# Big Data and Automated Content Analysis (12EC)
# Week 13: »Looking back and forward«
# Wednesday

Felicia Loecherbach
f.loecherbach@uva.nl

Friday, May 12, 2022

UvA RM Communication Science

# Today

Looking back

- Putting the pieces together
- A good workflow

Looking forward

- Techniqes we did not cover (or only briefly)

Final project

# Today: Looking forward

# Looking back

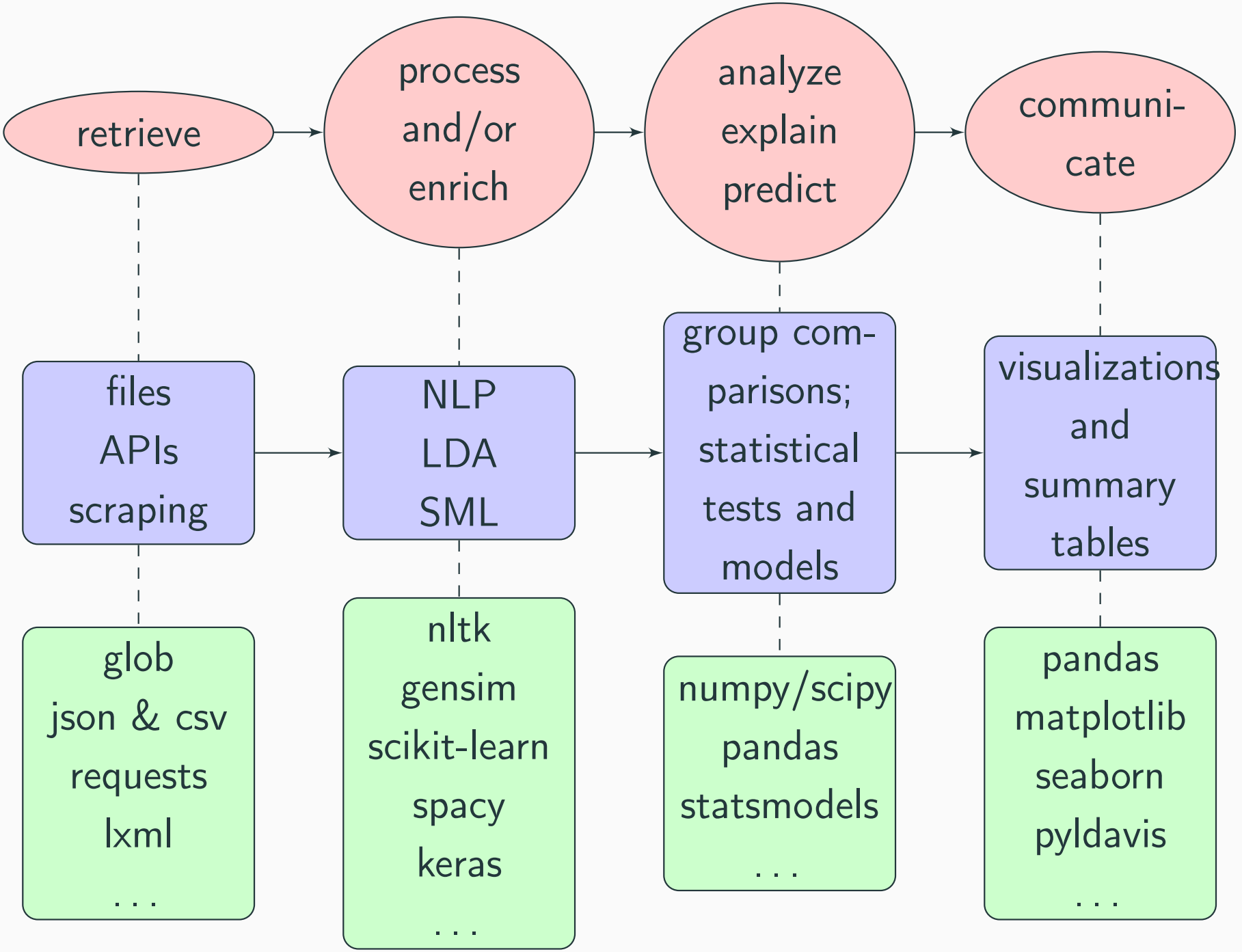
# Looking back

## Putting the pieces together

# 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 LDA SML
group com-parisons; statistical tests and models
visualizations and summary tables
glob json & csv requests lxml ...
nltk gensim scikit-learn spacy keras ...
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 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

