud import WordCloud
import matplotlib.pyplot as plt

vocab = vectorizer.get_feature_names_out()
lda = LatentDirichletAllocation(n_components=5, random_state=12)
lda.fit(X) #X is the tf-idf matrix from earlier
```

▼ LatentDirichletAllocation

```
LatentDirichletAllocation(n_components=5, random_state=12)
```

```
(-0.5, 799.5, 599.5, -0.5)
```

Structural Topic Model (STM)

- **LDA + Metadata**
- Includes contextual information by:
  - Making topic prevalence vary with metadata, *eg* left wing parties talk more about inequality than right wing parties
  - Topic content can vary with metadata, *eg* left wing parties talk more about education inequality than right wing parties

Application of a topic model

- Noailly et al. (2024): *Heard the news? Environmental policy and clean investments*

The paper in one line

- Develop a news index of US env and climate policy and look at link with actual regulations and financial investments

- Approach to topic modelling
  - Use LDA
  - Use a supervised approach (with SVM)

Deep Learning

BERT-like models

- Bidirectional Encoder Representation from Transformers
- Introduced by Google in 2018
- Masked Language Model: trained by trying to discover words that are randomly masked
- Good at text **classification**, **answering questions**
- Mainly focused on **understanding** language

GPT-like models

- Generative Pretrained Transformer
- Introduced by OpenAI in 2018
- Good at **text generation**, having **conversations**
- Autoregressive Language Modeling: trained to predict the next word
- Mainly focus on **generating** language

Using LLMs in economics text analysis

- **Very good** at certain tasks
- Easily accessible but might be **expensive** to run
- May not be a good option for sensitive private data (need to share data)
- Can use models installed **locally**
- Computationally intensive
- Not that interpretable (black-boxy) but can be validated against hand-labelled data
- May not be very robust to prompt variations + **non-replicable**

Application of BERT

- Moreno-Medina et al. (2025): *Officer-Involved: The Media Language of Police Killings*

> **The paper in one line**
>
> The forms of language used by the media to report on police killings affects how the readers hold them morally responsible

- Uses of BERT to identify:
  - All words that reference the same individual using Span-BERT; it **clusters** all the tokens that describe the same entity
  - Who did what to whom in sentences about the shooting using another BERT-like model to **annotate semantic roles**
- Then, based on that identify structures that appear in each sentence: an active-voice verb, a passive-voice verb, a nominalization, no agent, or an intransitive verb.

Conclusion

Summary of the lecture

- Need to **transform** unstructured text data to numeric format (typically to sequences of grams or matrices)
- Embeddings and dense vector representations allow to capture relations between words
- These representations allow to apply:
  - Direct analyses on these representations, *eg* similarity analyses via cosine-similarity
  - **Supervised** ML models learned in the rest of the class
  - **Unsupervised** ML algorithms such as topic modeling (*eg* LDA)
- Recent developments in NLP allow to compute context-specific representations and to implement more complex analyses

Summary of the class

- Typically, NLP allows to **build new metrics to plug into econometrics models**
- There are many steps involved in text analysis
- **Gathering data** may be one of the most time consuming ones
- Once we have transformed our text data, we can apply other typical ML tools to these representations
- There are currently tons of developments in NLP, hard to keep up
- Nowadays, we can basically do anything we want on text data $\Rightarrow$ importance of **identifying good research questions**

Back to your own question

On your own dataset

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

References

Almond, Douglas, Xinming Du, and Anna Papp. 2022. “Favourability Towards Natural Gas Relates to Funding Source of University Energy Centres.” *Nature Climate Change* 12 (12): 1122–28. https://doi.org/10.1038/s41558-022-01521-3.

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

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

Hassan, Tarek A, Stephan Hollander, Laurence van Lent, and Ahmed Tahoun. 2019. “Firm-Level Political Risk: Measurement and Effects.” *The Quarterly Journal of Economics* 134 (4): 2135–2202. https://doi.org/10.1093/qje/qjz021.

Moreno-Medina, Jonathan, Aurélie Ouss, Patrick Bayer, and Bocar A Ba. 2025. “Officer-Involved: The Media Language of Police Killings.” *The Quarterly Journal of Economics* 140 (2): 1525–80. https://doi.org/10.1093/qje/qjaf004.

Noailly, Joëlle, Laura Nowzohour, Matthias van den Heuvel, and Ireneu Pla. 2024. “Heard the News? Environmental Policy and Clean Investments.” *Journal of Public Economics* 238 (October): 105190. https://doi.org/10.1016/j.jpubeco.2024.105190.