# Workshop “Introduction to Python”

organized by the
Zeppelin University Friedrichshafen

Jan 24 and 25, 2022

# Contents

# 1 Introduction, Warm Up, Set Up

- Python puzzles / recap
  - data types
  - control structures
  - classes and objects
  - modules
- Python runtime and development environments
  - Python interpreter
  - editors, IDEs
  - Jupyter notebooks, Anaconda
  - virtual environment, Docker
  - Google Colaboratory

## 1.1 Python Puzzles / Recap

What will the Python3 interpreter return on the following statements…

### 1.1.1 Data Types

```
In [ ]: a = 3 # integer
        b = 2
        a * b
```

```
In [ ]: c = 2.0 # floating point number
        a * c
```

```
In [ ]: t = True # boolean value
        f = False
        t and f
```

```
In [ ]: t or f
```

```
In [ ]: s = 'foo' # string
        s + s
```

```
In [ ]: s[0]
```

```
In [ ]: l = [1, 2, 3] # list
        l[0]
```

```
In [ ]: l[3]
```

```
In [ ]: l[-1]
In [ ]: d = {'a': 1, 'b': 2, 'c': 3, 'b': 1.5} # dictionary
        d['b']
```

```
In [ ]: s = {'a', 'b', 'c', 'a'} # set
        s
```

```
In [ ]: t = (1, 2) # tuple
        t[0]
```

```
In [ ]: l[2] = 4
        l
```

```
In [ ]: t = (1, 2)
        t[1] = 3
```

#### Mutable and Immutable Data Types

- tuples are immutable, i.e. once created you cannot change the content
- lists, dictionaries, sets are mutable
- numbers and strings are also immutable
- immutable data types avoid programming errors and also allow for certain optimizations

```
In [ ]: s = 'foo'
        s[0] = 'F'
```

```
In [ ]: # but you can assign a new string to the variable `s`
        s = 'Foo'
        s
```

```
In [ ]: l = [1, 2, 3]
        l2 = l
        l2
```

```
In [ ]: l[2] = 4
        l2
```

### 1.1.2 Control Structures

#### Loops

```
In [ ]: l = [1, 2, 3]
        for i in l:
            print(i)
```

```
In [ ]: i = 1
        while i <= 3:
            print(i)
            i += 1
```

### If-Else Conditions

```
In [ ]: for i in range(0, 5):
            if i % 2 == 0:
                print("Even:", i)
            else:
                print("Odd:", i)
```

```
In [ ]: i = 1
        while True:
            print(i)
            if i == 3:
                break
            i += 1
```

### Functions

Functions are...

- code blocks only executed when called
- reusable (can be called repeatedly from various places in the code)
- the primary method to organize code and make it readable and understandable

```
In [ ]: def fun(n): # one required argument
            for i in range(0, n):
                print("You called me?")
        fun(2)
```

```
In [ ]: def fun(x='You'): # one optional argument
            """Ask whether X called me"""
            print(x, "called me?")

        fun()          # use default
        fun('Who')     # positional argument
        fun(x='They') # named "keyword" argument
```

```
In [ ]: def fun(x='You'):
            return "%s called me?" % x

        question = fun('Who')
        question
```

Functions may return a single value to the caller. Tuples (or lists, dictionaries, etc.) can be used to return multiple values. A return statement is optional. Multiple return statements in if-branches are possible.

### 1.1.3 Classes and Objects

The object-oriented programming paradigm combines data and code in “objects”. Every “object” is an instance of a “class”. The “class” defines

- the data types and possible values an object of the class holds

- “methods” - functions to read, write or interact with data values hold by the object

### Object Methods

Variables of built-in data types are all objects of built-in classes and provide multiple methods…

```
In [ ]: s = 'foo'
        s.capitalize() # call a method of a string object
```

Tip: many Python editors let you show a list of available methods for a given object variable.

In the Jupyter notebook editor: enter `s.` and press `<tab>` to get a list of methods of `str` objects.

```
In [ ]: #s.
```

```
In [ ]: type(s)
```

```
In [ ]: help(str)
```

```
In [ ]: help(str.endswith)
```

```
In [ ]: !pydoc str.endswith  # `!` runs another command (not the Python interpreter)
```

What could be the methods provided by the `list` built-in class? Think about it before calling `help(list)`!

### Defining Classes

```
In [ ]: class Sentiment:

            values = {'sad', 'neutral', 'happy'}

            def __init__(self, value='neutral'):
                if value not in Sentiment.values:
                    raise ValueError("Only the following values are supported: %s"
                                     % Sentiment.values)
                self.value = value

            def get(self):
                return self.value

            def __repr__(self):
                return self.value

            @staticmethod
            def guess(text):
                if 'happy' in text or 'excited' in text:
                    return Sentiment('happy')
                if 'sad' in text or 'angry' in text:
                    return Sentiment('sad')
                return Sentiment('neutral')
```

```
        im_feeling = Sentiment.guess("I'm really happy!")

        print(im_feeling)
```

### 1.1.4 Exceptions

It isn't easy maybe even impossible to foresee all special cases during the execution of a computer program. Exceptions offer a way to handle unforeseen conditions, for example the following erroneous or malicious input to our sentiment guesser:

```
In [ ]: im_feeling = Sentiment.guess(0)
```

```
In [ ]: im_feeling = Sentiment('neutral')

        try:
            im_feeling = Sentiment.guess(0)
            # continue using im_feeling
        except Exception as e:
            print('Got exception:', e)

        # the program continues
        print(im_feeling)
```

### 1.1.5 Modules

Modules make Python code reusable.

#### Create a Python Module

Copy the definition of the class "Sentiment" into a file sentiment.py in the folder `scripts`. Now you can load the class by…

```
In [ ]: from scripts.sentiment import Sentiment

        Sentiment()
```

#### The Python Standard Library

The Python Standard Library includes many modules to handle file formats, process texts, use the internet, etc., etc. Just import one of the modules or functions or classes defined there:

```
In [ ]: import time

        time.asctime()
```

```
In [ ]: from time import asctime, sleep

        print(asctime())
        sleep(3)
        print(asctime())
```

### Third-Party Modules

To install a package from the Python Package Index, run `pip install <package>`…

```
In [ ]: !pip install matplotlib
```

… but before run `pip list` or `pip show matplotlib` (or just try `import matplotlib`) to figure out whether it is already installed.

A good and common practice is to list all modules required by a project in a file `requirements.txt`. The entire list of requirements can then be installed by `pip install -r requirements.txt`. Also this project ships with a requirements.txt.

## 1.2 Python Runtime and Development Environments

### 1.2.1 The Python Interpreter

- installed from python.org
- on Linux: already installed or installable as package of the Linux Distribution (Debian, Ubuntu, Red Hat, SuSE, etc.)
- otherwise: it's recommended to rely on a distribution which bundles the Python interpreter with common Python modules and tools - esp. Anaconda, a distribution of Python and R for scientific computing

### 1.2.2 Jupyter Notebooks

The Jupyter notebook is an environment to interactively create a "notebook", a JSON-encoded document containing a list of input/output pairs (code, text using Markdown markup, images/plots). Notebooks are served by the notebook server and viewed/edited in the browser or can be converted into various document formats.

### 1.2.3 Editor and IDE

A good editor or an integrated development environment (IDE) will speed up coding by providing autocompletion, syntax highlighting and syntax checking. If your code gets bigger, an IDE supports the development by automated builds and deployments of the code, a runtime for tests and a visual debugger to locate errors ("bugs") in your code.

Unfortunately, there are many good IDEs available for Python, to list just a few:

- PyDev
- Visual Studio Code
- PyCharm (commercial)

### 1.2.4 Virtual Environment and Docker

Why you need encapsulated environments to run applications or projects? The documentation of the Python virtual environments explains…

> Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.
>
> This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

1. create a virtual environment in current director in the subfolder `.venv/`

   ```
   virtualenv .venv
   ```

2. activate the environment

   ```
   source .venv/bin/activate
   ```

3. install packages (placed below `./.venv/`)

   ```
   pip install ...
   ```

4. run Python…
5. deactivate the environment

   ```
   deactivate
   ```

If more than Python modules are project-specific: Docker allows to bundle a Python interpreter (eg. an older version), specific modules and additional software, pack it as runtime image and run it in a "container" without the need to install anything on the host system.

### 1.2.5 Google Colaboratory

Google Colaboratory or "Colab" is a Jupyter notebook environment running in the Google cloud. The notebooks are stored on Google Drive. Basic usage is free but requires a Google user account. The paid "Colab Pro" allows to use more hardware resources (RAM, CPU, GPU, TPU) and to run the notebooks without being connected to a web browser.

Colab supports loading notebooks on Github. To load one of the workshop notebooks, please navigate to https://colab.research.google.com/github/sebastian-nagel/introduction-to-python/blob/main/.

Running a notebook in the cloud requires that analyzed data is uploaded to the cloud or is available online. This might be a hurdle if the data is private or sensitive. In order to load the workshop data from Github into the workspace of a Colab notebook, simply add a cell to the beginning of a notebook with the following two instructions:

```
!git clone https://github.com/sebastian-nagel/introduction-to-python.git
%cd /content/introduction-to-python/
```

# 2 Working with Structured Data

- read data from local files
- read CSV and JSON
- first steps data analysis with data frames and the pandas library
- basic plotting of data

## 2.1 Example: “Tree Cadastre of the City of Konstanz”

First, get the tree cadastre data from the open data portal of the city of Konstanz. Save it on the file path shown below. The CSV file is then loaded into a pandas “DataFrame”:

```
In [ ]: ### if running on Google Colab:
        ### - load data from Github into the Google Colab notebook
        ![ -v COLAB_GPU ] && [[ ! -e /content/introduction-to-python ]] && git clone https://github.com/sebastian-nagel/
        ### - change into project directory to have data/ available
        ###   (supposed to fail if running a local Jupyter notebook)
        %cd -q /content/introduction-to-python/
```

```
In [1]: import pandas as pd

        tree_cadastre_file = 'data/KN_Baumkataster_2020.csv'
        df = pd.read_csv(tree_cadastre_file)
        df.shape  # table size (rows, columns)
```

```
Out[1]: (15711, 13)
```

Note: Pandas could read the CSV directly from the WWW if a URL is passed. With internet access and supposed the download URL is still valid, the data frame is also loaded by

```
df = pd.read_csv('https://opendata.arcgis.com/datasets/c160f0a79a584ddf80cc65477fe58f4e_0.csv')
```

Let’s now have a first and quick look into the data using pandas methods:

```
In [2]: df.head() # first lines of the table
```

```
Out[2]:           X          Y  OBJECTID  baumId  baumNr  baumart  hoeheM  \
        0  9.159063  47.739307         1       2       1       52    12.0
        1  9.158918  47.739471         2       4       4      182    11.0
        2  9.159193  47.739428         3       5       3       52    11.0
        3  9.158987  47.739541         4       6       5       37    14.0
        4  9.159219  47.739676         5       9       8      284    22.0

           kronendurchmesserM  stammumfangCM                    location  \
        0                   6           72.0  Bubenbad Dingelsdorf (754)
        1                  12          169.0  Bubenbad Dingelsdorf (754)
        2                   7           74.0  Bubenbad Dingelsdorf (754)
        3                   7          135.0  Bubenbad Dingelsdorf (754)
        4                  20          380.0  Bubenbad Dingelsdorf (754)

                          Name_dt             Name_lat AGOL_Name
        0      Erle, Schwarz-Erle      Alnus glutinosa     Alnus
        1       Nussbaum, Walnuss        Juglans regia   Juglans
        2      Erle, Schwarz-Erle      Alnus glutinosa     Alnus
        3       Ahorn, Berg-Ahorn  Acer pseudoplatanus      Acer
        4  Pappel, Schwarz-Pappel        Populus nigra   Populus
```

```
In [3]: df.describe() # descriptive statistics (numerical columns)
```

```
Out[3]:                    X             Y      OBJECTID        baumId        baumNr  \
        count  15711.000000  15711.000000  15711.000000  15711.000000  15711.000000
        mean       9.169897     47.681721   7856.000000  13361.111832     57.941315
        std        0.022084      0.023527   4535.519375   9558.292963    109.965696
        min        9.106630     47.653444      1.000000      2.000000      0.000000
        25%        9.153555     47.666961   3928.500000   5844.500000      5.000000
        50%        9.170588     47.674747   7856.000000  12181.000000     20.000000
        75%        9.180610     47.683773  11783.500000  17923.500000     58.000000
        max        9.217534     47.748520  15711.000000  39080.000000    805.000000

                    baumart        hoeheM  kronendurchmesserM  stammumfangCM
        count  15711.000000  15706.000000        15711.000000   15704.000000
        mean     307.457959     10.688718            6.124944     113.009488
        std      206.677390      6.416883            3.883879      83.834009
        min        1.000000      1.000000            0.000000       0.000000
        25%       77.000000      5.000000            3.000000      50.000000
        50%      322.000000      9.000000            6.000000      93.000000
        75%      501.000000     15.000000            8.000000     157.000000
        max      637.000000     40.000000           30.000000     900.000000
```

```
In [4]: df.nunique() # number of unique values in each column
```

```
Out[4]: X                     15705
        Y                     15705
        OBJECTID              15711
        baumId                15711
        baumNr                  801
        baumart                 296
        hoeheM                   36
        kronendurchmesserM       26
        stammumfangCM           464
        location                775
        Name_dt                 294
        Name_lat                296
        AGOL_Name                35
        dtype: int64
```

… and we identify the following columns (cf. the provided tree cadastre metadata):

- the pandas row index
- “X” and “Y”: geographic coordinates (longitude and latitude)
- “OBJECTID”, “baumid”, “baumNr”: three different tree IDs

- “baumart”: a nummeric species ID
- “hoeheM”: the tree height (m)
- “kronendurchmesserM”: treetop diameter (m)
- “stammumfangCM”: trunk perimeter (cm)
- “location”: coarse location of the tree (street name)
- “Name_dt”: German tree name
- “Name_lat”: Latin tree name
- “AGOL_Name”: vendor-specific name (“AGOL” = “ArcGIS Online”)

We clean up the data a little bit: - translate the German column names - drop the columns not used later on - use the column “OBJECTID” as row index

```
In [5]: df.rename(columns={'hoeheM': 'height (m)',
                           'kronendurchmesserM': 'treetop diameter (m)',
                           'stammumfangCM': 'trunk perimeter (cm)'},
                  inplace=True)
        df.drop(columns=['baumId', 'baumNr', 'baumart', 'AGOL_Name'], inplace=True)
        df.set_index('OBJECTID', inplace=True)
        df.head()
```

```
Out[5]:                  X          Y  height (m)  treetop diameter (m)  \
        OBJECTID
        1         9.159063  47.739307        12.0                     6
        2         9.158918  47.739471        11.0                    12
        3         9.159193  47.739428        11.0                     7
        4         9.158987  47.739541        14.0                     7
        5         9.159219  47.739676        22.0                    20

                  trunk perimeter (cm)                    location  \
        OBJECTID
        1                         72.0  Bubenbad Dingelsdorf (754)
        2                        169.0  Bubenbad Dingelsdorf (754)
        3                         74.0  Bubenbad Dingelsdorf (754)
        4                        135.0  Bubenbad Dingelsdorf (754)
        5                        380.0  Bubenbad Dingelsdorf (754)

                                 Name_dt             Name_lat
        OBJECTID
        1             Erle, Schwarz-Erle      Alnus glutinosa
        2              Nussbaum, Walnuss        Juglans regia
        3             Erle, Schwarz-Erle      Alnus glutinosa
        4              Ahorn, Berg-Ahorn  Acer pseudoplatanus
        5         Pappel, Schwarz-Pappel        Populus nigra
```

## 2.2 Count Items

```
In [6]: # count tree names and show the N most frequent tree names
        N = 20
        top_trees = df['Name_lat'].value_counts().head(N).to_frame()
        top_trees
```

```
Out[6]:                                       Name_lat
        Platanus x acerifolia                      887
        Betula pendula                             809
        Quercus robur                              667
        Fraxinus excelsior                         614
        Tilia cordata                              605
        Malus domestica                            539
        Salix alba                                 536
        Acer platanoides                           523
        Acer pseudoplatanus                        517
        Pyrus communis                             513
        Carpinus betulus                           503
        Acer campestre                             428
        Juglans regia                              397
        Aesculus hippocastanum                     372
        Fagus sylvatica                            293
        Fraxinus excelsior 'Westhof's Glorie'      261
        Tilia platyphyllos                         252
        Prunus avium                               250
        Tilia cordata 'Greenspire'                 244
        Gleditsia triacanthos 'Inermis'            234
```

```
In [7]: # also show the top N German names
        df['Name_dt'].value_counts().head(20).to_frame()
```

```
Out[7]:                                  Name_dt
        Platane                              952
        Birke, Sand-Birke                    809
        Eiche, Stiel-Eiche, Sommer-Eiche     667
        Esche, Esche gemeine                 614
        Linde, Winter-Linde                  605
        Kultur-Apfel                         539
        Weide, Silber-Weide                  536
        Ahorn, Spitz-Ahorn                   523
        Ahorn, Berg-Ahorn                    517
        Birne, Holz-Birne                    513
        Weißbuche, Hainbuche                 503
        Ahorn, Feld-Ahorn                    428
        Nussbaum, Walnuss                    397
        Rosskastanie                         372
        Buche, Rotbuche                      293
        Straßen-Esche                        261
        Linde, Sommer-Linde                  252
        Kirsche, Vogel-Kirsche               250
        Linde "Greespire"                    244
        Dornenlose Gleditschie               234
```

Obviously, German names are less specific (there are more items of “Platane” than “Platanus x acerifolia”). To avoid inconsistencies we’ll use the Latin names in the next steps. Because not everybody knows Latin well enough or studied botanology, let’s prepare a translation table to see the Latin and German names site by site. We will later look how we could get the tree names in other languages as well.

```
In [8]: tree_name_translation = df.loc[df['Name_lat'].isin(top_trees.index),
                                       ['Name_lat', 'Name_dt']]
        tree_name_translation['count'] = 1
        tree_name_translation.groupby(['Name_lat', 'Name_dt']).sum() \
            .sort_values('count', ascending=False)
```