## 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

## 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)
```

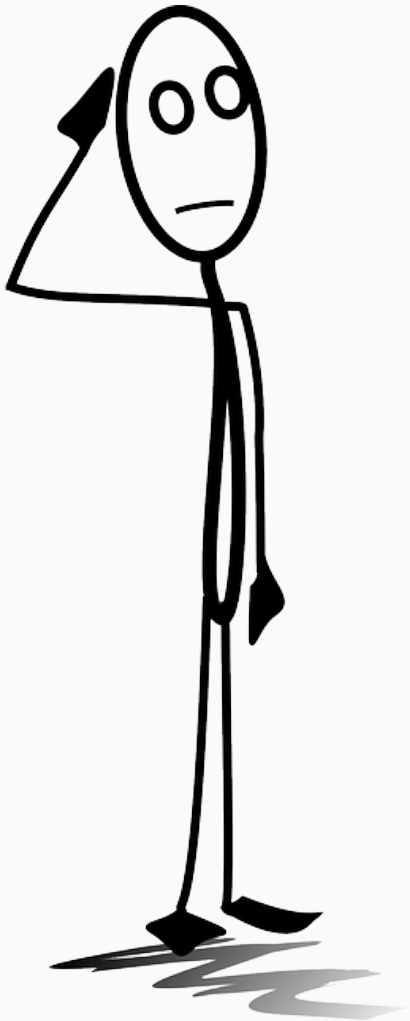
Even better: for-loop with functions

(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]
```



*Can we do even better? Maybe make it more robust?*

Directly append to a JSON-lines file so that we don't loose data if sth goes wrong

```
import json
from tqdm import tqdm   # provides a progress bar

with open("reviews.json", mode="w") as f:
   for url in tqdm(urls):
       f.write(json.dumps({"url":url, "reviews": getreviews(url)})
       f.write("\n")
```

# 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

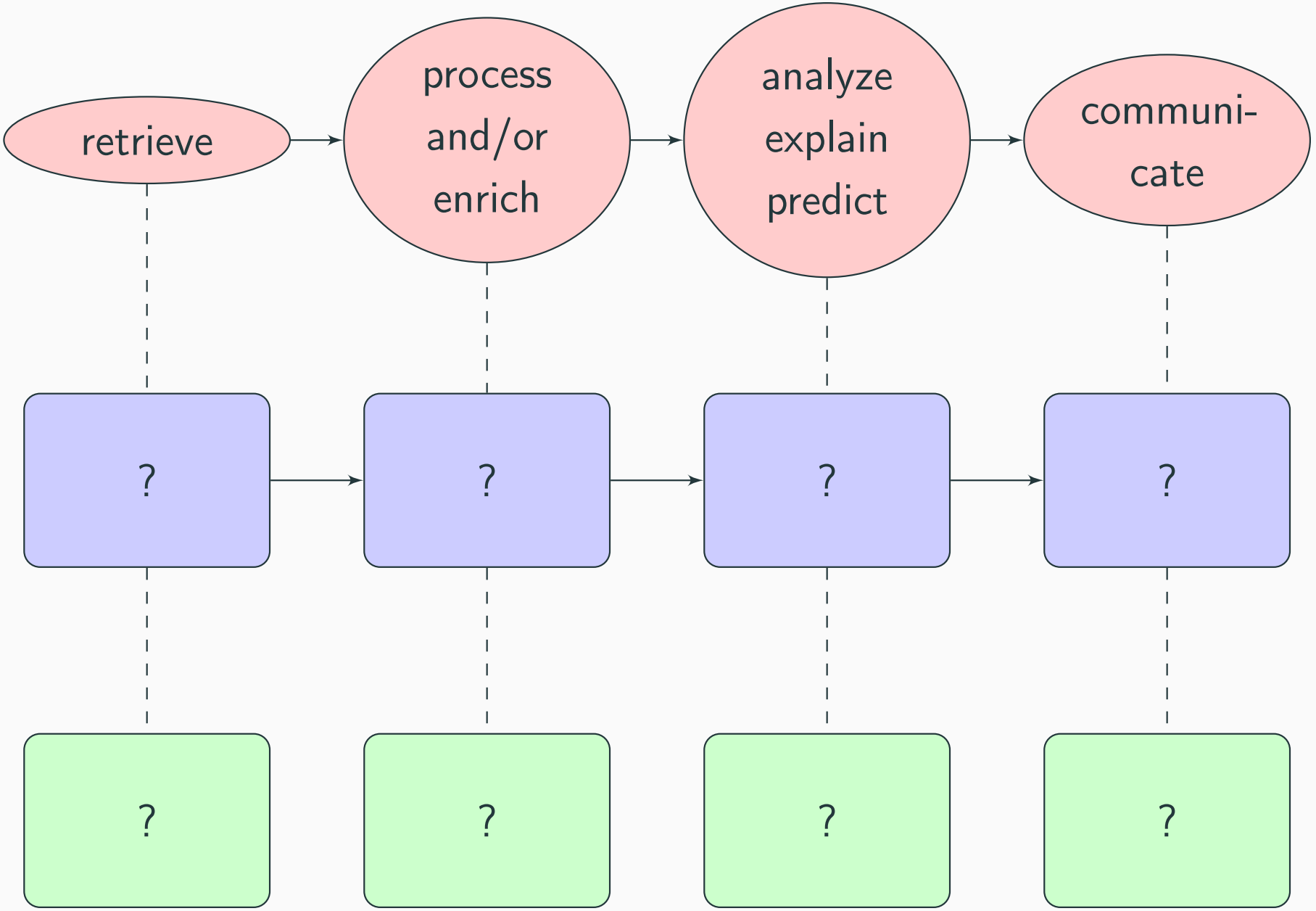
You cannot foresee every possible problem.
Most important: Make sure your program does not fail and loose all data just because something goes wrong at case 997/1000.

