Big Data & Automated Content Analysis
Week 13 – Wednesday: »Last Questions«

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

11 May 2021

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

Today

Looking back

- Putting the pieces together
- A good workflow

Looking forward

- Techniqes we did not cover
- Neural Networks

Looking back

Looking back

Putting the pieces together

Computational Social Science

"It is an approach to social inquiry defined by (1) the use of large, complex datasets, often—though not always— measured in terabytes or petabytes; (2) the frequent involvement of "naturally occurring" social and digital media sources and other electronic databases; (3) the use of computational or algorithmic solutions to generate patterns and inferences from these data; and (4) the applicability to social theory in a variety of domains from the study of mass opinion to public health, from examinations of political events to social movements"

Shah, D. V., Cappella, J. N., & Neuman, W. R. (2015). Big Data, digital media, and computational social science: Possibilities and perils. T*he ANNALS of the American Academy of Political and Social Science*, *659*(1), 6–13. doi:10.1177/0002716215572084

Computational Social Science

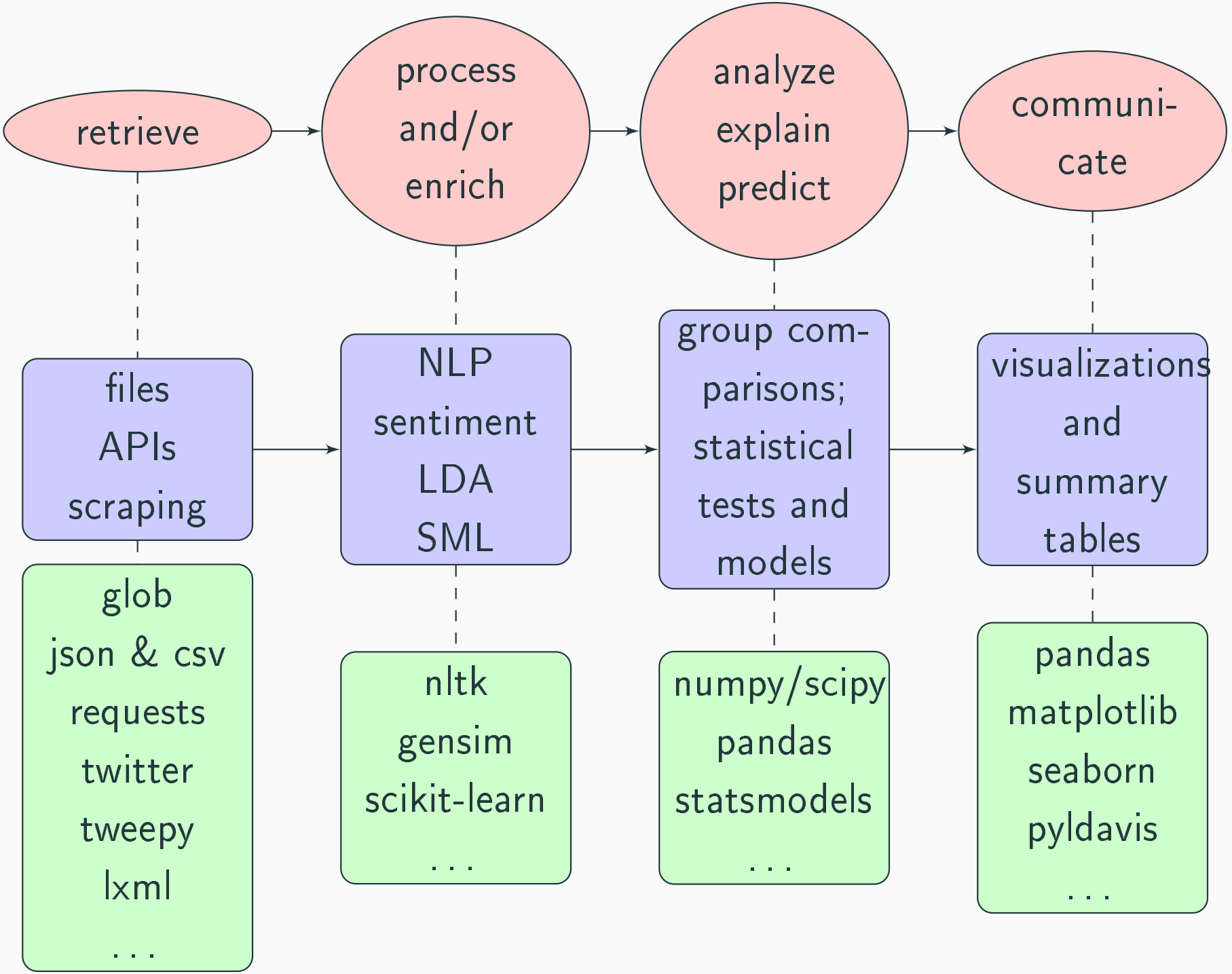
"[. . .] the computational social sciences employ the scientific method, complementing descriptive statistics with inferential statistics that seek to identify associations and causality. In other words, they are underpinned by an epistemology wherein the aim is to produce sophisticated statistical models that explain, simulate and predict human life."

Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society*, *1*(1), 1–12. doi:10.1177/2053951714528481

Steps of a CSS project

We learned techniques for:

- retrieving data
- processing data
- analyzing data
- visualising data


retrieve
process and/or enrich
analyze explain predict
communi-cate
files
APIs
scraping
NLP
sentiment
LDA
SML
group com-parisons; statistical tests and models
visualizations and summary tables
glob
json & csv
requests
twitter
tweepy
lxml
...
nltk
gensim
scikit-learn
...
numpy/scipy
pandas
statsmodels
...
pandas
matplotlib
seaborn
pyldavis
...

Looking back

A good workflow

The big picture

Start with pen and paper

1. Draw the Big Picture
2. Then work out what components you need

The big picture

Start with pen and paper

1. Draw the Big Picture
2. Then work out what components you need

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve one problem at a time (e.g., parsing a review page, or getting the URL to the next review page)
- (for instance, by using functions [next slide])

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve one problem at a time (e.g., parsing a review page, or getting the URL to the next review page)
- (for instance, by using functions [next slide])

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve one problem at a time (e.g., parsing a review page, or getting the URL to the next review page)
- (for instance, by using functions [next slide])

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve *one* problem at a time (e.g., parsing a review page; or getting the URLs of all review pages)
- (for instance, by using functions [next slides])

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve *one* problem at a time (e.g., parsing a review page; or getting the URLs of all review pages)
- (for instance, by using functions [next slides])

Develop components separately

If you copy-paste code, you are doing something wrong

- Write loops!
- If something takes more than a couple of lines, write a function!

Develop components separately

If you copy-paste code, you are doing something wrong

- Write loops!
- If something takes more than a couple of lines, write a function!

Copy-paste approach
(ugly, error-prone, hard to scale up)

```
allreviews = []

response = requests.get('http://xxxxx')
tree = fromstring(response.text)
reviewelements = tree.xpath('//div[@class="review"]')
reviews = [e.text for e in reviewelements]
allreviews.extend(reviews)

response = requests.get('http://yyyyy')
tree = fromstring(response.text)
reviewelements = tree.xpath('//div[@class="review"]')
reviews = [e.text for e in reviewelements]
allreviews.extend(reviews)
```

Better: for-loop

(easier to read, less error-prone, easier to scale up (e.g., more URLs, read URLs from a file or existing list)))

```
allreviews = []

urls = ['http://xxxxx', 'http://yyyyy']

for url in urls:
    response = requests.get(url)
    tree = fromstring(response.text)
    reviewelements = tree.xpath('//div[@class="review"]')
    reviews = [e.text for e in reviewelements]
    allreviews.extend(reviews)
```

Even better: for-loop with functions

(main loop is easier to read, function can be re-used in multiple contexts)

```
1  def getreviews(url):
2     response = requests.get(url)
3     tree = fromstring(response.text)
4     reviewelements = tree.xpath('//div[@class="review"]')
5     return [e.text for e in reviewelements]
6
7
8  urls = ['http://xxxxx', 'http://yyyyy']
9
10 allreviews = []
11 for url in urls:
12    allreviews.extend(getreviews(url))
```

And you can always do even better: including a docstring, use list comprehension

```
1  def getreviews(url):
2      '''scrapes all reviews from a given URL and returns a list of
           strings'''
3      response = requests.get(url)
4      tree = fromstring(response.text)
5      reviewelements = tree.xpath('//div[@class="review"]')
6      return [e.text for e in reviewelements]
7
8
9  urls = ['http://xxxxx', 'http://yyyyy']
10
11 allreviews = [getrieviews(url) for url in urls]
```