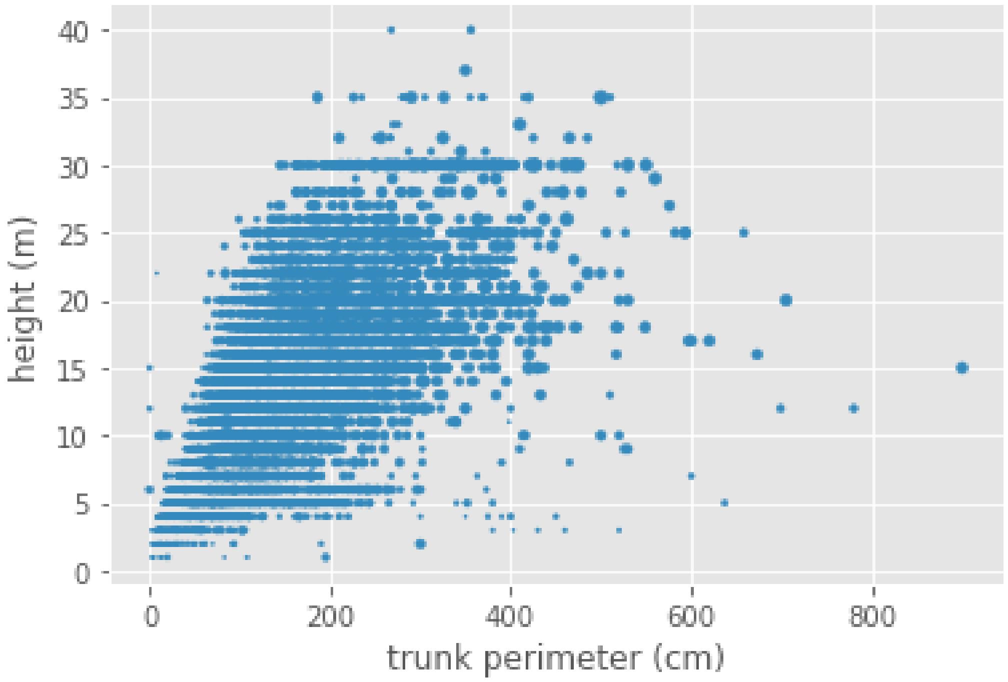
```
Out[8]:                                                                  count
        Name_lat                            Name_dt
        Platanus x acerifolia               Platane                        887
        Betula pendula                      Birke, Sand-Birke              809
        Quercus robur                       Eiche, Stiel-Eiche, Sommer-Eiche  667
        Fraxinus excelsior                  Esche, Esche gemeine           614
        Tilia cordata                       Linde, Winter-Linde            605
        Malus domestica                     Kultur-Apfel                   539
        Salix alba                          Weide, Silber-Weide            536
        Acer platanoides                    Ahorn, Spitz-Ahorn             523
        Acer pseudoplatanus                 Ahorn, Berg-Ahorn              517
        Pyrus communis                      Birne, Holz-Birne              513
        Carpinus betulus                    Weißbuche, Hainbuche           503
        Acer campestre                      Ahorn, Feld-Ahorn              428
        Juglans regia                       Nussbaum, Walnuss              397
        Aesculus hippocastanum              Rosskastanie                   372
        Fagus sylvatica                     Buche, Rotbuche                293
        Fraxinus excelsior 'Westhof's Glorie' Straßen-Esche                261
        Tilia platyphyllos                  Linde, Sommer-Linde            252
        Prunus avium                        Kirsche, Vogel-Kirsche         250
        Tilia cordata 'Greenspire'          Linde "Greespire"              244
        Gleditsia triacanthos 'Inermis'     Dornenlose Gleditschie         234
```

## 2.3 Plotting

We start with a first trivial scatter plot of the 3 metric values using the plot method of the DataFrame. We choose the matplotlib's style "ggplot" which mimics the look of the plots produced by a popular plotting package for R. There are many more styles available.

```
In [9]: import matplotlib
        import matplotlib.pyplot as plt
        plt.style.use('ggplot')

        df.plot(kind='scatter',
                x='trunk perimeter (cm)',
                y='height (m)',
                s='treetop diameter (m)')
```

```
Out[9]: <AxesSubplot:xlabel='trunk perimeter (cm)', ylabel='height (m)'>
```



Insights from the first plot: - data gathering: heights above 25m are rather estimates and tend to be rounded to the next 5m interval - some noise, eg. high trees with very small trunk perimeter - tree height, trunk perimeter and treetop diameter correlate but with broad variance

Let's look deeper into the correlation using the Pearson coefficient:

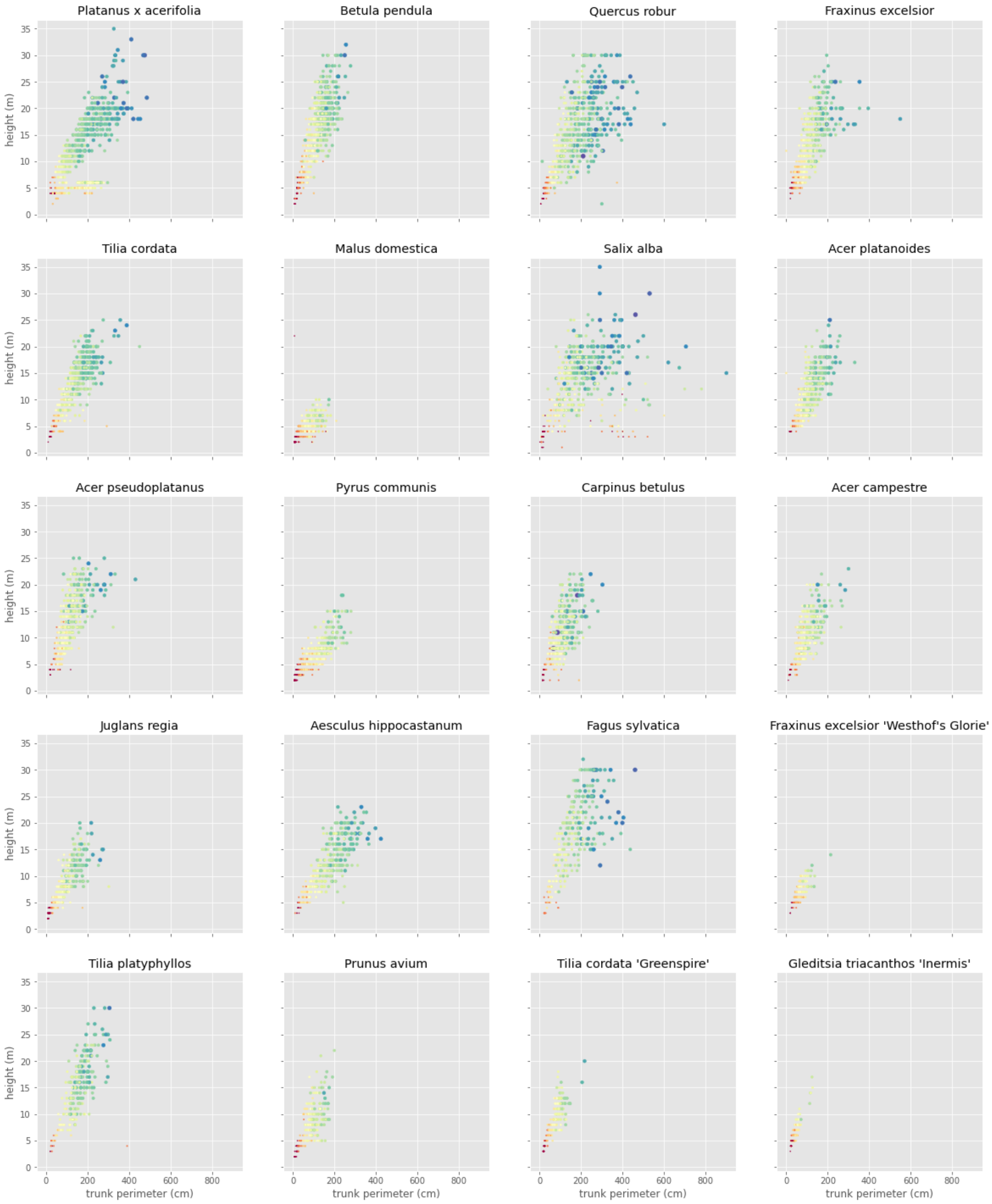
```
In [10]: metric_columns = ['trunk perimeter (cm)', 'height (m)', 'treetop diameter (m)']
         df[metric_columns].corr(method='pearson')
```

```
Out[10]:                      trunk perimeter (cm)  height (m)  treetop diameter (m)
         trunk perimeter (cm)              1.000000    0.761796              0.809358
         height (m)                        0.761796    1.000000              0.744626
         treetop diameter (m)              0.809358    0.744626              1.000000
```

Now we take into account the tree types. We'll… - focus on the top-20 most frequent names only and plot the metrics per tree on a 4x5 matrix - and add some color to the plots

```
In [11]: fig, axes = plt.subplots(nrows=5, ncols=4, sharex=True, sharey=True,
                                  squeeze=False, figsize=[20,25])


         n = 0
         for tree in top_trees.index.to_list():
             plot = df[df['Name_lat']==tree].plot(
                 kind='scatter',
                 ax=axes[int(n/4),n%4],
                 title=tree,
                 x='trunk perimeter (cm)',
                 y='height (m)',
                 s='treetop diameter (m)', # show by point size
```

```
        c='treetop diameter (m)', # also indicated by color
        colormap='Spectral',
        norm=matplotlib.colors.LogNorm(vmin=1, vmax=25),
        colorbar=None)
    n += 1
plt.savefig('figures/trees_size_by_species.svg')
```



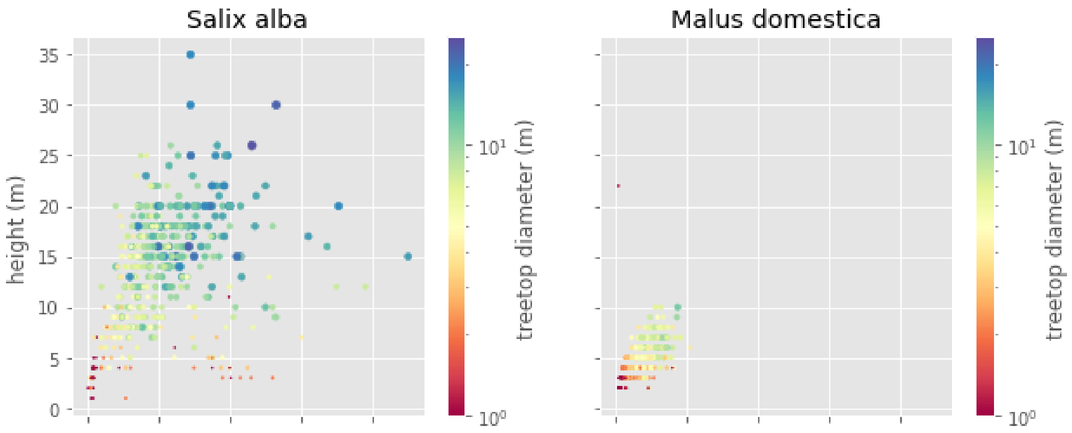
Notes about choosing the colormap for the treetop diameter: - the point size is hard to catch, while color is easier to discriminate (if not colorblind) - a spectral color map represents a continuous scale and allows for maximum discrimination - the range 1m - 25m (few trees reach 30m) is mapped on a logarithmic scale to make the smaller diameters (60% are 6m or smaller) look more different for small trees - of course, a logarithimic scale might be more difficult to read

See below the plot of willows and apple trees side by side. Try to change the color normalization!

```
In [12]: # distribution of treetop diameters
         df['treetop diameter (m)'].describe(percentiles=[i/20 for i in range(1, 20)])
```

```
Out[12]: count    15711.000000
         mean         6.124944
         std          3.883879
         min          0.000000
         5%           1.000000
         10%          1.000000
         15%          2.000000
         20%          2.000000
         25%          3.000000
         30%          4.000000
         35%          4.000000
         40%          5.000000
         45%          5.000000
         50%          6.000000
         55%          6.000000
         60%          6.000000
         65%          7.000000
         70%          8.000000
         75%          8.000000
         80%          9.000000
         85%         10.000000
         90%         12.000000
         95%         13.000000
         max         30.000000
         Name: treetop diameter (m), dtype: float64
```

```
In [13]: fig, axes = plt.subplots(nrows=1, ncols=2, sharex=True, sharey=True,
                                  squeeze=False, figsize=[10,4])


         n = 0
         for tree in ['Salix alba', 'Malus domestica']:
             df[df['Name_lat']==tree].plot(
                 kind='scatter',
                 ax=axes[0,n],
                 title=tree,
                 x='trunk perimeter (cm)',
                 y='height (m)',
                 s='treetop diameter (m)',
                 c='treetop diameter (m)',
                 colormap='Spectral',
                 norm=matplotlib.colors.LogNorm(vmin=1, vmax=25),
                 #norm=matplotlib.colors.Normalize(vmin=1, vmax=25),
                 colorbar=True)
             n += 1
```



… and a final look into correlations between height, trunk and treetop sizes - but now per tree type (we only take the 10 most common trees)

What could be the reasons that height and trunk perimeter show a quite low correlation for some tree types?

```
In [14]: df[df['Name_lat'].isin(top_trees.index.to_list()[0:10])] \
             .groupby(['Name_lat'])[metric_columns].corr(method='pearson')
```

```
Out[14]:                                          trunk perimeter (cm)  height (m)  \
         Name_lat
         Acer platanoides    trunk perimeter (cm)             1.000000    0.753107
                             height (m)                       0.753107    1.000000
                             treetop diameter (m)             0.841339    0.740206
         Acer pseudoplatanus trunk perimeter (cm)             1.000000    0.726985
                             height (m)                       0.726985    1.000000
                             treetop diameter (m)             0.807445    0.727782
         Betula pendula      trunk perimeter (cm)             1.000000    0.832376
                             height (m)                       0.832376    1.000000
                             treetop diameter (m)             0.865093    0.787967
         Fraxinus excelsior  trunk perimeter (cm)             1.000000    0.756545
                             height (m)                       0.756545    1.000000
                             treetop diameter (m)             0.874902    0.796629
         Malus domestica     trunk perimeter (cm)             1.000000    0.640308
                             height (m)                       0.640308    1.000000
                             treetop diameter (m)             0.801624    0.698249
         Platanus x acerifolia trunk perimeter (cm)           1.000000    0.684385
                             height (m)                       0.684385    1.000000
                             treetop diameter (m)             0.755265    0.862209
         Pyrus communis      trunk perimeter (cm)             1.000000    0.879626
                             height (m)                       0.879626    1.000000
                             treetop diameter (m)             0.914016    0.882129
         Quercus robur       trunk perimeter (cm)             1.000000    0.712255
                             height (m)                       0.712255    1.000000
                             treetop diameter (m)             0.821786    0.664783
         Salix alba          trunk perimeter (cm)             1.000000    0.381595
```

```
                      height (m)                    0.381595    1.000000
                      treetop diameter (m)          0.523723    0.716145
Tilia cordata         trunk perimeter (cm)          1.000000    0.832487
                      height (m)                    0.832487    1.000000
                      treetop diameter (m)          0.845741    0.819554

                                          treetop diameter (m)
Name_lat
Acer platanoides      trunk perimeter (cm)          0.841339
                      height (m)                    0.740206
                      treetop diameter (m)          1.000000
Acer pseudoplatanus   trunk perimeter (cm)          0.807445
                      height (m)                    0.727782
                      treetop diameter (m)          1.000000
Betula pendula        trunk perimeter (cm)          0.865093
                      height (m)                    0.787967
                      treetop diameter (m)          1.000000
Fraxinus excelsior    trunk perimeter (cm)          0.874902
                      height (m)                    0.796629
                      treetop diameter (m)          1.000000
Malus domestica       trunk perimeter (cm)          0.801624
                      height (m)                    0.698249
                      treetop diameter (m)          1.000000
Platanus x acerifolia trunk perimeter (cm)          0.755265
                      height (m)                    0.862209
                      treetop diameter (m)          1.000000
Pyrus communis        trunk perimeter (cm)          0.914016
                      height (m)                    0.882129
                      treetop diameter (m)          1.000000
Quercus robur         trunk perimeter (cm)          0.821786
                      height (m)                    0.664783
                      treetop diameter (m)          1.000000
Salix alba            trunk perimeter (cm)          0.523723
                      height (m)                    0.716145
                      treetop diameter (m)          1.000000
Tilia cordata         trunk perimeter (cm)          0.845741
                      height (m)                    0.819554
                      treetop diameter (m)          1.000000
```

## 2.4 Processing JSON

JSON is a standardized and common data format to store and interchange data independent from any programming language. JSON data types are numbers, Unicode strings, boolean values, the `null` value (`None`), arrays (Python lists) and objects (Python dictionaries). The JSON data types and the JSON syntax are similar to Python. But there are subtle differences and we use the json module of the Python standard library to read or write JSON data:

```
In [15]: import json

         data = [{"key1": "value1", "key2": 2, 'key3': [1, 2, 3]}, True, False, None, 17, 1.123]
         json_data = json.dumps(data)
         json_data

Out[15]: '[{"key1": "value1", "key2": 2, "key3": [1, 2, 3]}, true, false, null, 17, 1.123]'
```

```
In [16]: json.loads(json_data)

Out[16]: [{'key1': 'value1', 'key2': 2, 'key3': [1, 2, 3]},
          True,
          False,
          None,
          17,
          1.123]
```

```
In [17]: # load translations of tree names from a JSON file
         tree_translations = json.load(open('data/trees-wikispecies.json'))
```

```
In [18]: list(tree_translations.keys())[:10]

Out[18]: ['Platanus x acerifolia',
          'Platanus × hispanica',
          'Betula pendula',
          'Quercus robur',
          'Fraxinus excelsior',
          'Tilia cordata',
          'Malus domestica',
          'Salix alba',
          'Acer platanoides',
          'Acer pseudoplatanus']
```

### 2.4.1 Remark: Get Translations from Wikispecies

The translations of the tree names were obtained from the Wikispecies project via the Mediawiki API. We will later learn how to use an API (Application Programming Interface) and how to send requests over the internet. But here very short

```
import json
import requests

query_params = {
    'action': 'query',
    'format': 'json',
    'prop': 'iwlinks|langlinks|description',
    'lllimit': 200,
    'llprop': 'url|langname'
}

trees_wikispecies = {}

for tree in top_trees.index.to_list():
    if tree in trees_wikispecies:
        continue
    query_params['titles'] = tree.replace(' ', '_')
    response = requests.get('https://species.wikimedia.org/w/api.php',
                            params=query_params)
    trees_wikispecies[tree] = json.loads(response.text)

with open('trees-wikispecies.json', 'w') as fp:
    json.dump(trees_wikispecies, fp)
```