- 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 (or only briefly)


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

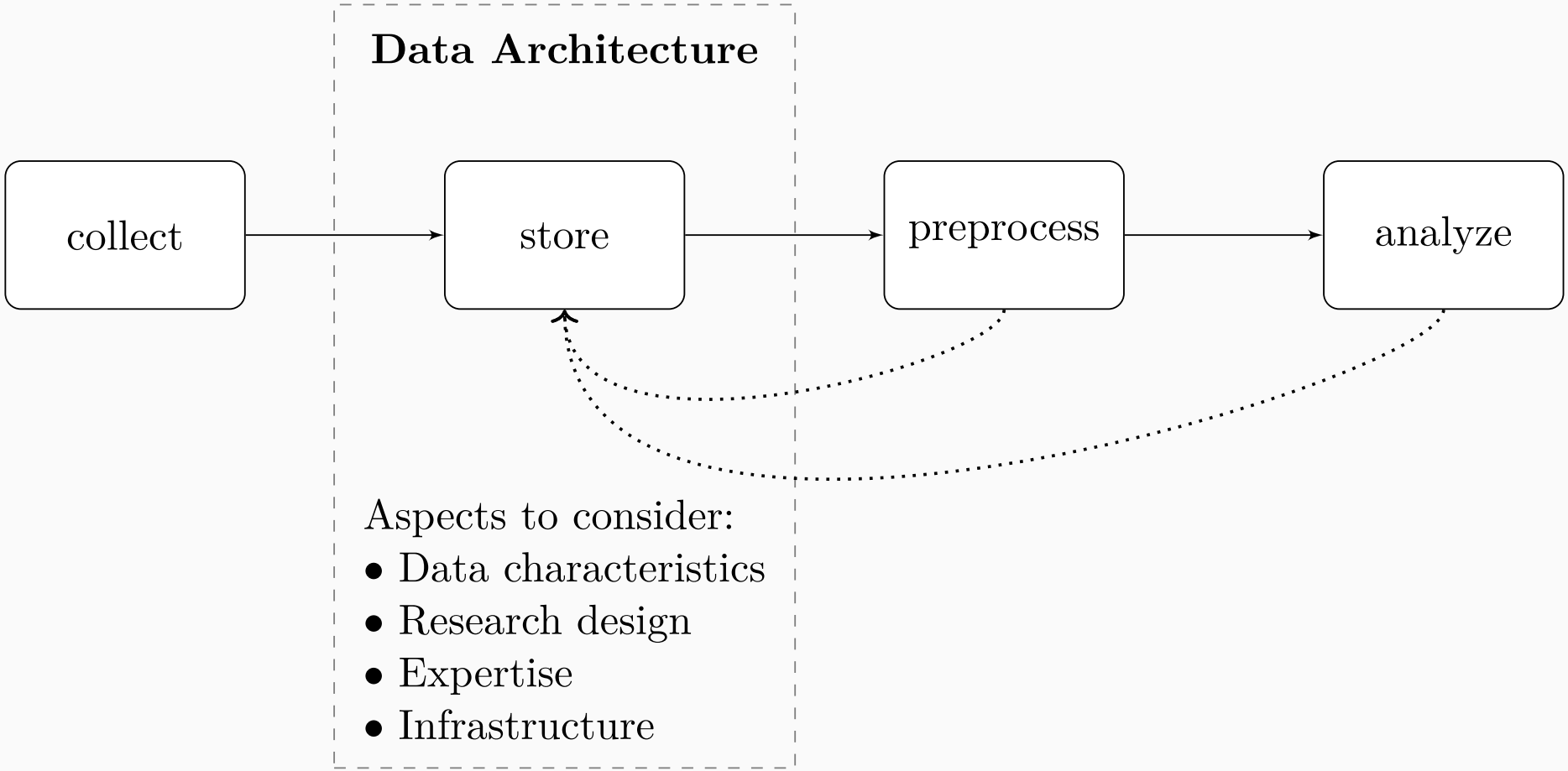
# Retrieve

## Use of databases (Günther et al., 2018)

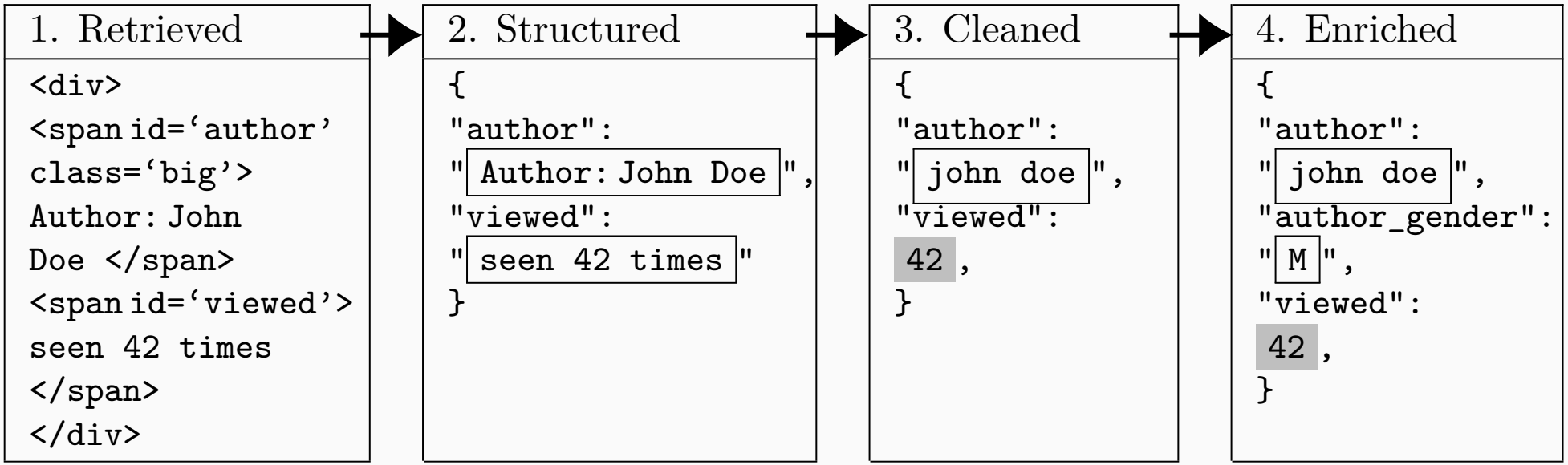
We did not discuss how to actually store the data

- 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)

# Storing data



# From retrieved data to enriched data



# Process and/or enrich

## Advanced NLP

We did a lot of BOW (and some part-of-speech (POS) tagging and named entity recognition (NER)), but we can do much more, such as

- State-of-the-art Dependency Parsing to find out exact relationships $\Rightarrow$ spacy, stanza (stanford NLP)
- ...

# 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

# Analyze/explain/predict

## Really go into deep learning

- We only got a brief intro to keras and to different archtectures – there is a lot to learn here.
- There is much more to learn about transformer-based models – check out huggingface.

# Some links about transformers

- https://nlp.seas.harvard.edu/2018/04/03/attention.html
- http://jalammar.github.io/illustrated-transformer/

# Final project

# Talk to me about your plans!

# How will the grade be determined?

Grading of the final project: 30% for the report and the documentation of the notebook; 70% for data, code, and analysis

## Report and documentation

(see also syllabus)

- Completeness and comprehensiveness
- Quality of argumentation
- Clear and correct presentation of and relevant selection of results
- Correctness and appropriateness of conclusion and suggestions for future research
- Outward appearance

# How will the grade be determined?

Grading of the final project: 30% for the report and the documentation of the notebook; 70% for data, code, and analysis

## Report and documentation

(see also syllabus)

- Completeness and comprehensiveness
- Quality of argumentation
- Clear and correct presentation of and relevant selection of results
- Correctness and appropriateness of conclusion and suggestions for future research
- Outward appearance

# How will the grade be determined?

Grading of the final project: 30% for the report and the documentation of the notebook; 70% for data, code, and analysis

## Data, code, and analysis

(see also syllabus)

- Covers techniques from most weeks
- Coding style and efficiency
- Follows best practices as you know *at the end* of the course (i.e., if you learned a better technique later, you typically should use (also) the better technique)
- Data quality and size (should be non-trivial; i.e., must make sense to use automated approach for it)
- Correctness of analysis and decisions
- Creativity, smart solutions and ideas, ...