Scaling up

If you continue working in this field, look into aspects like code style, re-usability, scalability

- Use functions and classes (we didn't cover the latter...) to make code more readable and re-usable
- Avoid re-calculating values
- Think about how to minimize memory usage (e.g., generators)
- Think about writing/reading data on-the-fly (generators, again)
- Do not hard-code values, file names, etc., but take them as arguments

Make it robust

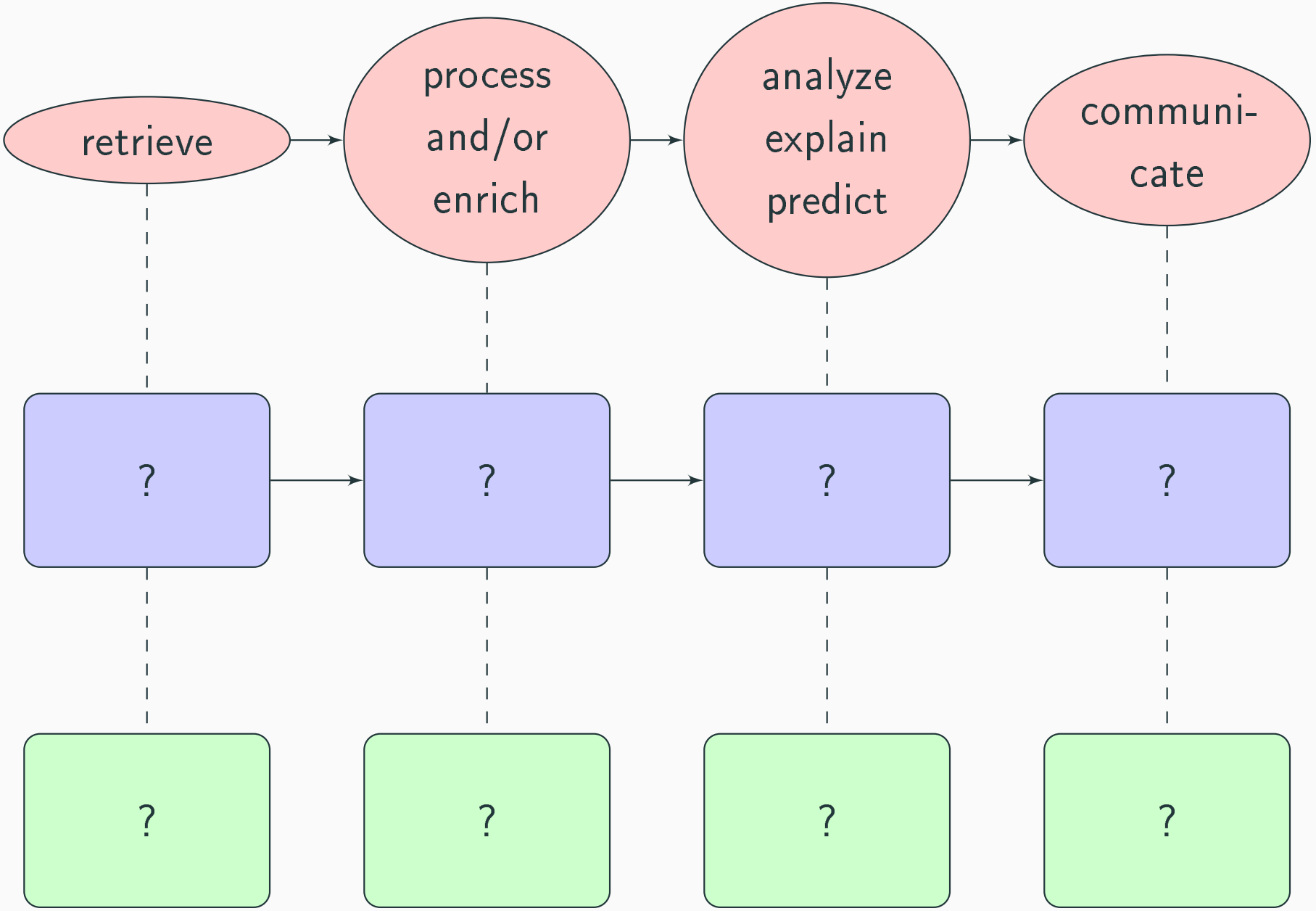
You cannot foresee every possible problem.
Most important: Make sure your program does not fail and loose all data just because something goes wrong at case 997/1000.