The script trees_wikispecies.py was used to create the data file

Because the data was queried from Wikispecies, the values per tree represent response to a query and we need to navigate into the result object to get the translations.

```
In [19]: tree_translations['Gleditsia triacanthos']
```

```
Out[19]: {'batchcomplete': '',
          'query': {'normalized': [{'from': 'Gleditsia_triacanthos',
             'to': 'Gleditsia triacanthos'}],
           'pages': {'124231': {'pageid': 124231,
             'ns': 0,
             'title': 'Gleditsia triacanthos',
             'iwlinks': [{'prefix': 'commons', '*': ''},
              {'prefix': 'commons', '*': 'Category:Gleditsia_triacanthos'},
              {'prefix': 'en', '*': 'International_Plant_Names_Index'},
              {'prefix': 'en', '*': 'Royal_Botanic_Gardens,_Kew'}],
             'langlinks': [{'lang': 'ar',
               'url': 'https://ar.wikipedia.org/wiki/%D8%BA%D9%84%D8%A7%D8%AF%D9%8A%D8%B4%D9%8A%D8%A9_%D8%AB%D9%84%D8%A7
               'langname': 'Arabic',
               '*': □□□□□□□□□' □□□□□□ {'□□□□□□□□,
              {'lang': 'arz',
               'url': 'https://arz.wikipedia.org/wiki/%D8%BA%D9%84%D8%A7%D8%AF%D9%8A%D8%B4%D9%8A%D9%87_%D8%AB%D9%84%D8%A
               'langname': 'Egyptian Arabic',
               '*': □□□□□□□□□' □□□□□□ {'□□□□□□□□,
              {'lang': 'az',
               'url': 'https://az.wikipedia.org/wiki/%C3%9C%C3%A7tikan_%C5%9Feytana%C4%9Fac%C4%B1',
               'langname': 'Azerbaijani',
               '*': 'Üçtikan şeytanağacı'},
              {'lang': 'ca',
               'url': 'https://ca.wikipedia.org/wiki/Ac%C3%A0cia_de_tres_punxes',
               'langname': 'Catalan',
               '*': 'Acàcia de tres punxes'},
              {'lang': 'ceb',
               'url': 'https://ceb.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Cebuano',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'cs',
               'url': 'https://cs.wikipedia.org/wiki/D%C5%99ezovec_trojtrnn%C3%BD',
               'langname': 'Czech',
               '*': 'Dřezovec trojtrnný'},
              {'lang': 'da',
               'url': 'https://da.wikipedia.org/wiki/Almindelig_tretorn',
               'langname': 'Danish',
               '*': 'Almindelig tretorn'},
              {'lang': 'de',
               'url': 'https://de.wikipedia.org/wiki/Amerikanische_Gleditschie',
               'langname': 'German',
               '*': 'Amerikanische Gleditschie'},
              {'lang': 'en',
               'url': 'https://en.wikipedia.org/wiki/Honey_locust',
               'langname': 'English',
               '*': 'Honey locust'},
              {'lang': 'eo',
               'url': 'https://eo.wikipedia.org/wiki/Kristodorna_gledi%C4%89io',
               'langname': 'Esperanto',
               '*': 'Kristodorna gledi̇ĉio'},
              {'lang': 'es',
               'url': 'https://es.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Spanish',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'eu',
               'url': 'https://eu.wikipedia.org/wiki/Akazia_hiruarantza',
               'langname': 'Basque',
               '*': 'Akazia hiruarantza'},
              {'lang': 'fa',
               'url': 'https://fa.wikipedia.org/wiki/%D9%84%DB%8C%D9%84%DA%A9%DB%8C_%D8%A2%D9%85%D8%B1%DB%8C%DA%A9%D8%A7
               'langname': 'Persian',
               '*': □□□□□' {'□□□□□□□□□,
              {'lang': 'fi',
               'url': 'https://fi.wikipedia.org/wiki/L%C3%A4nnenkolmioka',
               'langname': 'Finnish',
               '*': 'Lännenkolmioka'},
              {'lang': 'fr',
               'url': 'https://fr.wikipedia.org/wiki/F%C3%A9vier_d%27Am%C3%A9rique',
               'langname': 'French',
               '*': "Févier d'Amérique"},
              {'lang': 'ga',
               'url': 'https://ga.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Irish',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'hr',
               'url': 'https://hr.wikipedia.org/wiki/Ameri%C4%8Dka_gledi%C4%8Dija',
               'langname': 'Croatian',
               '*': 'Američka gledičija'},
              {'lang': 'hsb',
               'url': 'https://hsb.wikipedia.org/wiki/Ameriska_gledi%C4%8Dija',
               'langname': 'Upper Sorbian',
               '*': 'Ameriska gledičija'},
              {'lang': 'hu',
               'url': 'https://hu.wikipedia.org/wiki/T%C3%B6vises_lep%C3%A9nyfa',
               'langname': 'Hungarian',
               '*': 'Tövises lepényfa'},
              {'lang': 'hy',
               'url': 'https://hy.wikipedia.org/wiki/%D4%B3%D5%AC%D5%A5%D5%A4%D5%AB%D5%B9%D5%A1',
               'langname': 'Armenian',
               '*': '□□□□□□□'},
              {'lang': 'it',
               'url': 'https://it.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Italian',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'kbd',
               'url': 'https://kbd.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D1%8D%D0%B6%D1%8B%D0%B3',
               'langname': 'Kabardian',
               '*': 'Банэжыг'},
              {'lang': 'kk',
               'url': 'https://kk.wikipedia.org/wiki/%D2%AE%D1%88%D1%82%D1%96%D0%BA%D0%B5%D0%BD%D0%B4%D1%96_%D2%9B%D0%B0
               'langname': 'Kazakh',
               '*': 'Үштікенді қарамала'},
              {'lang': 'lt',
               'url': 'https://lt.wikipedia.org/wiki/Tridygl%C4%97_gledi%C4%8Dija',
               'langname': 'Lithuanian',
               '*': 'Tridyglė gledičija'},
              {'lang': 'nl',
               'url': 'https://nl.wikipedia.org/wiki/Valse_christusdoorn',
               'langname': 'Dutch',
               '*': 'Valse christusdoorn'},
              {'lang': 'no',
               'url': 'https://no.wikipedia.org/wiki/Korstorn',
               'langname': 'Norwegian',
               '*': 'Korstorn'},
              {'lang': 'nv',
               'url': 'https://nv.wikipedia.org/wiki/Naazt%C3%A1n%C3%AD',
               'langname': 'Navajo',
               '*': 'Naaztání'},
              {'lang': 'pl',
               'url': 'https://pl.wikipedia.org/wiki/Glediczja_tr%C3%B3jcierniowa',
               'langname': 'Polish',
               '*': 'Glediczja trójcierniowa'},
              {'lang': 'pms',
               'url': 'https://pms.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Piedmontese',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'pt',
               'url': 'https://pt.wikipedia.org/wiki/Gleditsia_triacanthos',
               'langname': 'Portuguese',
               '*': 'Gleditsia triacanthos'},
              {'lang': 'ro',
               'url': 'https://ro.wikipedia.org/wiki/Gl%C4%83di%C8%9B%C4%83',
               'langname': 'Romanian',
               '*': 'Glădiță'},
              {'lang': 'ru',
               'url': 'https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B5%D0%B4%D0%B8%D1%87%D0%B8%D1%8F_%D1%82%D1%80%D1%91
               'langname': 'Russian',
               '*': 'Гледичия трёхколючковая'},
              {'lang': 'sr',
               'url': 'https://sr.wikipedia.org/wiki/%D0%A2%D1%80%D0%BD%D0%BE%D0%B2%D0%B0%D1%86_(%D0%B1%D0%B8%D1%99%D0%B
               'langname': 'Serbian',
                      '*': 'Трновац (биљка)'},
                     {'lang': 'sv',
                      'url': 'https://sv.wikipedia.org/wiki/Gleditsia_triacanthos',
                      'langname': 'Swedish',
                      '*': 'Gleditsia triacanthos'},
                     {'lang': 'uk',
                      'url': 'https://uk.wikipedia.org/wiki/%D0%93%D0%BB%D0%B5%D0%B4%D0%B8%D1%87%D1%96%D1%8F_%D0%BA%D0%BE%D0%BB
                      'langname': 'Ukrainian',
                      '*': 'Гледичія колюча'},
                     {'lang': 'vi',
                      'url': 'https://vi.wikipedia.org/wiki/B%E1%BB%93_k%E1%BA%BFt_ba_gai',
                      'langname': 'Vietnamese',
                      '*': 'B□ k□t ba gai'},
                     {'lang': 'war',
                      'url': 'https://war.wikipedia.org/wiki/Gleditsia_triacanthos',
                      'langname': 'Waray',
                      '*': 'Gleditsia triacanthos'},
                     {'lang': 'zh',
                      'url': 'https://zh.wikipedia.org/wiki/%E7%BE%8E%E5%9B%BD%E7%9A%82%E8%8D%9A',
                      'langname': 'Chinese',
                      '*': '□□□□'}],
                    'description': 'species of tree',
                    'descriptionsource': 'central'}}}}
```

```
In [20]: languages = ['fr', 'ru', 'ar']

         # add new columns to cadastre table
         for lang in languages:
             df['Name_' + lang] = pd.Series([''] * df.shape[0], index=df.index)

         for tree in top_trees.index.to_list():
             if tree not in tree_translations:
                 continue
             for _id, result in tree_translations[tree]['query']['pages'].items():
                 for lang in languages:
                     for langlink in result['langlinks']:
                         if langlink['lang'] in languages:
                             # print(tree, langlink)
                             # add the translation to the table
                             df.loc[df['Name_lat']==tree, 'Name_' + langlink['lang']] = langlink['*']
```

```
In [21]: name_cols = ['Name_lat', 'Name_dt', *['Name_' + lang for lang in languages]]

         tree_name_translation = df.loc[df['Name_lat'].isin(top_trees.index), name_cols]
         tree_name_translation['count'] = 1
         tree_name_translation.groupby(name_cols).sum().sort_values('count', ascending=False)
```

```
Out[21]:
         Name_lat                              Name_dt                          Name_fr                    Name_ru
         Platanus x acerifolia                 Platane                          Platane commun             Платан кленол
         Betula pendula                        Birke, Sand-Birke                Bouleau verruqueux         Берёза повисл
         Quercus robur                         Eiche, Stiel-Eiche, Sommer-Eiche Chêne pédonculé            Дуб черешчаты
```

```
Fraxinus excelsior                    Esche, Esche gemeine             Frêne élevé                Ясень обыкнов
Tilia cordata                         Linde, Winter-Linde              Tilleul à petites feuilles Липа сердцеви
Malus domestica                       Kultur-Apfel                     Pommier domestique         Яблоня домашн
Salix alba                            Weide, Silber-Weide              Salix alba                 Ива белая
Acer platanoides                      Ahorn, Spitz-Ahorn               Érable plane               Клён остролис
Acer pseudoplatanus                   Ahorn, Berg-Ahorn                Érable sycomore            Клён белый
Pyrus communis                        Birne, Holz-Birne                Poirier commun             Груша обыкнов
Carpinus betulus                      Weißbuche, Hainbuche             Charme commun              Граб обыкнове
Acer campestre                        Ahorn, Feld-Ahorn                Érable champêtre           Клён полевой
Juglans regia                         Nussbaum, Walnuss                Noyer commun               Орех грецкий
Aesculus hippocastanum                Rosskastanie                     Aesculus hippocastanum     Конский кашта
Fagus sylvatica                       Buche, Rotbuche                  Hêtre commun               Бук европейск
Fraxinus excelsior 'Westhof's Glorie' Straßen-Esche                    Frêne élevé                Ясень обыкнов
Tilia platyphyllos                    Linde, Sommer-Linde              Tilleul à grandes feuilles Липа крупноли
Prunus avium                          Kirsche, Vogel-Kirsche           Prunus avium               Черешня
Tilia cordata 'Greenspire'            Linde "Greespire"                Tilleul à petites feuilles Липа сердцеви
Gleditsia triacanthos 'Inermis'       Dornenlose Gleditschie           Févier d'Amérique          Гледичия трёх
```

### 2.4.2 Remark: Advanced JSON processing with `jq`

Processing deeply nested JSON is cumbersome because the Pythone code may also require nested loops or recursive function calls. The JSON processor jq allows for easy processing (filter and transform) of JSON data. There exist Python bindings but it is primarily a command-line tool:

1. download one tree record from Wikispecies using curl:

   ```
   curl 'https://species.wikimedia.org/w/api.php?action=query&format=json&prop=iwlinks|langlinks|description&lllimit=
       >data/wikispecies-quercus-robur.json
   ```

2. inspect the JSON result (nicely formatted):

   ```
   jq . <data/wikispecies-quercus-robur.json
   ```

3. step by step drill down to extract the data

   ```
   jq -r '.["query"]["pages"][]["langlinks"][] | [.["lang","*"]] | join("\t")' \
       <data/quercus_robur-wikimedia-species.json \
     | head
   ```

   which will extract a map <language,name_of_tree>:

   ```
   af      Steeleik
   ar      □□□□□□ □□□
   arz     □□□□□□ □□□
   ast     Quercus robur
   az      Yay palıdı
   azb     □□□ □□□□□□
   bat-smg Ōžouls
   be      Дуб звычайны
   bg      Обикновен дъб
   bs      Hrast lužnjak
   ```

Using the jq Python bindings you could extract the data by …

```
In [22]: import jq

         q = jq.compile('.["query"]["pages"][]["langlinks"][] | [.["lang","*"]]')
         translations_quercus_robur = dict(
             q.input(
                 json.load(
                     open('data/quercus_robur-wikimedia-species.json'))).all())
         translations_quercus_robur['fr']

Out[22]: 'Chêne pédonculé'
```

## 2.5 Mapping Geographic Data

To show the trees on the map we use the package Folium. See also the quickstart and API docs.

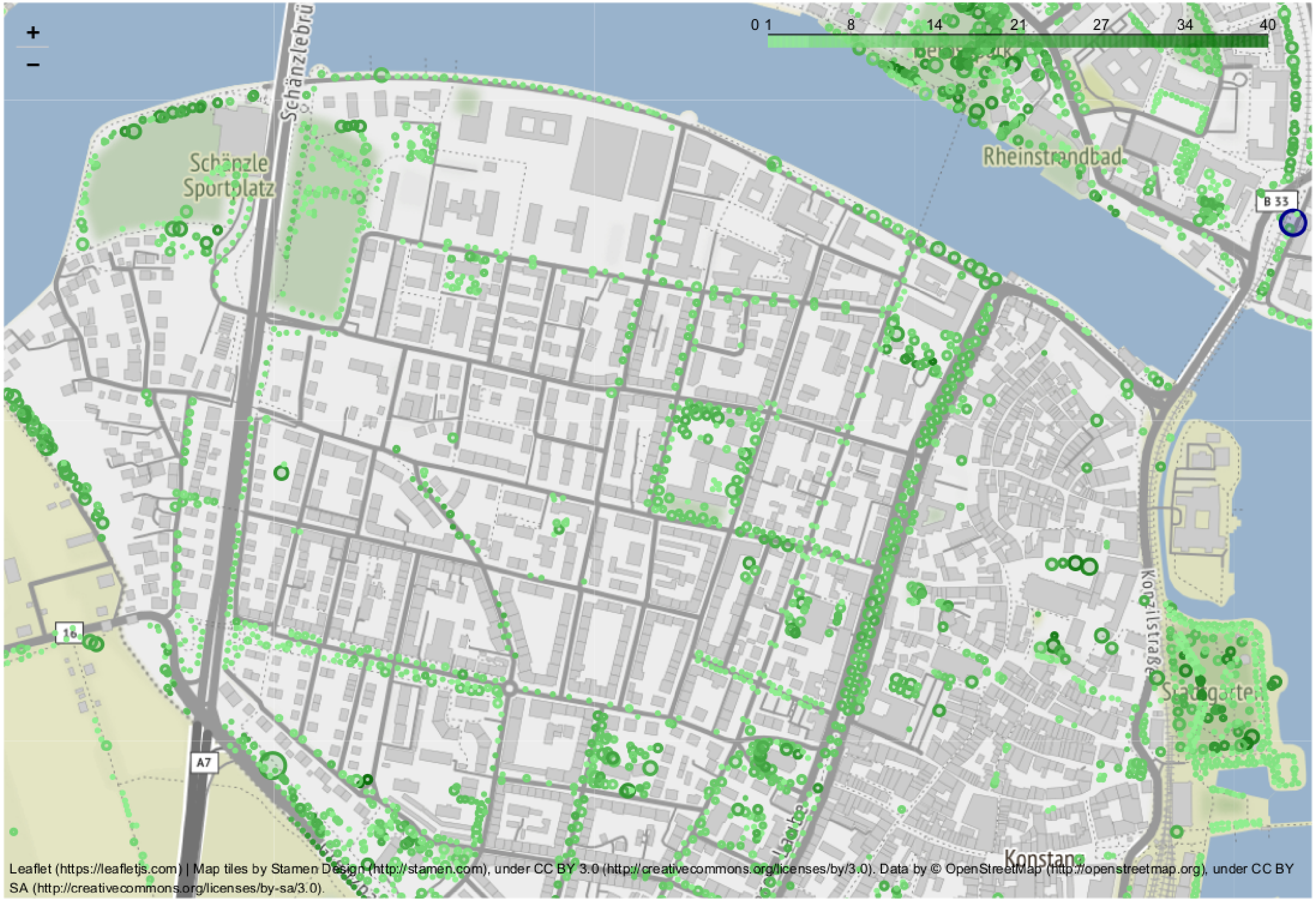
```
In [23]: import folium
         import math
         import branca.colormap as cm

         map = folium.Map(location=[47.66336, 9.17598],
                          tiles = 'Stamen Terrain',
                          zoom_start=16)

         colormap = cm.LinearColormap(colors=['lightgreen','darkgreen'],
                                      vmin=1, vmax=40).to_step(n=12)

         def color_height(height):
             if 1.0 <= height <= 40.0:
                 return colormap(height)
             else:
                 return 'darkblue'

         def map_tree(row):
             marker = folium.CircleMarker(
                         location=(row['Y'], row['X']),
                         tooltip=folium.Tooltip(row['Name_lat']),
                         radius=row['treetop diameter (m)']/4,
                         fill=True,
                         color=color_height(row['height (m)']),
                       )
             marker.add_to(map)

         # for development: select a subset because plotting 16k trees takes long
         #   df[df['location']=='Münsterplatz (27)']
         #   df.head(500)

         df.apply(map_tree, axis=1)
```

```
map.add_child(colormap, name='height (m)')
map
```

```
Out[23]: <folium.folium.Map at 0x7fc481f76dc0>
```

## 2.6 Links and References

- Pandas getting started
- matplotlib cheatsheet (beginners sheet)
- processing JSON data from the course “Data Analysis and Visualization with Python for Social Scientists” (https://datacarpentry.org/python-socialsci/)



Leaflet (https://leafletjs.com) | Map tiles by Stamen Design (http://stamen.com), under CC BY 3.0 (http://creativecommons.org/licenses/by/3.0). Data by © OpenStreetMap (http://openstreetmap.org), under CC BY SA (http://creativecommons.org/licenses/by-sa/3.0).

# 3 The Twitter API

- what is an API?
- get access to the Twitter API
- use a client: DocNow/twarc
- tweets, user timelines, followers, trends
- text statistics, language, sentiment

## 3.1 What is an API?

The Application Programming Interface (API) allows computer programs to interact with software libraries (the pandas API) or services (eg. Twitter or Mediawiki) in a similar way a user interface allows humans to interact with computers.

## 3.2 Why social media and why Twitter?

Social media is an important data source for social science research:

> social media platforms are, in one sense, vast collections of freely available unscripted opinions, experiences and insights on any number of topics“ (Phillip D. Brooker Section **??**)

The Twitter API is easy to set up and usage is less restrictive compared to the APIs of other social media platforms.