Being able to conduct all steps of your own computational social science research project is one of the main learning goals of this course. This gives you a lot of freedom, but also inevitably means that many of you will end up doing very different things – and that reproducing some recipe is not what is tested and graded.

# Let's try to put some numbers on it

Some hypothetical scenario's (as an **indication**):

- Existing dataset downloaded from Kaggle, limited NLP, simple basic LDA: $< 5.0$
- Existing dataset, but good and extensive NLP, extensive machine learning approaches; yet, not all best practices followed: $6 - 7$
- Data aquired via APIs or webscraping, multiple NLP techniques applied, SML including extensive comparisons, hyperparameter tuning, visualizations of results, advanced models, maybe some statistical tests for comparisons after enriching: $7 - 8$
- Next to fulfilling all criteria, the projct solves a complex task that is going clearly beyond the specific examples used in class and/or in the take-home assignments (e.g., develops a complex web scraper for the data collection stage): $8+$

(all of this assuming that the methods are suitable to answer the RQ, otherwise the grade is of course lower)

# Let's try to put some numbers on it

Some hypothetical scenario's (as an **indication**):

- Existing dataset downloaded from Kaggle, limited NLP, simple basic LDA: $< 5.0$
- Existing dataset, but good and extensive NLP, extensive machine learning approaches; yet, not all best practices followed: $6 - 7$
- Data aquired via APIs or webscraping, multiple NLP techniques applied, SML including extensive comparisons, hyperparameter tuning, visualizations of results, advanced models, maybe some statistical tests for comparisons after enriching: $7 - 8$
- Next to fulfilling all criteria, the projct solves a complex task that is going clearly beyond the specific examples used in class and/or in the take-home assignments (e.g., develops a complex web scraper for the data collection stage): $8+$

(all of this assuming that the methods are suitable to answer the RQ, otherwise the grade is of course lower)

# Let's try to put some numbers on it

Some hypothetical scenario's (as an **indication**):

- Existing dataset downloaded from Kaggle, limited NLP, simple basic LDA: $< 5.0$
- Existing dataset, but good and extensive NLP, extensive machine learning approaches; yet, not all best practices followed: $6 - 7$
- Data aquired via APIs or webscraping, multiple NLP techniques applied, SML including extensive comparisons, hyperparameter tuning, visualizations of results, advanced models, maybe some statistical tests for comparisons after enriching: $7 - 8$
- Next to fulfilling all criteria, the projct solves a complex task that is going clearly beyond the specific examples used in class and/or in the take-home assignments (e.g., develops a complex web scraper for the data collection stage): $8+$

(all of this assuming that the methods are suitable to answer the RQ, otherwise the grade is of course lower)

# Let's try to put some numbers on it

Some hypothetical scenario's (as an **indication**):

- Existing dataset downloaded from Kaggle, limited NLP, simple basic LDA: $< 5.0$
- Existing dataset, but good and extensive NLP, extensive machine learning approaches; yet, not all best practices followed: $6 - 7$
- Data aquired via APIs or webscraping, multiple NLP techniques applied, SML including extensive comparisons, hyperparameter tuning, visualizations of results, advanced models, maybe some statistical tests for comparisons after enriching: $7 - 8$
- Next to fulfilling all criteria, the projct solves a complex task that is going clearly beyond the specific examples used in class and/or in the take-home assignments (e.g., develops a complex web scraper for the data collection stage): $8+$

(all of this assuming that the methods are suitable to answer the RQ, otherwise the grade is of course lower)

Rules of thumb: The more its mere replication, the lower the grade. The more you cover *each* of the steps (today's slides) extensively, the higher the grade.

# Your questions?

# GOOD LUCK!

Next week, open lab if you have questions.

# References


Günther, E., Trilling, D., & Van de Velde, R. N. (2018). **But how do we store it? (Big) data architecture in the social-scientic research process.** In C. M. Stuetzer, M. Welker, & M. Egger (Eds.), *Computational social science in the age of Big Data. Concepts, methodologies, tools, and applications*. von Halem.