- Use `try/except` to explicitly tell the program how to handle errors
- Write data to files (or database) in between
- Use `assert len(x) == len(y)` for sanity checks

Looking forward

Looking forward

Techniqes we did not cover


retrieve
process and/or enrich
analyze explain predict
communi- cate
?
?
?
?
?
?
?
?

Retrieve

Webscraping with Selenium

- If content is dynamically loaded (e.g., with JavaScript), our approach doesn't work (because we don't have a browser).
- Solution: Have Python literally open a browser and literally click on things
- $\Rightarrow$ Appendix E

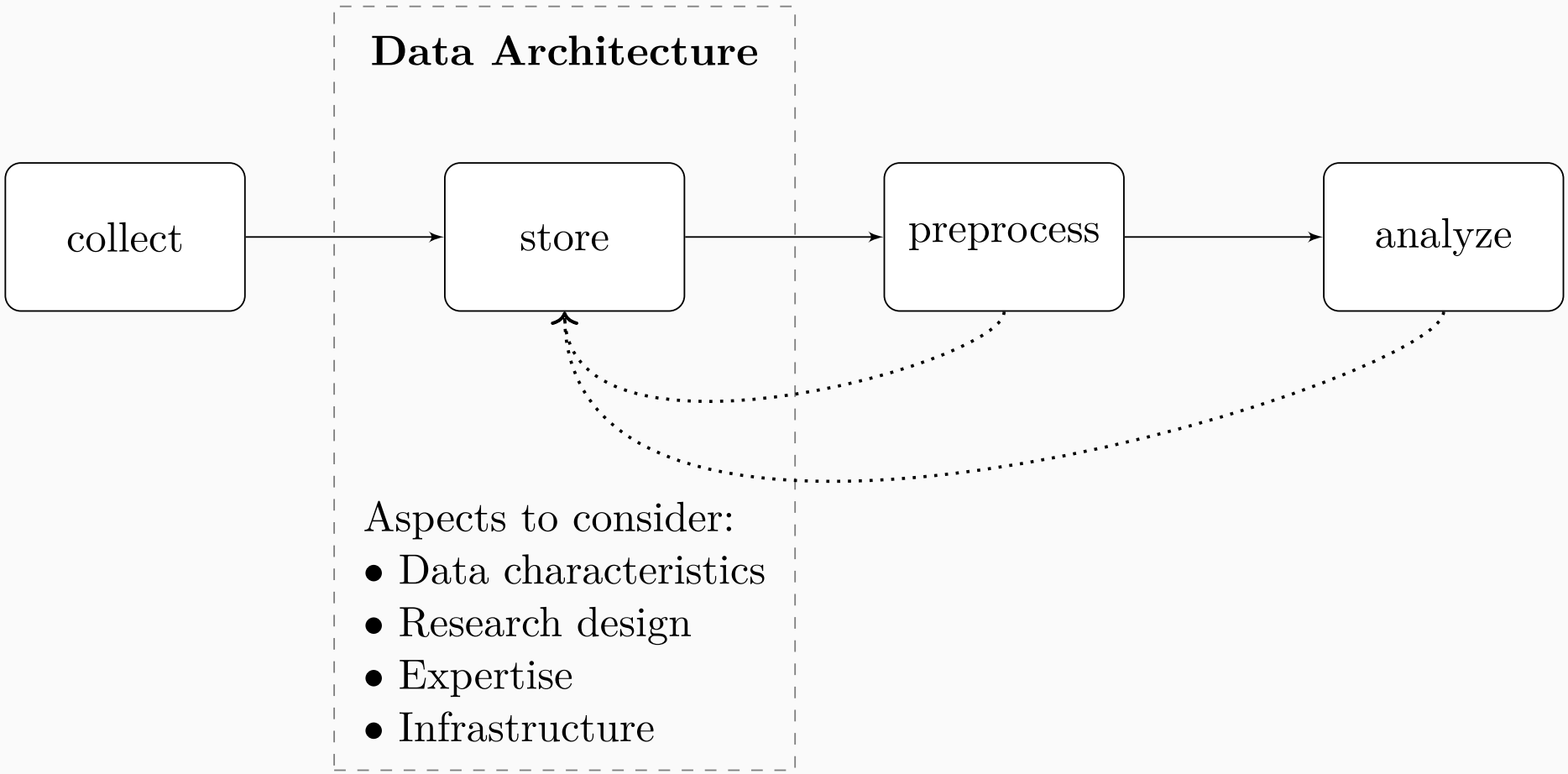
Retrieve

Use of databases

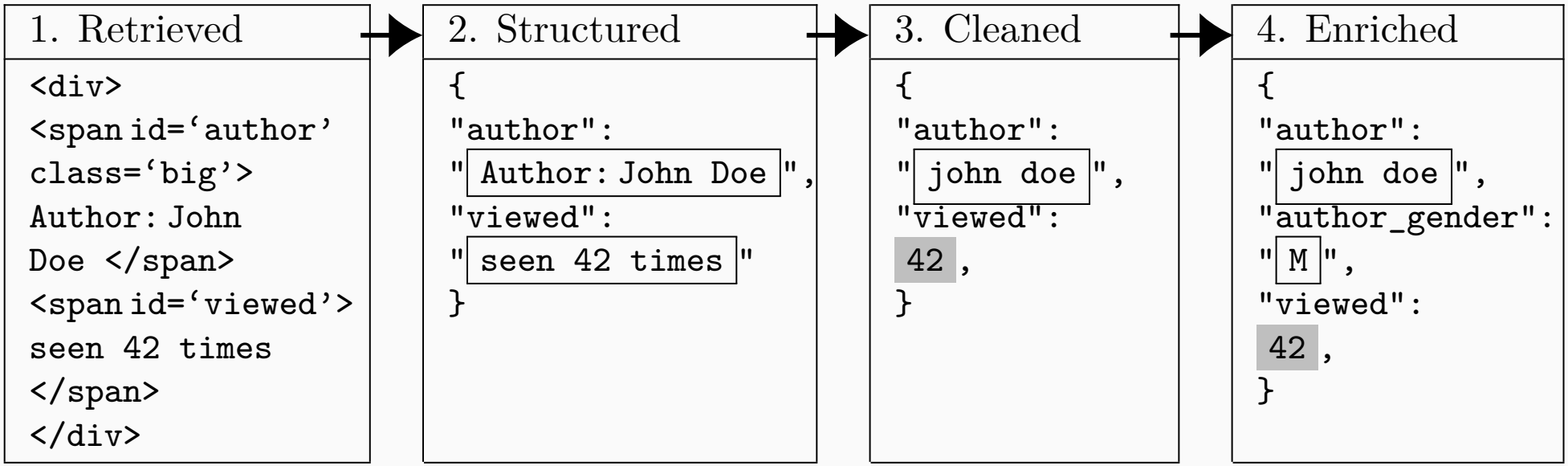
We did not discuss how to actually store the data

- We basically stored our data in files (often, one CSV or JSON file)
- But that's not very efficient if we have large datasets; especially if we want to select subsets later on
- SQL-databases to store tables (e.g., MySQL)
- NoSQL-databases to store less structured data (e.g., JSON with unknown keys) (e.g., MongoDB, ElasticSearch)
- $\Rightarrow$ Günther, E., Trilling, D., & Van de Velde, R.N. (2018). But how do we store it? (Big) data architecture in the social-scientific research process. In: *Stuetzer, C.M., Welker, M., & Egger, M. (eds.): Computational Social Science in the Age of Big Data. Concepts, Methodologies, Tools, and Applications*. Cologne, Germany: Herbert von Halem.

Storing data



From retrieved data to enriched data



Process and/or enrich

Advanced NLP

We did a lot of BOW (and some POS-tagging), but we can get more

- Named Entity Recognition (NER) to get names of people, organizations, ...
- State-of-the-art Dependency Parsing to find out exact relationships $\Rightarrow$ spacy, stanza (stanford NLP)

Process and/or enrich

Use images

- Supervised Machine learning does not care about what the features mean, so instead of texts we can also classify pictures
- Example: Teach the computer to decide whether an avatar on a social medium is an actual photograph of a person or a drawn image of something else
- This principle can be applied to many fields and disciplines – for example, it is possible to teach a computer to indicate if a tumor is present or not on X-rays of people's brains

Process and/or enrich

Image classification (see book)