## 3.3 Get Access to the Twitter API

Before apply for access you definitely should read about the restrictions on using and sharing Twitter data. You may also start browsing the API documentation.

After having registered for an API account, you need to follow the documentation about getting started.

Note that

- the registration and setup process requires some time
- the examples given below can only replayed if you have registered for the Twitter API

## 3.4 Install and Setup Twarc

Twarc is

> a command line tool and Python library for archiving Twitter JSON data. Each tweet is represented as a JSON object that is exactly what was returned from the Twitter API. Tweets are stored as line-oriented JSON. twarc will handle Twitter API's rate limits for you. In addition to letting you collect tweets twarc can also help you collect users, trends and hydrate tweet ids. (from the Twarc documentation)

Installation and setup is done in just two steps:

- install

  ```
  pip install twarc
  ```

- configure twarc to use your Twitter API credentials

  ```
  twarc configure
  ```

  or for version 2 of the API

  ```
  twarc2 configure
  ```

See the Twarc documentation for more details and also for first examples to work with Twarc.

We will use twarc2 to access version 2 of the Twitter API. We focus on the command-line tool only - there is no need to use the Twarc API unless there are very specific requirements or using Twarc is part of a more complex data acquisition process.

First, we call `twarc2 --help` to figure out which options and commands are provided:

```
In [1]: !twarc2 --help
```

```
Usage: twarc2 [OPTIONS] COMMAND [ARGS]…

  Collect data from the Twitter V2 API.

Options:
  --consumer-key TEXT         Twitter app consumer key (aka "App Key")
  --consumer-secret TEXT      Twitter app consumer secret (aka "App Secret")
  --access-token TEXT         Twitter app access token for user
                              authentication.
  --access-token-secret TEXT  Twitter app access token secret for user
                              authentication.
  --bearer-token TEXT         Twitter app access bearer token.
  --app-auth / --user-auth    Use application authentication or user
                              authentication. Some rate limits are higher with
                              user authentication, but not all endpoints are
                              supported.  [default: app-auth]
  -l, --log TEXT
  --verbose
  --metadata / --no-metadata  Include/don't include metadata about when and
                              how data was collected.  [default: metadata]
  --config FILE               Read configuration from FILE.
```

```
  --help                      Show this message and exit.

Commands:
  compliance-job  Create, retrieve and list batch compliance jobs for…
  configure       Set up your Twitter app keys.
  conversation    Retrieve a conversation thread using the tweet id.
  conversations   Fetch the full conversation threads that the input…
  counts          Return counts of tweets matching a query.
  dehydrate       Extract tweet or user IDs from a dataset.
  flatten         "Flatten" tweets, or move expansions inline with tweet…
  followers       Get the followers for a given user.
  following       Get the users that a given user is following.
  hydrate         Hydrate tweet ids.
  mentions        Retrieve max of 800 of the most recent tweets…
  places          Search for places by place name, geo coordinates or ip…
  sample          Fetch tweets from the sample stream.
  search          Search for tweets.
  searches        Execute each search in the input file, one at a time.
  stream          Fetch tweets from the live stream.
  stream-rules    List, add and delete rules for your stream.
  timeline        Retrieve recent tweets for the given user.
  timelines       Fetch the timelines of every user in an input source of…
  tweet           Look up a tweet using its tweet id or URL.
  users           Get data for user ids or usernames.
  version         Return the version of twarc that is installed.
```

… and to get the command-specific options:

```
In [2]: !twarc2 timeline --help
```

```
Usage: twarc2 timeline [OPTIONS] USER_ID [OUTFILE]

  Retrieve recent tweets for the given user.

Options:
  --start-time [%Y-%m-%d|%Y-%m-%dT%H:%M:%S]
                                  Match tweets created after UTC time (ISO
                                  8601/RFC 3339),  e.g.  --start-time
                                  "2021-01-01T12:31:04"
  --end-time [%Y-%m-%d|%Y-%m-%dT%H:%M:%S]
                                  Match tweets sent before UTC time (ISO
                                  8601/RFC 3339),  e.g.  --end-time
                                  "2021-01-01T12:31:04"
  --since-id INTEGER              Match tweets sent after tweet id
  --until-id INTEGER              Match tweets sent prior to tweet id
  --use-search                    Use the search/all API endpoint which is not
                                  limited to the last 3200 tweets, but
                                  requires Academic Product Track access.
  --exclude-retweets              Exclude retweets from timeline
  --exclude-replies               Exclude replies from timeline
  --no-context-annotations        By default twarc gets all available data.
```

```
                                This leaves out context annotations (Twitter
                                API limits --max-results to 100 if these are
                                requested). Setting this makes --max-results
                                500 the default. NOTE: This argument is
                                mutually exclusive with  arguments:
                                [--tweet-fields, --place-fields, --user-
                                fields, --expansions, --media-fields,
                                --counts-only, --minimal-fields, --poll-
                                fields].
  --minimal-fields              By default twarc gets all available data.
                                This option requests the minimal retrievable
                                amount of data - only IDs and object
                                references are retrieved. Setting this makes
                                --max-results 500 the default. NOTE: This
                                argument is mutually exclusive with
                                arguments: [--tweet-fields, --place-fields,
                                --user-fields, --media-fields, --expansions,
                                --counts-only, --no-context-annotations,
                                --poll-fields].
  --expansions TEXT             Comma separated list of expansions to
                                retrieve. Default is all available.
  --tweet-fields TEXT           Comma separated list of tweet fields to
                                retrieve. Default is all available.
  --user-fields TEXT            Comma separated list of user fields to
                                retrieve. Default is all available.
  --media-fields TEXT           Comma separated list of media fields to
                                retrieve. Default is all available.
  --place-fields TEXT           Comma separated list of place fields to
                                retrieve. Default is all available.
  --poll-fields TEXT            Comma separated list of poll fields to
                                retrieve. Default is all available.
  --hide-progress               Hide the Progress bar. Default: show
                                progress, unless using pipes.
  --limit INTEGER               Maximum number of tweets to return
  --help                        Show this message and exit.
```

## 3.5 Analyzing Tweets from a User Timeline

For a first trial we download 500 tweets from the timeline of [@EXCInequality](https://twitter.com/EXCInequality) and save it to a file:

```
twarc2 timeline EXCInequality --limit 500 >data/twitter/timeline.EXCInequality.jsonl
```

Note that the Twitter developer terms of use do not allow to share the content of tweets. That's why not tweet data is included in this repository, or only in aggregations on the level of words. You need to apply for API access in order to replay the examples.

```
In [3]: import json
        import pandas as pd
        def load_tweets(file):
            tweets = []
            with open(file) as stream:
                for line in stream:
                    api_response = json.loads(line)
                    for tweet in api_response['data']:
                        tweets.append(tweet)
            return tweets

        tweets = load_tweets('data/twitter/timeline.EXCInequality.jsonl')

        len(tweets)
```

```
Out[3]: 500
```

Let's look into the one of the tweets to understand the data structure and compare this with the tweet object model documentation.

```
In [4]: #tweets[1]
```

Note: it's possible to load the tweets into a pandas dataframe but some cells still contain nested JSON elements:

```
df = pd.DataFrame(tweets)
```

Pandas provides normalization routines to flatten nested data.

But we will work with the JSON data directly and first extract which hashtags are frequently used in the Tweets of [@EXCInequality](https://twitter.com/EXCInequality):

```
In [5]: from collections import Counter

        aggregation_on = ('hashtags', 'tag')

        # instead of hashtags count other items in the `entities` object:
        # aggregation_on = ('annotations', 'normalized_text')
        # aggregation_on = ('mentions', 'username')
        # aggregation_on = ('urls', 'url')

        counts = Counter()


        for t in tweets:
            if 'entities' not in t:
                continue
            if aggregation_on[0] in t['entities']:
                for obj in t['entities'][aggregation_on[0]]:
                    counts[obj[aggregation_on[1]]] += 1

        counts.most_common()[0:20]
```

```
Out[5]: [('inequality', 28),
         ('UniKonstanz', 25),
         ('jobsinacademia', 21),
         ('COVID19', 20),
         ('jobsinscience', 20),
         ('MeetTheCluster', 17),
         ('ClusterColloquium', 13),
         ('PolicyPaper', 13),
         ('InequalityMagazine', 9),
         ('research', 8),
         ('Homeoffice', 8),
         ('scicomm', 6),
         ('outsoon', 6),
         ('InequalityConf', 5),
         ('ThePoliticsOfInequality', 5),
         ('Konstanz', 5),
         ('PGS21', 4),
         ('Ungleichheit', 4),
         ('NewPublication', 4),
         ('Exzellenzcluster', 4)]
```

### 3.5.1 Find the Most Commonly Used Words in Tweets

We will now look into the tweets itself and - split the text into words - count word occurrences and - generate a word cloud to visualize word frequencies or the "importance" of words

```
In [6]: words = Counter()

        for t in tweets:
            for word in t['text'].split(' '):
                words[word] += 1

        words.most_common()[0:10]
```

```
Out[6]: [('the', 306),
         ('in', 244),
         ('of', 235),
         ('to', 234),
         ('and', 200),
         ('RT', 193),
         ('a', 179),
         ('on', 143),
         ('our', 123),
         ('for', 120)]
```

This initial attempt shows that we need to skip over the most common functional words, in text processing called "stop words".

```
In [7]: from stop_words import get_stop_words

        stop_words = set(get_stop_words('en'))
        stop_words.update(get_stop_words('de'))
        def word_counts(tweets):
            words = Counter()
            for t in tweets:
                for word in t['text'].split(' '):
                    word = word.lower()
                    if word in stop_words:
                        continue
                    words[word] += 1
            return words

        word_counts(tweets).most_common()[0:25]
```

```
Out[7]: [('rt', 193),
         ('&amp;', 95),
         ('-', 82),
         ('cluster', 59),
         ('@unikonstanz', 49),
         ('@unikonstanz:', 48),
         ('new', 44),
         ('research', 41),
         ('@excinequality', 36),
         ('us', 30),
         ('paper', 30),
         ('–', 28),
         ('work', 26),
         ('just', 26),
         ('social', 25),
         ('project', 24),
         ('inequality', 22),
         ('#unikonstanz', 22),
         ('welcome', 22),
         ('great', 21),
         ('job', 20),
         ('take', 20),
         ('here:', 20),
         ('talk', 20),
         ('#inequality', 20)]
```

… and we also need to skip mentions, hashtags, URLs and everything which does not look like a word. We simply skip all words containing any other characters except letters (alphabetical characters). Note that this approach is simple and effective but it will also remove words such as "Covid-19".

```
In [8]: stop_words.add('rt') # retweet


        def word_counts(tweets):
            words = Counter()
            for t in tweets:
                for word in t['text'].split(' '):
                    word = word.lower()
                    if word in stop_words:
```

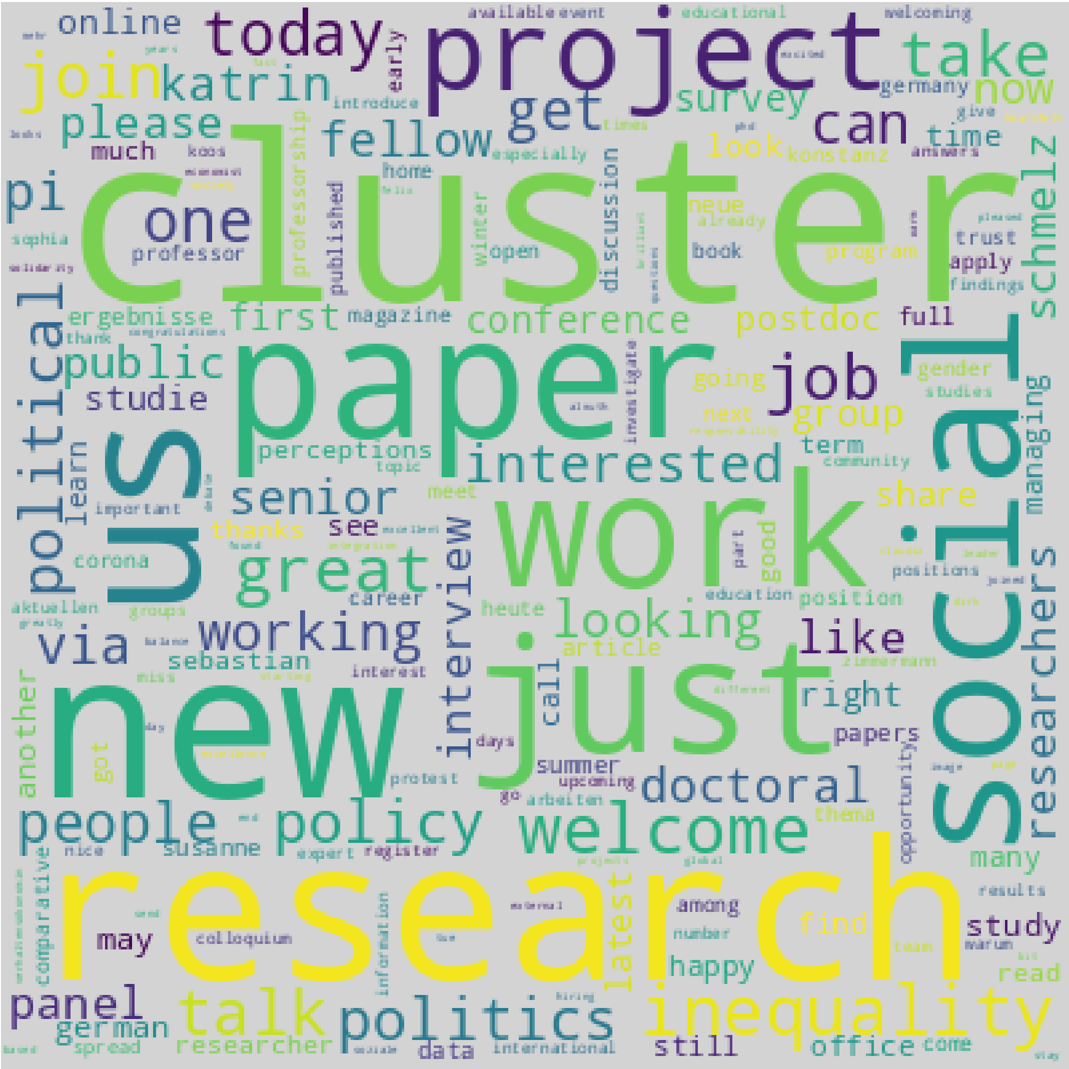
```
                continue
            if not word.isalpha():
                # skip words containing non-alphabetical characters
                continue
            words[word] += 1
    return words

word_counts(tweets).most_common()[0:25]
```

```
Out[8]: [('cluster', 59),
         ('new', 44),
         ('research', 41),
         ('us', 30),
         ('paper', 30),
         ('work', 26),
         ('just', 26),
         ('social', 25),
         ('project', 24),
         ('inequality', 22),
         ('welcome', 22),
         ('great', 21),
         ('job', 20),
         ('take', 20),
         ('talk', 20),
         ('today', 19),
         ('join', 19),
         ('one', 19),
         ('politics', 18),
         ('policy', 18),
         ('political', 18),
         ('people', 17),
         ('can', 17),
         ('get', 17),
         ('working', 16)]
```

Word clouds are generated using the wordcloud package, see also: - API docs of the WordCloud class - more examples

```
In [9]: from wordcloud import WordCloud

        wordcloud = WordCloud(width=400, height=400,
                              background_color='lightgrey') \
                    .generate_from_frequencies(word_counts(tweets))

        wordcloud.to_image()
```

```
Out[9]:
```



### 3.5.2 Words Used by the Official Twitter Accounts of German Political Parties

Let's download tweets from the official Twitter accounts of the political parties currently. We limit the download to the 1,000 most recent tweets.

Solution 1: wrap the calls of Twarc into a loop in the command-line shell

```
mkdir -p data/twitter/ppart/timeline/
for pp in CDU CSU spdde Die_Gruenen fdp dieLinke AfD; do
    twarc2 timeline $pp \
           --limit 1000 \
           >data/twitter/ppart/timeline/$pp.jsonl
done
```

Solution 2: pure Python - you'd need to add your Twitter authentication token(s) - and need to handle limits while iterating over result pages

```
In [10]: %%script false --no-raise-error

         from twarc.client2 import Twarc2
         from twarc.expansions import ensure_flattened


         parties = 'CDU CSU spdde Die_Gruenen fdp dieLinke AfD'.split()

         tw = Twarc2(bearer_token='...')

         for party in parties:

             with open('data/twitter/ppart/timeline/' + party + '.jsonl', 'w') as result_stream:

                 n_tweets = 0

                 results = tw.timeline(party, max_results=100)

                 for page in results:

                     # dump as JSON
                     print(json.dumps(page), file=result_stream)

                     tweets = ensure_flattened(page)

                     n_tweets += len(tweets)
                     if n_tweets >= 1000:
                         break

                     # alternatively, process the tweets directly
                     for tweet in tweets:
                         #print(tweet['text'])
```

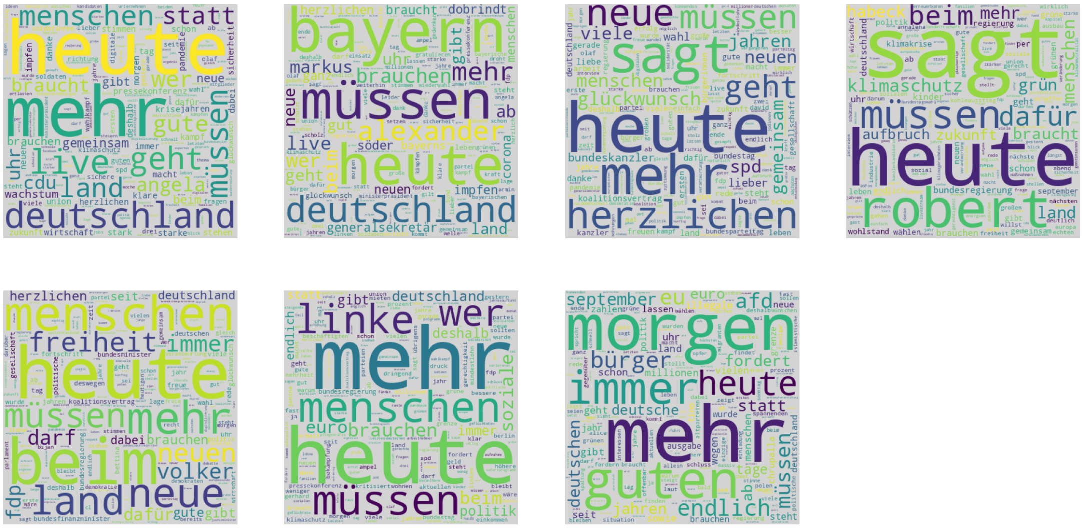
Now we load the data in Python, extract the word counts and generate the word clouds...

```
In [11]: parties = 'CDU CSU spdde Die_Gruenen fdp dieLinke AfD'.split()

         words = {}

         for party in parties:
             tweets = load_tweets('data/twitter/ppart/timeline/%s.jsonl' % party)
             words[party] = word_counts(tweets)
             # show some stats
             print(party, len(tweets), 'tweets')
             print('\t', word_counts(tweets).most_common()[0:3])
```

```
CDU 1099 tweets
         [('heute', 113), ('mehr', 99), ('deutschland', 67)]
CSU 1000 tweets
         [('bayern', 90), ('heute', 74), ('müssen', 67)]
spdde 1094 tweets
         [('heute', 93), ('mehr', 57), ('sagt', 51)]
```

```
Die_Gruenen 1000 tweets
         [('sagt', 137), ('heute', 112), ('robert', 104)]
fdp 1000 tweets
         [('heute', 85), ('beim', 64), ('menschen', 50)]
dieLinke 1000 tweets
         [('mehr', 73), ('heute', 61), ('menschen', 50)]
AfD 1000 tweets
         [('mehr', 50), ('morgen', 45), ('guten', 42)]
```

```
In [12]: import math
         import matplotlib.pyplot as plt

         ncols = 4
         nrows = math.ceil(len(parties)/ncols)

         fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=[ncols*10,nrows*10])

         n = 0
         for party in parties:
             wordcloud = WordCloud(width=400, height=400,
                                   background_color='lightgrey') \
                         .generate_from_frequencies(words[party])
             axis = axes[int(n/ncols),n%ncols]
             axis.imshow(wordcloud)
             axis.axis('off') # do not show x/y scale
             n += 1

         # fill the grid with empty subplots
         while n < (ncols*nrows):
             axes[int(n/ncols),n%ncols].axis('off') # do not show x/y scale
             n += 1

         plt.show()
```



## 3.6 Links and References

- Phillip Brooker's book Programming with Python for Social Scientists includes a chapter about using the Twitter API
- https://developer.twitter.com/en/products/twitter-api
- https://twitter.com/TwitterAPI
- https://developer.twitter.com/en/use-cases/do-research
- https://developer.twitter.com/en/products/twitter-api/academic-research
- https://twarc-project.readthedocs.io/en/latest/
- https://scholarslab.github.io/learn-twarc/
- https://github.com/DocNow/twarc/tree/main/utils (for JSON data downloaded using the v1 API)

# 4 Web Scraping

- HTTP requests
- HTML, XML, DOM, CSS selectors, XPath
- browser automation
- cleanse and export extracted data

Web-based (or browser-based) user interfaces are ubiquitous

- web browser as universal platform to run software (at least, the user interface)
- if a human is able to access information in WWW using a web browser, also a computer program can access the same information and automatically extract it
- challenges: navigate a web page, execute user interaction (mouse clicks, forms)
- real challenges: login forms, captchas, IP blocking, etc.
    - not covered here
    - also: ethical considerations whether or not to get around access blocking
- well-defined technology stack
    - HTTP
    - HTML / XML
    - DOM
    - CSS
    - XPath
    - JavaScript

## 4.1 Web Browser

- render HTML page to make it readable for humans
- basic navigation in the WWW (follow links)
- text-based browsers

```
lynx https://www.bundestag.de/parlament/fraktionen/cducsu
```

- modern graphical browsers
    - interpret JavaScript
    - show multi-media content
    - run "web applications"
- headless vs. headful browsers
    - headful: graphical user interface attached
    - headless
        * controlled programmatically or via command-line
        * interaction but no mandatory page rendering (saves resources: CPU, RAM)

```
In [1]: (1 + 2
         + 3)

Out[1]: 6
```

### 4.1.1 Tip: Extract Text and Links Using a Text-Based Browser

Tip: text-based browsers usually have an option to “dump” the text and/or link lists into a file, e.g.

```
lynx -dump https://www.bundestag.de/parlament/fraktionen/cducsu \
    >data/bundestag/fraktionen.cducsu.txt
```

### 4.1.2 Tip: Explore Web Pages and Web Technologies using the Developer Tool of your Web Browser

Modern web browsers (Firefox, Chromium, IE, etc.) include a set of web development tools. Originally addressed to web developers to test and debug the code (HTML, CSS, Javascript) used to build a web site, the browser web developer tools are the easiest way to explore and understand the technologies used to build a web site. The initial exploration later helps to scrape data from the web site.

### 4.1.3 Browser Automation

- load a page by URL including page dependencies (CSS, Javascript, images, media)
- simulate user interaction (clicks, input, scrolling)
- take screenshots
- access the DOM tree or the HTML modified by executed Javascript and user interactions from/in the browser to extract data

## 4.2 Process HTML Pages in Python

- requests to fetch pages via HTTP
- beautifulsoup to parse HTML

```
In [2]: import requests

        request_url = 'https://www.bundestag.de/parlament/fraktionen/cducsu'
        response = requests.get(request_url)

        response

Out[2]: <Response [200]>

In [3]: response.headers

Out[3]: {'date': 'Sun, 30 Jan 2022 20:52:48 GMT', 'content-type': 'text/html;charset=UTF-8', 'content-length': '19906',

In [4]: response.status_code
```

```
Out[4]: 200
```

```
In [5]: !pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in ./.venv/lib/python3.9/site-packages (4.10.0)
Requirement already satisfied: soupsieve>1.2 in ./.venv/lib/python3.9/site-packages (from beautifulsoup4) (2.3.1)
```

```
In [6]: from bs4 import BeautifulSoup

        html = BeautifulSoup(response.text)

        html.head.title  # tree-style path addressing of HTML elements
```

```
Out[6]: <title>Deutscher Bundestag - CDU/CSU-Fraktion</title>
```

Note: the HTML document can be represented as a tree structure aka. DOM tree:

```
html
├── head
|   ├── meta
|   |   └── @charset=utf-8
|   └── title
|       └── ...(text)
└── body
    └── ...
```

The tree above is an equivalent representation for the HTML snippet

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Deutscher Bundestag - CDU/CSU-Fraktion</title>
</head>
<body>
  ...
</body>
</html>
```

```
In [7]: # access the plain text of an HTML element
        # (inside the opening and closing tag)
        html.head.title.text
```

```
Out[7]: 'Deutscher Bundestag - CDU/CSU-Fraktion'
```

```
In [8]: # beautifulsoup also allows to select elements by tag name without a tree-like path

        html.find('title').text
```

```
Out[8]: 'Deutscher Bundestag - CDU/CSU-Fraktion'
```

```
In [9]: # or if a tag is expected to appear multiple times:
        # select all `a` elements and show the first three
        html.findAll('a')[0:3]
```

```
Out[9]: [<a class="sr-only sr-only-focusable" href="#main" title="Direkt zum Hauptinhalt springen">Direkt zum Hauptinhal
         <a class="sr-only sr-only-focusable" href="#main-menu" title="Direkt zum Hauptmenü springen">Direkt zum Hauptme
         <a href="https://www.bundestag.de/webarchiv" hreflang="de" lang="de" title="Archiv" xml:lang="de">
         <span class="sr-only-sm-down">Archiv</span>
         <span class="visible-xs-inline">Archiv</span>
         </a>]
```

```
In [10]: # selection by CSS class name
         html.find(class_='bt-standard-content')
```

```
Out[10]: <article class="bt-artikel col-xs-12 col-md-6 bt-standard-content">
         <h3 class="bt-artikel__title">CDU/CSU-Fraktion</h3>
         <div class="bt-bild-standard bt-bild-max" data-nosnippet="true">
         <img alt="Beschilderung einer Tür im Bereich der CDU/CSU-Fraktion auf der Fraktionsebene im Reichstagsgebäude."
         <span class="bt-bild-info-icon"><i aria-hidden="true" class="icon-info-1"></i>
         </span>
         <div class="bt-bild-info-text">
         <span class="bt-bild-info-close">
         <i aria-hidden="true" class="icon-close"></i>
         </span>
         <p>
                 Logo der CDU/CSU-Fraktion</p>
         <p>© DBT/Axel Hartmann Fotografie</p>
         </div>
         </div>
         <div class="bt-standard-content">
         <h4>Fraktionsvorsitzender:</h4><p><a href="/abgeordnete/biografien/B/brinkhaus_ralph-857212" target="_self">Ral
         </article>
```

```
In [11]: html.find(class_='bt-standard-content').findAll('a')
```

```
Out[11]: [<a href="/abgeordnete/biografien/B/brinkhaus_ralph-857212" target="_self">Ralph Brinkhaus</a>,
          <a href="/abgeordnete/biografien/D/dobrindt_alexander-857264" target="_self">Alexander Dobrindt</a>,
          <a href="/abgeordnete/biografien/B/baer_dorothee-857110" target="_self">Dorothee Bär</a>,
          <a href="/abgeordnete/biografien/B/bilger_steffen-857146" target="_self">Steffen Bilger</a>,
          <a href="/abgeordnete/biografien/G/groehe_hermann-857336" target="_self">Hermann Gröhe</a>,
          <a href="/abgeordnete/biografien/L/lange_ulrich-863316" target="_self">Ulrich Lange</a>,
          <a href="/abgeordnete/biografien/L/lindholz_andrea-857700" target="_self">Andrea Lindholz</a>,
          <a href="/abgeordnete/biografien/L/lips_patricia-857666" target="_self">Patricia Lips</a>,
          <a href="/abgeordnete/biografien/M/middelberg_mathias-857718" target="_self">Dr. Matthias Middelberg</a>,
          <a href="/abgeordnete/biografien/M/mueller_sepp-857766" target="_self">Sepp Müller</a>,
          <a href="/abgeordnete/biografien/S/schoen_nadine-858006" target="_self">Nadine Schön</a>,
          <a href="/abgeordnete/biografien/S/spahn_jens-858012" target="_self">Jens Spahn</a>,
          <a href="/abgeordnete/biografien/W/wadephul_johann_david-858166" target="_self">Dr. Johann David Wadephul</a>,
          <a href="/abgeordnete/biografien/F/frei_thorsten-857308" target="_self">Thorsten Frei</a>,
          <a href="/abgeordnete/biografien/M/mueller_stefan-857778" target="_self">Stefan Müller</a>,
          <a href="/abgeordnete/biografien/H/hoppenstedt_hendrik-857418" target="_self">Dr. Hendrik Hoppenstedt</a>,
          <a href="/abgeordnete/biografien/S/schnieder_patrick-857992" target="_self">Patrick Schnieder</a>,
          <a href="/abgeordnete/biografien/W/warken_nina-858188" target="_self">Nina Warken</a>,
          <a href="/abgeordnete/biografien/F/frieser_michael-857294" target="_self">Michael Frieser</a>,
          <a href="/abgeordnete/biografien/H/heveling_ansgar-857460" target="_self">Ansgar Heveling</a>]
```

```
In [12]: # but we are also interested in the function of the members:
         html.find(class_='bt-standard-content').findAll('h4')
```

```
Out[12]: [<h4>Fraktionsvorsitzender:</h4>,
          <h4>Erster Stellvertretender Fraktionsvorsitzender:</h4>,
          <h4>Stellvertretende Fraktionsvorsitzende:</h4>,
          <h4>Erster Parlamentarischer Geschäftsführer:</h4>,
          <h4>Stellvertreter des Ersten Parlamentarischen Geschäftsführers:</h4>,
          <h4>Parlamentarische Geschäftsführer:</h4>,
          <h4>Justiziare:</h4>]
```

```
In [13]: from urllib.parse import urljoin

         for role_node in html.find(class_='bt-standard-content').findAll('h4'):

             role = role_node.text.rstrip(':')

             for link_node in role_node.next_sibling.findAll('a'):
                 name = link_node.text
                 link = urljoin(request_url, link_node.get('href'))
                 print(role, name, link)
```

```
Fraktionsvorsitzender Ralph Brinkhaus https://www.bundestag.de/abgeordnete/biografien/B/brinkhaus_ralph-857212
Erster Stellvertretender Fraktionsvorsitzender Alexander Dobrindt https://www.bundestag.de/abgeordnete/biografien/D/dobr
Stellvertretende Fraktionsvorsitzende Dorothee Bär https://www.bundestag.de/abgeordnete/biografien/B/baer_dorothee-85711
Stellvertretende Fraktionsvorsitzende Steffen Bilger https://www.bundestag.de/abgeordnete/biografien/B/bilger_steffen-85
Stellvertretende Fraktionsvorsitzende Hermann Gröhe https://www.bundestag.de/abgeordnete/biografien/G/groehe_hermann-857
Stellvertretende Fraktionsvorsitzende Ulrich Lange https://www.bundestag.de/abgeordnete/biografien/L/lange_ulrich-863316
Stellvertretende Fraktionsvorsitzende Andrea Lindholz https://www.bundestag.de/abgeordnete/biografien/L/lindholz_andrea-
Stellvertretende Fraktionsvorsitzende Patricia Lips https://www.bundestag.de/abgeordnete/biografien/L/lips_patricia-8576
Stellvertretende Fraktionsvorsitzende Dr. Matthias Middelberg https://www.bundestag.de/abgeordnete/biografien/M/middelbe
Stellvertretende Fraktionsvorsitzende Sepp Müller https://www.bundestag.de/abgeordnete/biografien/M/mueller_sepp-857766
Stellvertretende Fraktionsvorsitzende Nadine Schön https://www.bundestag.de/abgeordnete/biografien/S/schoen_nadine-85800
Stellvertretende Fraktionsvorsitzende Jens Spahn https://www.bundestag.de/abgeordnete/biografien/S/spahn_jens-858012
Stellvertretende Fraktionsvorsitzende Dr. Johann David Wadephul https://www.bundestag.de/abgeordnete/biografien/W/wadeph
Erster Parlamentarischer Geschäftsführer Thorsten Frei https://www.bundestag.de/abgeordnete/biografien/F/frei_thorsten-8
Stellvertreter des Ersten Parlamentarischen Geschäftsführers Stefan Müller https://www.bundestag.de/abgeordnete/biografi
Parlamentarische Geschäftsführer Dr. Hendrik Hoppenstedt https://www.bundestag.de/abgeordnete/biografien/H/hoppenstedt_h
Parlamentarische Geschäftsführer Patrick Schnieder https://www.bundestag.de/abgeordnete/biografien/S/schnieder_patrick-8
Parlamentarische Geschäftsführer Nina Warken https://www.bundestag.de/abgeordnete/biografien/W/warken_nina-858188
Justiziare Michael Frieser https://www.bundestag.de/abgeordnete/biografien/F/frieser_michael-857294
Justiziare Ansgar Heveling https://www.bundestag.de/abgeordnete/biografien/H/heveling_ansgar-857460
```

Now we put everything together, so that we can run this for all factions of the parliament: - we use a function to - fetch the page of the faction and - extract the members from the page content - iterate over all factions - store the list of faction roles and MPs in a data frame and CSV

In [14]:

```
%%script false --no-raise-error
# uncomment the above instruction to run this code
# note: do not run the cell by default
# because sending 6 HTTP requests may take long

import requests
from time import sleep
from urllib.parse import urljoin
from bs4 import BeautifulSoup
import pandas as pd

request_base_url = 'https://www.bundestag.de/parlament/fraktionen/'
factions = 'cducsu spd fdp linke gruene afd'.split()

def get_members_of_faction(faction):
    global request_base_url

    url = request_base_url + faction

    response = requests.get(url)
    if not response.ok:
        return

    result = []

    html = BeautifulSoup(response.text)

    for role_node in html.find(class_='bt-standard-content').findAll('h4'):

        role = role_node.text.strip().rstrip(':')

        if not role_node.next_sibling:
            continue

        for link_node in role_node.next_sibling.findAll('a'):
            name = link_node.text
            link = urljoin(url, link_node.get('href'))
            result.append([name, faction, role, link])

    return result


faction_roles = []

for faction in factions:
    if faction_roles:
        # be polite and wait before the next request
        sleep(5)
    faction_roles += get_members_of_faction(faction)

df_faction_roles = pd.DataFrame(faction_roles, columns=['name', 'faction', 'role', 'link'])
```

```
         df_faction_roles.to_csv('data/bundestag/faction_roles.csv')
```

```
In [15]: import pandas as pd

         df_faction_roles = pd.read_csv('data/bundestag/faction_roles.csv')
         df_faction_roles.value_counts('faction')
```

```
Out[15]: faction
         cducsu    20
         spd       15
         linke     13
         afd       12
         gruene    11
         fdp       11
         dtype: int64
```

```
In [16]: # not all members of the parliament have a role in their faction
         # and are listed on the landing page of the faction
         df_faction_roles.shape
```

```
Out[16]: (82, 5)
```

```
In [17]: df_faction_roles[df_faction_roles['role'].str.startswith('Fraktionsvorsitzend')]
```

```
Out[17]:     Unnamed: 0                 name faction                   role  \
         0            0      Ralph Brinkhaus  cducsu  Fraktionsvorsitzender
         20          20       Rolf Mützenich     spd  Fraktionsvorsitzender
         35          35       Christian Dürr     fdp  Fraktionsvorsitzender
         46          46    Amira Mohamed Ali   linke   Fraktionsvorsitzende
         47          47  Dr. Dietmar Bartsch   linke   Fraktionsvorsitzende
         59          59      Katharina Dröge  gruene   Fraktionsvorsitzende
         60          60     Britta Haßelmann  gruene   Fraktionsvorsitzende
         70          70     Dr. Alice Weidel     afd   Fraktionsvorsitzende
         71          71       Tino Chrupalla     afd   Fraktionsvorsitzende

                                                          link
         0   https://www.bundestag.de/abgeordnete/biografie…
         20  https://www.bundestag.de/abgeordnete/biografie…
         35  https://www.bundestag.de/abgeordnete/biografie…
         46  https://www.bundestag.de/abgeordnete/biografie…
         47  https://www.bundestag.de/abgeordnete/biografie…
         59  https://www.bundestag.de/abgeordnete/biografie…
         60  https://www.bundestag.de/abgeordnete/biografie…
         70  https://www.bundestag.de/abgeordnete/biografie…
         71  https://www.bundestag.de/abgeordnete/biografie…
```

```
In [18]: # now let's try whether we can fetch the biography and other information of a single MP

         member_url = df_faction_roles.loc[df_faction_roles['name']=='Jens Spahn','link'].values[0]
         member_response = requests.get(member_url)
         member_html = BeautifulSoup(member_response.text)
```

```python
# let's try first using the CSS class "bundestag-standard-content"
for node in member_html.findAll(class_='bt-standard-content'):
    print(node.text)
```

```
Abgeordnetenbüro
Deutscher BundestagPlatz der Republik 111011 Berlin




                    Kontakt




Wahlkreisbüro
Wüllener Str. 1148683 Ahaus


Profile im Internet


                Homepage

                Facebook

                Twitter

                LinkedIn

                Instagram






Geboren am 16. Mai 1980 in Ahaus; römisch-katholisch.Nach dem Abitur Ausbildung zum Bankkaufmann, anschließend Angestell