Analyze/explain/predict

More advanced modelling

We only did some basic statistical tests

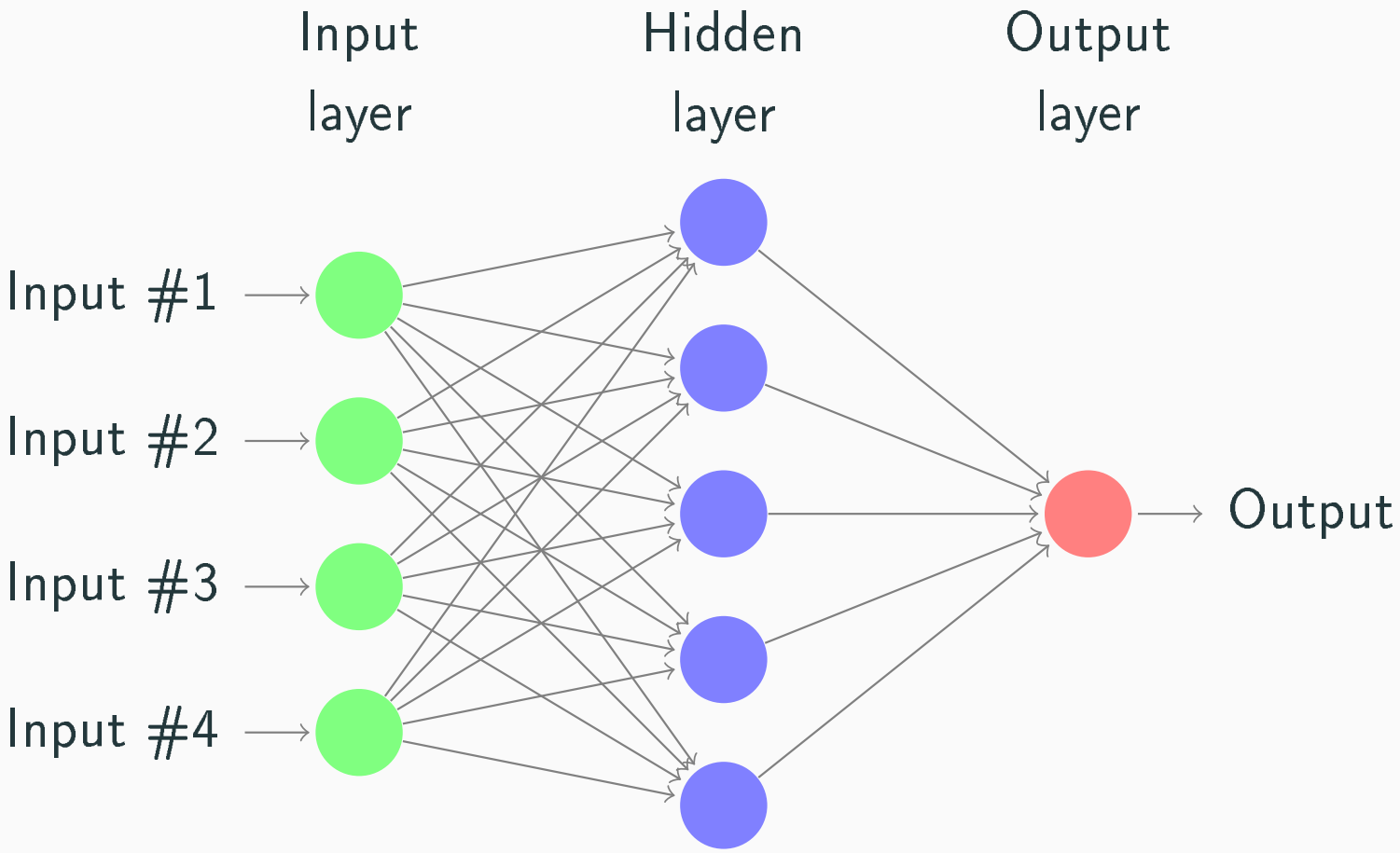
- There are more advanced regression techniques and dimension-reduction techniques tailored to data that are, e.g., large-scale, sparse, have a lot of features, ...
- $\Rightarrow$ scikit-learn, statsmodels

Looking forward

Neural Networks

Neural Networks

- In “classical” machine learning, we predict an outcome directly based on the input features
- In neural networks, we can have “hidden layers” that we predict
- These layers are not necessarily interpretable
- “Neurons” that “fire” based on an “activation function”



$\Rightarrow$ If we had multiple hidden layers in a row, we'd call it a *deep* network.

Your questions?

GOOD LUCK!