Nordrhein-Westfalen


Wahlkreis 124: Steinfurt I - Borken I
```

```
  Stellvertretendes Mitglied



  Wirtschaftsausschuss

  Ausschuss für Klimaschutz und Energie







Mitglieder des Bundestages haben gemäß § 45 Absatz 5 Abgeordnetengesetz innerhalb einer Frist von drei Monaten nach Erwe
Die veröffentlichungspflichtigen Angaben der Abgeordneten der 20. Wahlperiode werden grundsätzlich nach Ablauf dieser Fr
Die veröffentlichungspflichtigen Angaben der Abgeordneten der vergangenen Wahlperioden finden Sie im Archiv.
```

```
In [19]: member_html.text
```

```
Out[19]: '\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nDeutscher Bundestag - Jens Spahn\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nDire
```

### 4.2.1 Automatic Cleansing of Text

A trivial extraction of all text in the body of web page would include a lot of unwanted content (navigation menus, header, footer, side bars), the "main" content could be even only a small part in the middle of the page. There are heuristics and algorithms for automatic removal of "boilerplate" content:

- Mozilla Readability: the reader view of the Firefox browser
  - originally implemented in JavaScript, see Readability.js
  - but there is a Python port - ReadabiliPy or ReadabiliPy on pypi
- jusText or jusText on pypi
- trafilatura - the documentation and the academic paper include a list of other boilerplate-removal Python packages

First, an example usage of ReadabiliPy with the latest fetched page (without any manual selection of elements by CSS class):

```
In [20]: !pip install readabilipy

         from readabilipy import simple_json_from_html_string
```

```
        article = simple_json_from_html_string(member_response.text, use_readability=True)
```

```
Requirement already satisfied: readabilipy in ./.venv/lib/python3.9/site-packages (0.2.0)
Requirement already satisfied: regex in ./.venv/lib/python3.9/site-packages (from readabilipy) (2022.1.18)
Requirement already satisfied: html5lib in ./.venv/lib/python3.9/site-packages (from readabilipy) (1.1)
Requirement already satisfied: beautifulsoup4>=4.7.1 in ./.venv/lib/python3.9/site-packages (from readabilipy) (4.10.0)
Requirement already satisfied: lxml in ./.venv/lib/python3.9/site-packages (from readabilipy) (4.7.1)
Requirement already satisfied: soupsieve>1.2 in ./.venv/lib/python3.9/site-packages (from beautifulsoup4>=4.7.1->readabi
Requirement already satisfied: webencodings in ./.venv/lib/python3.9/site-packages (from html5lib->readabilipy) (0.5.1)
Requirement already satisfied: six>=1.9 in ./.venv/lib/python3.9/site-packages (from html5lib->readabilipy) (1.16.0)
```

```
In [21]: for paragraph in article['plain_text']:
             print(paragraph['text'])
             print()
```

```
Jens Spahn

© Jens Spahn/ Anne Hufnagl

Jens Spahn, CDU/CSU

Bankkaufmann

Abgeordnetenbüro

Deutscher BundestagPlatz der Republik 111011 Berlin

Kontakt

Wahlkreisbüro

Wüllener Str. 1148683 Ahaus

Geboren am 16. Mai 1980 in Ahaus; römisch-katholisch.Nach dem Abitur Ausbildung zum Bankkaufmann, anschließend Angestell

[Anmerkung der Redaktion: Die biografischen Angaben beruhen auf den Selbstauskünften der Abgeordneten.]

9766+OR+6761+OR+8521+OR+2893+OR+9597+OR+12752

Veröffentlichung

Abstimmungsthema

Abstimmungsverhalten

Mitglieder des Bundestages haben gemäß § 45 Absatz 5 Abgeordnetengesetz innerhalb einer Frist von drei Monaten nach Erwe

Die veröffentlichungspflichtigen Angaben der Abgeordneten der 20. Wahlperiode werden grundsätzlich nach Ablauf dieser Fr

Die veröffentlichungspflichtigen Angaben der Abgeordneten der vergangenen Wahlperioden finden Sie im Archiv.
```

```
In [22]: # but there's also a "readable" and simple HTML snippet
         # (shown as rendered HTML in the output)
         from IPython.core.display import HTML

         HTML(article['plain_content'])

Out[22]: <IPython.core.display.HTML object>
```

And now another package to strip boilerplate content, jusText…

As expected ReadibiliPy and jusText - differ in their API - have slightly different results for the given input - use different approaches under the hood

```
In [23]: !pip install jusText

Requirement already satisfied: jusText in ./.venv/lib/python3.9/site-packages (3.0.0)
Requirement already satisfied: lxml>=4.4.2 in ./.venv/lib/python3.9/site-packages (from jusText) (4.7.1)
```

```
In [24]: import justext

         paragraphs = justext.justext(member_response.text, justext.get_stoplist("German"))

         true_text = [p.text for p in paragraphs if not p.is_boilerplate]
         boilerplate = [p.text for p in paragraphs if p.is_boilerplate]

         true_text

Out[24]: ['Wahlkreisbüro',
          'Profile im Internet',
          'Biografie',
          'Geboren am 16. Mai 1980 in Ahaus; römisch-katholisch. Nach dem Abitur Ausbildung zum Bankkaufmann, anschließe
          '[Anmerkung der Redaktion: Die biografischen Angaben beruhen auf den Selbstauskünften der Abgeordneten.]',
          'Stellvertretendes Mitglied',
          'Veröffentlichungspflichtige Angaben',
          'Mitglieder des Bundestages haben gemäß § 45 Absatz 5 Abgeordnetengesetz innerhalb einer Frist von drei Monate
          'Die veröffentlichungspflichtigen Angaben der Abgeordneten der 20. Wahlperiode werden grundsätzlich nach Ablau
          'Die veröffentlichungspflichtigen Angaben der Abgeordneten der vergangenen Wahlperioden finden Sie im Archiv.'
```

```
In [25]: # what has been skipped?
         boilerplate

Out[25]: ['Direkt zum Hauptinhalt springenDirekt zum Hauptmenü springen',
          'ArchivArchiv',
          'Gebärdensprache',
          'Leichte Sprache',
          'Sprachen/LanguagesDE',
          'Arabisch\n□□□□□□□',
          'Bulgarisch\nбългарски',
          'Chinesisch\n□□',
          'Dänisch\ndansk',
          'Deutsch\nDeutsch',
          'Englisch\nEnglish',
          'Französisch\nfrançais',
          'Griechisch\nΕλληνικά',
          'Italienisch\nitaliano',
          'Kroatisch\nhrvatski',
          'Niederländisch\nNederlands',
          'Polnisch\npolski',
          'Portugiesisch\nportuguês',
          'Rumänisch\nromână',
          'Russisch\nрусский',
          'Serbisch\nсрпски',
          'Spanisch\nespañol',
          'Tschechisch\nčeština',
          'Türkisch\nTürkçe',
          'schließen',
          'Deutscher Bundestag',
          'Suche',
          'Startseite',
          'Abgeordnete',
          'Parlament',
          'Ausschüsse',
          'Internationales',
          'Dokumente',
          'Mediathek',
          'Presse',
          'Besuch',
          'Service',
          'Startseite',
          'AbgeordneteAbgeordnete: Untermenü anzeigen',
          'StartseiteStartseite',
          'AbgeordneteAbgeordnete: Untermenü ausblenden',
          'Biografien',
          'StartseiteStartseite',
          'AbgeordneteAbgeordnete: Untermenü anzeigen',
          'BiografienBiografien: Untermenü anzeigen',
          'Nebentätigkeiten',
          'Entschädigung',
          'Wahlergebnisse',
          'ParlamentParlament: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü ausblenden',
          'Bundestagswahl 2021',
          'Grundgesetz',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'GrundgesetzGrundgesetz: Untermenü anzeigen',
          'AufgabenAufgaben: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'AufgabenAufgaben: Untermenü anzeigen',
          'Gesetzgebung',
          'Kontrolle der Regierung',
          'Der Bundeshaushalt',
          'Wahl des Kanzlers/der Kanzlerin',
          'Wahl des Bundespräsidenten',
          'Rechtliche Grundlagen',
          'PlenumPlenum: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'PlenumPlenum: Untermenü anzeigen',
          'Tagesordnungen',
          'Namentliche Abstimmungen',
          'Sitzverteilung des 20. Deutschen Bundestages',
          'Sitzungskalender',
          'Schriftführer',
          'PräsidiumPräsidium: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'PräsidiumPräsidium: Untermenü anzeigen',
          'Funktion und Aufgabe',
          'Wahl des Präsidiums',
          'Reden und Beiträge der Präsidenten',
          'Parteienfinanzierung',
          'Ältestenrat',
          'FraktionenFraktionen: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'FraktionenFraktionen: Untermenü anzeigen',
          'SPD',
          'CDU/CSU',
          'Bündnis 90/ Die Grünen',
          'FDP',
          'AfD',
          'Die Linke',
          'Petitionen',
          'SED-Opferbeauftragte',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'SED-OpferbeauftragteSED-Opferbeauftragte: Untermenü anzeigen',
          'Wehrbeauftragte',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'WehrbeauftragteWehrbeauftragte: Untermenü anzeigen',
          'Verwaltung',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'VerwaltungVerwaltung: Untermenü anzeigen',
          'GeschichteGeschichte: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'GeschichteGeschichte: Untermenü anzeigen',
          'Historische Ausstellungen',
          'Deutscher Parlamentarismus',
          'Parlamentarische Schauplätze',
          'Bundestagspräsidenten seit 1949',
          'Herbst 1918: Vom Kaiserreich zur Republik',
          'Staatliche Symbole',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'Staatliche SymboleStaatliche Symbole: Untermenü anzeigen',
          'ParlamentspreiseParlamentspreise: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'ParlamentspreiseParlamentspreise: Untermenü anzeigen',
          'Medienpreis',
          'Deutsch-Französischer Parlamentspreis',
          'Wissenschaftspreis',
          'WahlenWahlen: Untermenü anzeigen',
          'StartseiteStartseite',
          'ParlamentParlament: Untermenü anzeigen',
          'WahlenWahlen: Untermenü anzeigen',
          'Wahlergebnisse',
          'Wahltermine in Deutschland',
          'Lobbyregister',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü ausblenden',
          'Arbeit und Soziales',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Arbeit und SozialesArbeit und Soziales: Untermenü anzeigen',
          'Auswärtiges',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'AuswärtigesAuswärtiges: Untermenü anzeigen',
          'Bildung, Forschung und Technikfolgenabschätzung',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Bildung, Forschung und TechnikfolgenabschätzungBildung, Forschung und Technikfolgenabschätzung: Untermenü anz
          'Digitales',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'DigitalesDigitales: Untermenü anzeigen',
          'Ernährung und Landwirtschaft',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Ernährung und LandwirtschaftErnährung und Landwirtschaft: Untermenü anzeigen',
          'Europäische Union',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Europäische UnionEuropäische Union: Untermenü anzeigen',
          'Familie, Senioren, Frauen und Jugend',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Familie, Senioren, Frauen und JugendFamilie, Senioren, Frauen und Jugend: Untermenü anzeigen',
          'Finanzen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'FinanzenFinanzen: Untermenü anzeigen',
          'Gesundheit',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'GesundheitGesundheit: Untermenü anzeigen',
          'HaushaltHaushalt: Untermenü anzeigen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'HaushaltHaushalt: Untermenü anzeigen',
          'Rechnungsprüfungsausschuss',
          'Unterausschuss zu Fragen der Europäischen Union',
          'Inneres und Heimat',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Inneres und HeimatInneres und Heimat: Untermenü anzeigen',
          'Klimaschutz und Energie',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Klimaschutz und EnergieKlimaschutz und Energie: Untermenü anzeigen',
          'Kultur und Medien',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Kultur und MedienKultur und Medien: Untermenü anzeigen',
          'Menschenrechte und humanitäre HilfeMenschenrechte und humanitäre Hilfe: Untermenü anzeigen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Menschenrechte und humanitäre HilfeMenschenrechte und humanitäre Hilfe: Untermenü anzeigen',
          'Programm „Parlamentarier schützen Parlamentarier“',
          'Petitionsausschuss',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'PetitionsausschussPetitionsausschuss: Untermenü anzeigen',
          'Recht',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'RechtRecht: Untermenü anzeigen',
          'Sport',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'SportSport: Untermenü anzeigen',
          'Tourismus',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'TourismusTourismus: Untermenü anzeigen',
          'Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Umwelt, Naturschutz, nukleare Sicherheit und VerbraucherschutzUmwelt, Naturschutz, nukleare Sicherheit und Ve
          'Verkehr',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'VerkehrVerkehr: Untermenü anzeigen',
          'Vermittlungsausschuss',
          'Verteidigung',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'VerteidigungVerteidigung: Untermenü anzeigen',
          'Wahlprüfung',
          'Wahlprüfung, Immunität und Geschäftsordnung',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Wahlprüfung, Immunität und GeschäftsordnungWahlprüfung, Immunität und Geschäftsordnung: Untermenü anzeigen',
          'Wirtschaft',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'WirtschaftWirtschaft: Untermenü anzeigen',
          'wirtschaftliche Zusammenarbeit und Entwicklung',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'wirtschaftliche Zusammenarbeit und Entwicklungwirtschaftliche Zusammenarbeit und Entwicklung: Untermenü anzei
          'Wohnen, Stadtentwicklung, Bauwesen und Kommunen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'Wohnen, Stadtentwicklung, Bauwesen und KommunenWohnen, Stadtentwicklung, Bauwesen und Kommunen: Untermenü anz
          'Hauptausschuss',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'HauptausschussHauptausschuss: Untermenü anzeigen',
          'weitere Gremienweitere Gremien: Untermenü anzeigen',
          'StartseiteStartseite',
          'AusschüsseAusschüsse: Untermenü anzeigen',
          'weitere Gremienweitere Gremien: Untermenü anzeigen',
          'Bundesfinanzierungsgremium',
          'Deutsch-Französische Parlamentarische Versammlung',
          'Parlamentarisches Kontrollgremium',
          'G 10-Kommission',
          'InternationalesInternationales: Untermenü anzeigen',
          'StartseiteStartseite',
          'InternationalesInternationales: Untermenü ausblenden',
          'Europapolitik im BundestagEuropapolitik im Bundestag: Untermenü anzeigen',
          'StartseiteStartseite',
          'InternationalesInternationales: Untermenü anzeigen',
          'Europapolitik im BundestagEuropapolitik im Bundestag: Untermenü anzeigen',
          'Mitwirkungsrechte des Deutschen Bundestages',
          'Europa in den Ausschüssen',
          'Verbindungsbüro Brüssel',
          'Zusammenarbeit der Parlamente in Europa',
          'Internationale parlamentarische VersammlungenInternationale parlamentarische Versammlungen: Untermenü anzeige
          'StartseiteStartseite',
          'InternationalesInternationales: Untermenü anzeigen',
          'Internationale parlamentarische VersammlungenInternationale parlamentarische Versammlungen: Untermenü anzeige
          'Parlamentarische Versammlung der OSZE',
          'Parlamentarische Versammlung der NATO',
          'Parlamentarische Versammlung des Europarates',
          'Interparlamentarische Union',
          'Stabilität, wirtschaftspolitische Koordinierung und Steuerung in der EU',
          'Gemeinsame Außen-, Sicherheits- und Verteidigungspolitik',
          'Konferenzen der Präsidentinnen und Präsidenten der Parlamente',
          'Parlamentarische Versammlung der Union für den Mittelmeerraum',
          'Ostseeparlamentarierkonferenz',
          'Parlamentarische Versammlung der Schwarzmeerwirtschaftskooperation',
          'Parlamentariergruppen',
          'StartseiteStartseite',
          'InternationalesInternationales: Untermenü anzeigen',
          'ParlamentariergruppenParlamentariergruppen: Untermenü anzeigen',
          'Internationales Parlaments-Stipendium (IPS)',
          'Parlamentarisches Patenschafts-Programm (PPP)',
          'StartseiteStartseite',
          'InternationalesInternationales: Untermenü anzeigen',
          'Parlamentarisches Patenschafts-Programm (PPP)Parlamentarisches Patenschafts-Programm (PPP): Untermenü anzeige
          'Wahltermine EU',
          'Parlamentarische Dimension der EU-Ratspräsidentschaft',
          'DokumenteDokumente: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü ausblenden',
          'Drucksachen',
          'Dokumentations- und Informationssystem (DIP)',
          'ParlamentsdokumentationParlamentsdokumentation: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'ParlamentsdokumentationParlamentsdokumentation: Untermenü anzeigen',
          'Corona-Dossier',
          'ProtokolleProtokolle: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'ProtokolleProtokolle: Untermenü anzeigen',
          'Tagesaktuelles Plenarprotokoll',
          'Endgültige Plenarprotokolle',
          'Amtliche Protokolle',
          'Gutachten und Ausarbeitungen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'Gutachten und AusarbeitungenGutachten und Ausarbeitungen: Untermenü anzeigen',
          'ParlamentsarchivParlamentsarchiv: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'ParlamentsarchivParlamentsarchiv: Untermenü anzeigen',
          'Datenhandbuch',
          'Registrierte Verbände (Öffentliche Liste)',
          'BibliothekBibliothek: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'BibliothekBibliothek: Untermenü anzeigen',
          'Bibliothekskatalog',
          'Pressedokumentation',
          'Webarchiv',
          'TextarchivTextarchiv: Untermenü anzeigen',
          'StartseiteStartseite',
          'DokumenteDokumente: Untermenü anzeigen',
          'TextarchivTextarchiv: Untermenü anzeigen',
          '2022',
          '2021',
          '2020',
          '2019',
          '2018',
          '2017',
          'MediathekMediathek: Untermenü anzeigen',
          'StartseiteStartseite',
          'MediathekMediathek: Untermenü ausblenden',
          'Das Parlamentsfernsehen',
          'Gebärdensprache',
          'Empfang',
          'Audio-Übertragungen',
          'PressePresse: Untermenü anzeigen',
          'StartseiteStartseite',
          'PressePresse: Untermenü ausblenden',
          'PressemitteilungenPressemitteilungen: Untermenü anzeigen',
          'StartseiteStartseite',
          'PressePresse: Untermenü anzeigen',
          'PressemitteilungenPressemitteilungen: Untermenü anzeigen',
          '2022',
          '2021',
          'Kurzmeldungen (hib)',
          'Akkreditierung',
          'Bilddatenbank',
          'Mitschnittservice',
          'Kontakt',
          'BesuchBesuch: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü ausblenden',
          'Kuppel',
          'Barrierefreier Zugang',
          'Plenarsitzung',
          'FührungenFührungen: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'FührungenFührungen: Untermenü anzeigen',
          'Plenarsitzung',
          'Einladung durch Abgeordnete',
          'Angebote für Kinder und Jugendliche',
          'AusstellungenAusstellungen: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'AusstellungenAusstellungen: Untermenü anzeigen',
          'Parlamentshistorische Ausstellung im Deutschen Dom',
          'Politisch-parlamentarische Ausstellungen',
          'Kunstausstellungen',
          'Bundestag in Ihrer Nähe',
          'Online-Anmeldung',
          'KunstKunst: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'KunstKunst: Untermenü anzeigen',
          'Kunst am Bau',
          'Artothek - die Kunstsammlung',
          'Workshops',
          'Kunstbeirat',
          'Aufträge an zeitgenössische Künstler',
          'Mauer-Mahnmal',
          'Gedenktafeln',
          'Ausgewählt - Aus der Kunstsammlung des Deutschen Bundestages',
          'ArchitekturArchitektur: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'ArchitekturArchitektur: Untermenü anzeigen',
          'Reichstagsgebäude',
          'Jakob-Kaiser-Haus',
          'Paul-Löbe-Haus',
          'Marie-Elisabeth-Lüders-Haus',
          'Weitere Bundestagsgebäude',
          'Energiekonzept',
          'SeminareSeminare: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'SeminareSeminare: Untermenü anzeigen',
          'Das Parlamentsseminar des Deutschen Bundestages',
          'Seminare für Lehrerinnen und Lehrer',
          'Seminare für Journalistenschülerinnen und -schüler',
          'Bundestag in Ihrer NäheBundestag in Ihrer Nähe: Untermenü anzeigen',
          'StartseiteStartseite',
          'BesuchBesuch: Untermenü anzeigen',
          'Bundestag in Ihrer NäheBundestag in Ihrer Nähe: Untermenü anzeigen',
          'Infomobil',
          'Wanderausstellung',
          'Messestand',
          'ServiceService: Untermenü anzeigen',
          'StartseiteStartseite',
          'ServiceService: Untermenü ausblenden',
          'Parlamentsbegriffe A – Z',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'Parlamentsbegriffe A - ZParlamentsbegriffe A - Z: Untermenü anzeigen',
          'Häufig gestellte Fragen',
          'Virtueller Berater',
          'Informationsmaterial',
          'Bundestagsshop',
          'Newsletter',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'NewsletterNewsletter: Untermenü anzeigen',
          'Karriere',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'KarriereKarriere: Untermenü anzeigen',
          'Das Quiz zum Deutschen Bundestag',
          'Formulare und Anträge',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'Formulare und AnträgeFormulare und Anträge: Untermenü anzeigen',
          'Open Data',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'Open DataOpen Data: Untermenü anzeigen',
          'Kontakt',
          'StartseiteStartseite',
          'ServiceService: Untermenü anzeigen',
          'KontaktKontakt: Untermenü anzeigen',
          'Abgeordnete',
          'schließen',
          'Biografien',
          'Nebentätigkeiten',
          'Entschädigung',
          'Wahlergebnisse',
          'Parlament',
          'schließen',
          'Grundgesetz',
          'Aufgaben',
          'Gesetzgebung',
          'Kontrolle der Regierung',
          'Der Bundeshaushalt',
          'Wahl des Kanzlers/der Kanzlerin',
          'Wahl des Bundespräsidenten',
          'Rechtliche Grundlagen',
          'Plenum',
          'Tagesordnungen',
          'Namentliche Abstimmungen',
          'Sitzverteilung des 20. Deutschen Bundestages',
          'Sitzungskalender',
          'Schriftführer',
          'Präsidium',
          'Funktion und Aufgabe',
          'Wahl des Präsidiums',
          'Reden und Beiträge der Präsidenten',
          'Parteienfinanzierung',
          'Ältestenrat',
          'Fraktionen',
          'SPD',
          'CDU/CSU',
          'Bündnis 90/ Die Grünen',
          'FDP',
          'AfD',
          'Die Linke',
          'Petitionen',
          'SED-Opferbeauftragte',
          'Wehrbeauftragte',
          'Verwaltung',
          'Geschichte',
          'Historische Ausstellungen',
          'Deutscher Parlamentarismus',
          'Parlamentarische Schauplätze',
          'Bundestagspräsidenten seit 1949',
          'Herbst 1918: Vom Kaiserreich zur Republik',
          'Staatliche Symbole',
          'Parlamentspreise',
          'Medienpreis',
          'Deutsch-Französischer Parlamentspreis',
          'Wissenschaftspreis',
          'Wahlen',
          'Wahlergebnisse',
          'Wahltermine in Deutschland',
          'Lobbyregister',
          'Ausschüsse',
          'schließen',
          'Arbeit und Soziales',
          'Auswärtiges',
          'Bildung, Forschung und Technikfolgenabschätzung',
          'Digitales',
          'Ernährung und Landwirtschaft',
          'Europäische Union',
          'Familie, Senioren, Frauen und Jugend',
          'Finanzen',
          'Gesundheit',
          'Haushalt',
          'Rechnungsprüfungsausschuss',
          'Unterausschuss zu Fragen der Europäischen Union',
          'Inneres und Heimat',
          'Klimaschutz und Energie',
          'Kultur und Medien',
          'Menschenrechte und humanitäre Hilfe',
          'Petitionsausschuss',
          'Recht',
          'Sport',
          'Tourismus',
          'Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz',
          'Verkehr',
          'Vermittlungsausschuss',
          'Verteidigung',
          'Wahlprüfung',
          'Wahlprüfung, Immunität und Geschäftsordnung',
          'Wirtschaft',
          'wirtschaftliche Zusammenarbeit und Entwicklung',
          'Wohnen, Stadtentwicklung, Bauwesen und Kommunen',
          'Hauptausschuss',
          'weitere Gremien',
          'Bundesfinanzierungsgremium',
          'Deutsch-Französische Parlamentarische Versammlung',
          'Parlamentarisches Kontrollgremium',
          'G 10-Kommission',
          'Internationales',
          'schließen',
          'Europapolitik im Bundestag',
          'Mitwirkungsrechte des Deutschen Bundestages',
          'Europa in den Ausschüssen',
          'Verbindungsbüro Brüssel',
          'Zusammenarbeit der Parlamente in Europa',
          'Internationale parlamentarische Versammlungen',
          'Parlamentarische Versammlung der OSZE',
          'Parlamentarische Versammlung der NATO',
          'Parlamentarische Versammlung des Europarates',
          'Interparlamentarische Union',
          'Stabilität, wirtschaftspolitische Koordinierung und Steuerung in der EU',
          'Gemeinsame Außen-, Sicherheits- und Verteidigungspolitik',
          'Konferenzen der Präsidentinnen und Präsidenten der Parlamente',
          'Parlamentarische Versammlung der Union für den Mittelmeerraum',
          'Ostseeparlamentarierkonferenz',
          'Parlamentarische Versammlung der Schwarzmeerwirtschaftskooperation',
          'Parlamentariergruppen',
          'Internationales Parlaments-Stipendium (IPS)',
          'Parlamentarisches Patenschafts-Programm (PPP)',
          'Wahltermine EU',
          'Parlamentarische Dimension der EU-Ratspräsidentschaft',
          'Dokumente',
          'schließen',
          'Drucksachen',
          'Dokumentations- und Informationssystem (DIP)',
          'Parlamentsdokumentation',
          'Corona-Dossier',
          'Protokolle',
          'Tagesaktuelles Plenarprotokoll',
          'Endgültige Plenarprotokolle',
          'Amtliche Protokolle',
          'Gutachten und Ausarbeitungen',
          'Parlamentsarchiv',
          'Datenhandbuch',
          'Registrierte Verbände (Öffentliche Liste)',
          'Bibliothek',
          'Bibliothekskatalog',
          'Pressedokumentation',
          'Webarchiv',
          'Textarchiv',
          '2022',
          '2021',
          '2020',
          '2019',
          '2018',
          '2017',
          'Mediathek',
          'schließen',
          'Das Parlamentsfernsehen',
          'Gebärdensprache',
          'Empfang',
          'Audio-Übertragungen',
          'Presse',
          'schließen',
          'Pressemitteilungen',
          '2022',
          '2021',
          'Kurzmeldungen (hib)',
          'Akkreditierung',
          'Bilddatenbank',
          'Mitschnittservice',
          'Kontakt',
          'Besuch',
          'schließen',
          'Kuppel',
          'Barrierefreier Zugang',
          'Plenarsitzung',
          'Führungen',
          'Plenarsitzung',
          'Einladung durch Abgeordnete',
          'Angebote für Kinder und Jugendliche',
          'Ausstellungen',
          'Parlamentshistorische Ausstellung im Deutschen Dom',
          'Politisch-parlamentarische Ausstellungen',
          'Kunstausstellungen',
          'Bundestag in Ihrer Nähe',
          'Online-Anmeldung',
          'Kunst',
          'Kunst am Bau',
          'Artothek - die Kunstsammlung',
          'Workshops',
          'Kunstbeirat',
          'Aufträge an zeitgenössische Künstler',
          'Mauer-Mahnmal',
          'Gedenktafeln',
          'Ausgewählt - Aus der Kunstsammlung des Deutschen Bundestages',
          'Architektur',
          'Reichstagsgebäude',
          'Jakob-Kaiser-Haus',
          'Paul-Löbe-Haus',
          'Marie-Elisabeth-Lüders-Haus',
          'Weitere Bundestagsgebäude',
          'Energiekonzept',
          'Seminare',
          'Das Parlamentsseminar des Deutschen Bundestages',
          'Seminare für Lehrerinnen und Lehrer',
          'Seminare für Journalistenschülerinnen und -schüler',
          'Bundestag in Ihrer Nähe',
          'Infomobil',
          'Wanderausstellung',
          'Messestand',
          'Service',
          'schließen',
          'Parlamentsbegriffe A – Z',
          'Häufig gestellte Fragen',
          'Virtueller Berater',
          'Informationsmaterial',
          'Bundestagsshop',
          'Newsletter',
          'Karriere',
          'Das Quiz zum Deutschen Bundestag',
          'Formulare und Anträge',
          'Open Data',
          'Kontakt',
          'Abgeordnete',
          'Startseite',
          'Abgeordnete',
          'Biografien',
          'S',
          'Jens Spahn',
          '© Jens Spahn/ Anne Hufnagl',
          'Jens Spahn, CDU/CSU',
          'Bankkaufmann',
          'Kontakt',
          'Bereich "Kontakt" ein-/ausklappen',
          'Abgeordnetenbüro',
          'Deutscher Bundestag Platz der Republik 1 11011 Berlin',
          'Kontakt',
          'Wüllener Str. 11 48683 Ahaus',
          'Homepage',
          'Facebook',
          'Twitter',
          'LinkedIn',
          'Instagram',
          'Biografie',
          'Reden',
          'Abstimmungen',
          'Bereich "Biografie" ein-/ausklappen',
          'Reden',
          'Bereich "Reden" ein-/ausklappen',
          '9766+OR+6761+OR+8521+OR+2893+OR+9597+OR+12752',
          'Abstimmungen',
          'Bereich "Abstimmungen" ein-/ausklappen',
          'Veröffentlichung',
          'Abstimmungsthema',
          'Abstimmungsverhalten',
          'Direkt gewählt',
          'Bereich "Direkt gewählt" ein-/ausklappen',
          'Wahlkreis 124: Steinfurt I - Borken I',
          'Nordrhein-Westfalen',
          'Wahlkreis 124: Steinfurt I - Borken I',
          'Mitgliedschaften und Ämter im Bundestag',
          'Bereich "Mitgliedschaften und Ämter im Bundestag" ein-/ausklappen',
          'Wirtschaftsausschuss',
          'Ausschuss für Klimaschutz und Energie',
          'Bereich "Veröffentlichungspflichtige Angaben" ein-/ausklappen',
          'Startseite',
          'Abgeordnete',
          'Biografien',
          'S',
          'Hilfe',
          'Kontakt',
          'Inhaltsübersicht',
          '© Deutscher Bundestag',
          'Barrierefreiheit',
          'Datenschutz',
          'Impressum',
          'Ausdruck aus dem Internet-Angebot des Deutschen Bundestages',
          'https://www.bundestag.de/abgeordnete/biografien/S/spahn_jens-858012',
          'Stand: 30.01.2022']
```

## 4.3 Processing XML

The Open Data portal of the German parliament offers a zip file “Stammdaten aller Abgeordneten seit 1949 im XML-Format (Stand 04.11.2021)” for free download. Most likely we should get the information about all PMs from this source. But how do we process XML?

Assumed the zip archive has been downloaded, unzipped and the files are all placed in `data/bundestag/`, we can simply read the file and pass it to beautifulsoup which will parse it. But we request a specific parser feature (`lxml-xml`) so that the casing of XML elements is preserved.

```
In [26]: from bs4 import BeautifulSoup

         xml = BeautifulSoup(open('data/bundestag/MDB_STAMMDATEN.XML').read(),
                             features='lxml-xml')

         xml.MDB
```

```
Out[26]: <MDB>
         <ID>11000001</ID>
         <NAMEN>
         <NAME>
         <NACHNAME>Abelein</NACHNAME>
         <VORNAME>Manfred</VORNAME>
         <ORTSZUSATZ/>
         <ADEL/>
         <PRAEFIX/>
         <ANREDE_TITEL>Dr.</ANREDE_TITEL>
         <AKAD_TITEL>Prof. Dr.</AKAD_TITEL>
         <HISTORIE_VON>19.10.1965</HISTORIE_VON>
         <HISTORIE_BIS/>
         </NAME>
         </NAMEN>
         <BIOGRAFISCHE_ANGABEN>
         <GEBURTSDATUM>20.10.1930</GEBURTSDATUM>
         <GEBURTSORT>Stuttgart</GEBURTSORT>
         <GEBURTSLAND/>
         <STERBEDATUM>17.01.2008</STERBEDATUM>
         <GESCHLECHT>männlich</GESCHLECHT>
         <FAMILIENSTAND>keine Angaben</FAMILIENSTAND>
         <RELIGION>katholisch</RELIGION>
         <BERUF>Rechtsanwalt, Wirtschaftsprüfer, Universitätsprofessor</BERUF>
         <PARTEI_KURZ>CDU</PARTEI_KURZ>
         <VITA_KURZ/>
         <VEROEFFENTLICHUNGSPFLICHTIGES/>
         </BIOGRAFISCHE_ANGABEN>
         <WAHLPERIODEN>
         <WAHLPERIODE>
         <WP>5</WP>
         <MDBWP_VON>19.10.1965</MDBWP_VON>
         <MDBWP_BIS>19.10.1969</MDBWP_BIS>
         <WKR_NUMMER>174</WKR_NUMMER>
         <WKR_NAME/>
         <WKR_LAND>BWG</WKR_LAND>
         <LISTE/>
         <MANDATSART>Direktwahl</MANDATSART>
         <INSTITUTIONEN>
         <INSTITUTION>
         <INSART_LANG>Fraktion/Gruppe</INSART_LANG>
         <INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
         <MDBINS_VON/>
         <MDBINS_BIS/>
         <FKT_LANG/>
         <FKTINS_VON/>
         <FKTINS_BIS/>
         </INSTITUTION>
```

```
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>6</WP>
<MDBWP_VON>20.10.1969</MDBWP_VON>
<MDBWP_BIS>22.09.1972</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON/>
<MDBINS_BIS/>
<FKT_LANG/>
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>7</WP>
<MDBWP_VON>13.12.1972</MDBWP_VON>
<MDBWP_BIS>13.12.1976</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON/>
<MDBINS_BIS/>
<FKT_LANG/>
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>8</WP>
<MDBWP_VON>14.12.1976</MDBWP_VON>
<MDBWP_BIS>04.11.1980</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
```

```
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON/>
<MDBINS_BIS/>
<FKT_LANG/>
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>9</WP>
<MDBWP_VON>04.11.1980</MDBWP_VON>
<MDBWP_BIS>29.03.1983</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON/>
<MDBINS_BIS/>
<FKT_LANG/>
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>10</WP>
<MDBWP_VON>29.03.1983</MDBWP_VON>
<MDBWP_BIS>18.02.1987</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON/>
<MDBINS_BIS/>
<FKT_LANG/>
```

```
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
<WAHLPERIODE>
<WP>11</WP>
<MDBWP_VON>18.02.1987</MDBWP_VON>
<MDBWP_BIS>20.12.1990</MDBWP_BIS>
<WKR_NUMMER>174</WKR_NUMMER>
<WKR_NAME/>
<WKR_LAND>BWG</WKR_LAND>
<LISTE/>
<MANDATSART>Direktwahl</MANDATSART>
<INSTITUTIONEN>
<INSTITUTION>
<INSART_LANG>Fraktion/Gruppe</INSART_LANG>
<INS_LANG>Fraktion der Christlich Demokratischen Union/Christlich - Sozialen Union</INS_LANG>
<MDBINS_VON>18.02.1987</MDBINS_VON>
<MDBINS_BIS>20.12.1990</MDBINS_BIS>
<FKT_LANG/>
<FKTINS_VON/>
<FKTINS_BIS/>
</INSTITUTION>
</INSTITUTIONEN>
</WAHLPERIODE>
</WAHLPERIODEN>
</MDB>
```

```
In [27]: len(xml.findAll('MDB'))
```

```
Out[27]: 4365
```

For the 4365 members of the German parliament (active and previous members), let's now look for their academic titles...

```
In [28]: from collections import defaultdict, Counter

         mp_acad_title = Counter()

         mp_with_acad_title, mp_total = 0, 0

         election_periods = defaultdict(Counter)


         for mp in xml.findAll('MDB'):

             mp_total += 1

             has_academic_title = False
             for nn in mp.findAll("NAME"):
                 if nn.AKAD_TITEL.text:
                     has_academic_title = True
                     mp_acad_title[nn.AKAD_TITEL.text] += 1

             if has_academic_title:
                 # count a title only once (in case of multiple names)
                 mp_with_acad_title += 1

             for ep in mp.findAll('WAHLPERIODE'):
                 period = int(ep.WP.text)
                 election_periods[period]['mp_total'] += 1
                 if has_academic_title:
                     election_periods[period]['mp_with_academic_title'] += 1

         mp_with_acad_title / mp_total
```

```
Out[28]: 0.2510882016036655
```

```
In [29]: election_periods
```

```
Out[29]: defaultdict(collections.Counter,
                     {5: Counter({'mp_total': 559, 'mp_with_academic_title': 173}),
                      6: Counter({'mp_total': 556, 'mp_with_academic_title': 182}),
                      7: Counter({'mp_total': 549, 'mp_with_academic_title': 185}),
                      8: Counter({'mp_total': 553, 'mp_with_academic_title': 187}),
                      9: Counter({'mp_total': 549, 'mp_with_academic_title': 171}),
                      10: Counter({'mp_total': 576, 'mp_with_academic_title': 178}),
                      11: Counter({'mp_total': 702, 'mp_with_academic_title': 241}),
                      3: Counter({'mp_total': 562, 'mp_with_academic_title': 169}),
                      4: Counter({'mp_total': 580, 'mp_with_academic_title': 182}),
                      2: Counter({'mp_total': 558, 'mp_with_academic_title': 166}),
                      12: Counter({'mp_total': 699, 'mp_with_academic_title': 218}),
                      13: Counter({'mp_total': 693, 'mp_with_academic_title': 167}),
                      14: Counter({'mp_total': 699, 'mp_with_academic_title': 146}),
                      15: Counter({'mp_total': 628, 'mp_with_academic_title': 115}),
                      16: Counter({'mp_total': 642, 'mp_with_academic_title': 132}),
                      1: Counter({'mp_total': 474, 'mp_with_academic_title': 149}),
                      17: Counter({'mp_total': 652, 'mp_with_academic_title': 130}),
                      18: Counter({'mp_total': 658, 'mp_with_academic_title': 127}),
                      19: Counter({'mp_total': 750, 'mp_with_academic_title': 149}),
                      20: Counter({'mp_total': 736, 'mp_with_academic_title': 121})})
```

```
In [30]: df = pd.DataFrame.from_dict(election_periods, orient='index').sort_index()

         df['% with acad. title'] = 100.0 * df['mp_with_academic_title'] / df['mp_total']
         df
```

```
Out[30]:     mp_total  mp_with_academic_title  % with acad. title
         1        474                     149           31.434599
         2        558                     166           29.749104
         3        562                     169           30.071174
         4        580                     182           31.379310
         5        559                     173           30.948122
         6        556                     182           32.733813
```

```
7       549                      185            33.697632
8       553                      187            33.815552
9       549                      171            31.147541
10      576                      178            30.902778
11      702                      241            34.330484
12      699                      218            31.187411
13      693                      167            24.098124
14      699                      146            20.886981
15      628                      115            18.312102
16      642                      132            20.560748
17      652                      130            19.938650
18      658                      127            19.300912
19      750                      149            19.866667
20      736                      121            16.440217
```

```
In [31]: mp_acad_title.most_common()
```

```
Out[31]: [('Dr.', 965),
          ('Prof. Dr.', 85),
          ('Dr. h. c.', 42),
          ('Dr. Dr. h. c.', 17),
          ('Prof.', 13),
          ('Dr. - Ing.', 12),
          ('Prof. Dr. h. c.', 3),
          ('Dipl. - Ing.', 3),
          ('Dr. Dr.', 3),
          ('Prof. Dr. Dr. h. c.', 3),
          ('Dr. - Ing. e. h.', 2),
          ('Prof. Dr. Dr.', 2),
          ('Dr. - Ing. Dr. h. c.', 1),
          ('Prof. h. c.', 1),
          ('Prof. Dr. - Ing.', 1),
          ('Dr. h. c. Dr. - Ing. e. h.', 1),
          ('Dr. - Ing. Dr. - Ing. e. h. Dr. h. c.', 1),
          ('Dr. h. c. Dr. e. h.', 1),
          ('Prof. h. c. Dr.', 1),
          ('Dr. h. c. (Univ Kyiv)', 1),
          ('HonD', 1),
          ('Dr. h. c. (NUACA)', 1)]
```

A final note: Reading the XML file describing the members of the German parliament into a tabular data structure will be painful (similar as for JSON data source) because of - the nested structure - some list-like data, for example the fact that one MP can have multiple names

Instead of coding the conversion in Python: with XSLT there is a dedicated language for transforming XML documents into other document formats.

The Open Discourse projects hosts the proceedings of the German parliament and also a list of MPs in data formats easy to consume. See the Open Discourse data sets page.

## 4.4 Browser automation with Python

- Selenium
  - nice example: impf-botpy
- Playwright
  - Playwright on pypi including nice examples (some cited below)
  - Python API docs

Note: Playwright does not run in a Jupyter notebook. We'll run the scripts directly in the Python interpreter.

Installation:

```
pip install playwright
playwright install
```

Take a screenshot using two different browsers:

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    for browser_type in [p.chromium, p.firefox]:
        browser = browser_type.launch()
        page = browser.new_page()
        page.goto('http://whatsmyuseragent.org/')
        _ = page.screenshot(path=f'figures/example-{browser_type.name}.png')
        browser.close()
```

Just run the script `scripts/playwright_whatsmyuseragent_screenshot.py` in the console / shell:

```
python ./scripts/playwright_whatsmyuseragent_screenshot.py
```

The screenshots are then found in the folder `figures/` for chromium and firefox.

Playwright can record user interactions (mouse clicks, keyboard input) and create Python code to replay the recorded actions:

```
playwright codegen https://www.bundestag.de/abgeordnete/biografien
```

The created Python code is then modified, here to loop over all overlays showing the members of the parliament:

```
from time import sleep

from playwright.sync_api import sync_playwright

def run(playwright):
    browser = playwright.chromium.launch(headless=False)
    context = browser.new_context(viewport={'height': 1080, 'width': 1920})
    page = context.new_page()
    page.goto("https://www.bundestag.de/abgeordnete/biografien")
    while True:
        try:
            sleep(3)
            page.click("button:has-text(\"Vor\")")
        except Exception:
            break

with sync_playwright() as p:
    run(p)
```

Again: best run the replay script in the console:

```
python ./scripts/playwright_replay.py
```

# 5 Text Processing and Machine Learning

- quick overview on natural language processing
- linear regression and classification
- pre-processing and tokenization (splitting text into words)
- n-grams, vectorization and word embeddings
- train and evaluate a text classifier
- a short look into Hugging Face's transformers library

## 5.1 Natural Language Processing

Natural language processing (NLP) is about programming computers to process and analyze natural language data (text and speech).

For Python, there are two main NLP modules: - spaCy or spaCy on pypi - NLTK or NLTK on pypi

Both modules implement the following NLP applications (and more), at least, for some languates: - named entity recognition (NER) - sentiment detection - tokenization: splitting a text into words (aka. tokens) - part-of-speech tagging (POS) - lemmatization: mapping a word in text to its base form (aka. lemma) - syntax parsing - semantic representation of words

We'll first look into spaCy to explore NLP applications. Later, during the text classification we'll also touch some aspects of NLP, namely tokenization and the semantic representation of words in a vector space (word embeddings).

Work with spaCy also requires to install code modules for the natural languages to be processed:

```
In [1]: %%script false --no-raise-error
        # comment the above instruction to run this code
        #
        # spaCy installation and core modules for English and German
        # - https://spacy.io/models/en
        # - https://spacy.io/models/de
        # note: download modules only once!
        !pip install spacy
        !python -m spacy download en_core_web_sm
        !python -m spacy download de_core_news_sm
```

```
In [2]: import spacy


        nlp = spacy.load("de_core_news_sm")


        # we use a text from notebook 4 "Web Scraping"
        text = 'Geboren am 16. Mai 1980 in Ahaus; römisch-katholisch.Nach dem Abitur Ausbildung zum Bankkaufmann, anschl

        doc = nlp(text)
```

```
doc.to_json()
```

```
Out[2]: {'text': 'Geboren am 16. Mai 1980 in Ahaus; römisch-katholisch.Nach dem Abitur Ausbildung zum Bankkaufmann, ansc
         'ents': [{'start': 27, 'end': 32, 'label': 'LOC'},
          {'start': 189, 'end': 192, 'label': 'ORG'},
          {'start': 221, 'end': 246, 'label': 'ORG'},
          {'start': 280, 'end': 300, 'label': 'MISC'},
          {'start': 394, 'end': 421, 'label': 'LOC'},
          {'start': 451, 'end': 480, 'label': 'ORG'},
          {'start': 536, 'end': 539, 'label': 'ORG'},
          {'start': 540, 'end': 562, 'label': 'MISC'}],
         'sents': [{'start': 0, 'end': 33},
          {'start': 34, 'end': 53},
          {'start': 53, 'end': 124},
          {'start': 125, 'end': 193},
          {'start': 194, 'end': 247},
          {'start': 248, 'end': 301},
          {'start': 301, 'end': 336},
          {'start': 337, 'end': 422},
          {'start': 423, 'end': 481},
          {'start': 482, 'end': 563}],
         'tokens': [{'id': 0,
           'start': 0,
           'end': 7,
           'tag': 'VVPP',
           'pos': 'VERB',
           'morph': 'VerbForm=Part',
           'lemma': 'Geboren',
           'dep': 'ROOT',
           'head': 0},
          {'id': 1,
           'start': 8,
           'end': 10,
           'tag': 'APPRART',
           'pos': 'ADP',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'am',
           'dep': 'mnr',
           'head': 0},
          {'id': 2,
           'start': 11,
           'end': 14,
           'tag': 'ADJA',
           'pos': 'ADJ',
           'morph': 'Case=Dat|Degree=Pos|Gender=Masc|Number=Sing',
           'lemma': '16.',
           'dep': 'nk',
           'head': 3},
          {'id': 3,
           'start': 15,
           'end': 18,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Mai',
           'dep': 'nk',
           'head': 1},
          {'id': 4,
           'start': 19,
           'end': 23,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '1980',
           'dep': 'nk',
           'head': 3},
          {'id': 5,
           'start': 24,
           'end': 26,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'in',
           'dep': 'mo',
           'head': 0},
          {'id': 6,
           'start': 27,
           'end': 32,
           'tag': 'NE',
           'pos': 'PROPN',
           'morph': 'Case=Dat|Gender=Neut|Number=Sing',
           'lemma': 'Ahaus',
           'dep': 'nk',
           'head': 5},
          {'id': 7,
           'start': 32,
           'end': 33,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 0},
          {'id': 8,
           'start': 34,
           'end': 52,
           'tag': 'ADJD',
           'pos': 'ADV',
           'morph': 'Degree=Pos',
           'lemma': 'römisch-katholisch',
           'dep': 'ROOT',
           'head': 8},
          {'id': 9,
           'start': 52,
           'end': 53,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': '.',
           'dep': 'punct',
           'head': 8},
          {'id': 10,
           'start': 53,
           'end': 57,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'Nach',
           'dep': 'ROOT',
           'head': 10},
          {'id': 11,
           'start': 58,
           'end': 61,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Dat|Definite=Def|Gender=Masc|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 12},
          {'id': 12,
           'start': 62,
           'end': 68,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Abitur',
           'dep': 'nk',
           'head': 10},
          {'id': 13,
           'start': 69,
           'end': 79,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Fem|Number=Sing',
           'lemma': 'Ausbildung',
           'dep': 'nk',
           'head': 12},
          {'id': 14,
           'start': 80,
           'end': 83,
           'tag': 'APPRART',
           'pos': 'ADP',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'zum',
           'dep': 'mnr',
           'head': 13},
          {'id': 15,
           'start': 84,
           'end': 96,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Bankkaufmann',
           'dep': 'nk',
           'head': 14},
          {'id': 16,
           'start': 96,
           'end': 97,
           'tag': '$,',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ',',
           'dep': 'punct',
           'head': 10},
          {'id': 17,
           'start': 98,
           'end': 110,
           'tag': 'ADJD',
           'pos': 'ADV',
           'morph': 'Degree=Pos',
           'lemma': 'anschließen',
           'dep': 'mo',
           'head': 18},
          {'id': 18,
           'start': 111,
           'end': 123,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Sing',
           'lemma': 'Angestellter',
           'dep': 'par',
           'head': 10},
          {'id': 19,
           'start': 123,
           'end': 124,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 18},
          {'id': 20,
           'start': 125,
           'end': 132,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Neut|Number=Sing',
           'lemma': 'Studium',
           'dep': 'ROOT',
           'head': 20},
          {'id': 21,
           'start': 133,
           'end': 136,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Neut|Number=Plur|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 22},
          {'id': 22,
           'start': 137,
           'end': 158,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Neut|Number=Plur',
           'lemma': 'Politikwissenschaften',
           'dep': 'ag',
           'head': 20},
          {'id': 23,
           'start': 159,
           'end': 160,
           'tag': '$(',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': '(',
           'dep': 'punct',
           'head': 24},
          {'id': 24,
           'start': 160,
           'end': 170,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'M.A.).Seit',
           'dep': 'par',
           'head': 20},
          {'id': 25,
           'start': 171,
           'end': 175,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '1997',
           'dep': 'nk',
           'head': 24},
          {'id': 26,
           'start': 176,
           'end': 184,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Neut|Number=Sing',
           'lemma': 'Mitglied',
           'dep': 'nk',
           'head': 24},
          {'id': 27,
           'start': 185,
           'end': 188,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Fem|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 28},
          {'id': 28,
           'start': 189,
           'end': 192,
           'tag': 'NE',
           'pos': 'PROPN',
           'morph': 'Case=Gen|Gender=Fem|Number=Sing',
           'lemma': 'CDU',
           'dep': 'ag',
           'head': 26},
          {'id': 29,
           'start': 192,
           'end': 193,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 24},
          {'id': 30,
           'start': 194,
           'end': 198,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'seit',
           'dep': 'ROOT',
           'head': 30},
          {'id': 31,
           'start': 199,
           'end': 203,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2005',
           'dep': 'nk',
           'head': 32},
          {'id': 32,
           'start': 204,
           'end': 216,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Sing',
           'lemma': 'Vorsitzender',
           'dep': 'nk',
           'head': 30},
          {'id': 33,
           'start': 217,
           'end': 220,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Masc|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 34},
          {'id': 34,
           'start': 221,
           'end': 239,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Masc|Number=Sing',
           'lemma': 'CDU-Kreisverbandes',
           'dep': 'ag',
           'head': 32},
          {'id': 35,
           'start': 240,
           'end': 246,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Neut|Number=Sing',
           'lemma': 'Borke',
           'dep': 'nk',
           'head': 34},
          {'id': 36,
           'start': 246,
           'end': 247,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 30},
          {'id': 37,
           'start': 248,
           'end': 252,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'seit',
           'dep': 'mo',
           'head': 40},
          {'id': 38,
           'start': 253,
           'end': 261,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Dezember',
           'dep': 'nk',
           'head': 37},
          {'id': 39,
           'start': 262,
           'end': 266,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2014',
           'dep': 'nk',
           'head': 38},
          {'id': 40,
           'start': 267,
           'end': 275,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Neut|Number=Sing',
           'lemma': 'Mitglied',
           'dep': 'ROOT',
           'head': 40},
          {'id': 41,
           'start': 276,
           'end': 279,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Neut|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 42},
          {'id': 42,
           'start': 280,
           'end': 300,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Neut|Number=Sing',
           'lemma': 'CDU-Bundespräsidiums',
           'dep': 'ag',
           'head': 40},
          {'id': 43,
           'start': 300,
           'end': 301,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': '.',
           'dep': 'punct',
           'head': 40},
          {'id': 44,
           'start': 301,
           'end': 309,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Neut|Number=Sing',
           'lemma': 'Mitglied',
           'dep': 'ROOT',
           'head': 44},
          {'id': 45,
           'start': 310,
           'end': 313,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Masc|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 46},
          {'id': 46,
           'start': 314,
           'end': 325,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Masc|Number=Sing',
           'lemma': 'Bundestag',
           'dep': 'ag',
           'head': 44},
          {'id': 47,
           'start': 326,
           'end': 330,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'seit',
           'dep': 'mnr',
           'head': 44},
          {'id': 48,
           'start': 331,
           'end': 335,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2002',
           'dep': 'nk',
           'head': 47},
          {'id': 49,
           'start': 335,
           'end': 336,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 44},
          {'id': 50,
           'start': 337,
           'end': 341,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2015',
           'dep': 'nmc',
           'head': 53},
          {'id': 51,
           'start': 342,
           'end': 345,
           'tag': 'APPR',
           'pos': 'CCONJ',
           'morph': '',
           'lemma': 'bis',
           'dep': 'cd',
           'head': 50},
          {'id': 52,
           'start': 346,
           'end': 350,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Acc|Gender=Masc|Number=Sing',
           'lemma': 'März',
           'dep': 'cj',
           'head': 51},
          {'id': 53,
           'start': 351,
           'end': 355,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2018',
           'dep': 'nk',
           'head': 55},
          {'id': 54,
           'start': 356,
           'end': 373,
           'tag': 'ADJA',
           'pos': 'ADJ',
           'morph': 'Case=Nom|Degree=Pos|Gender=Masc|Number=Sing',
           'lemma': 'Parlamentarischer',
           'dep': 'nk',
           'head': 55},
          {'id': 55,
           'start': 374,
           'end': 388,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Sing',
           'lemma': 'Staatssekretär',
           'dep': 'ROOT',
           'head': 55},
          {'id': 56,
           'start': 389,
           'end': 393,
           'tag': 'APPRART',
           'pos': 'ADP',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'beim',
           'dep': 'mnr',
           'head': 55},
          {'id': 57,
           'start': 394,
           'end': 408,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Bundesminister',
           'dep': 'nk',
           'head': 56},
          {'id': 58,
           'start': 409,
           'end': 412,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Fem|Number=Plur|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 59},
          {'id': 59,
           'start': 413,
           'end': 421,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Fem|Number=Plur',
           'lemma': 'Finanz',
           'dep': 'ag',
           'head': 57},
          {'id': 60,
           'start': 421,
           'end': 422,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 55},
          {'id': 61,
           'start': 423,
           'end': 427,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Sing',
           'lemma': 'März',
           'dep': 'ROOT',
           'head': 61},
          {'id': 62,
           'start': 428,
           'end': 432,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2018',
           'dep': 'nk',
           'head': 61},
          {'id': 63,
           'start': 433,
           'end': 436,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'bis',
           'dep': 'mo',
           'head': 61},
          {'id': 64,
           'start': 437,
           'end': 445,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Acc|Gender=Masc|Number=Sing',
           'lemma': 'Dezember',
           'dep': 'nk',
           'head': 63},
          {'id': 65,
           'start': 446,
           'end': 450,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2021',
           'dep': 'nk',
           'head': 66},
          {'id': 66,
           'start': 451,
           'end': 465,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Plur',
           'lemma': 'Bundesminister',
           'dep': 'pd',
           'head': 61},
          {'id': 67,
           'start': 466,
           'end': 469,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'für',
           'dep': 'mnr',
           'head': 66},
          {'id': 68,
           'start': 470,
           'end': 480,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Acc|Gender=Fem|Number=Sing',
           'lemma': 'Gesundheit',
           'dep': 'nk',
           'head': 67},
          {'id': 69,
           'start': 480,
           'end': 481,
           'tag': '$.',
           'pos': 'PUNCT',
           'morph': '',
           'lemma': ';',
           'dep': 'punct',
           'head': 61},
          {'id': 70,
           'start': 482,
           'end': 486,
           'tag': 'APPR',
           'pos': 'ADP',
           'morph': '',
           'lemma': 'seit',
           'dep': 'ROOT',
           'head': 70},
          {'id': 71,
           'start': 487,
           'end': 495,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Dat|Gender=Masc|Number=Sing',
           'lemma': 'Dezember',
           'dep': 'nk',
           'head': 70},
          {'id': 72,
           'start': 496,
           'end': 500,
           'tag': 'CARD',
           'pos': 'NUM',
           'morph': '',
           'lemma': '2021',
           'dep': 'nk',
           'head': 71},
          {'id': 73,
           'start': 501,
           'end': 518,
           'tag': 'ADJA',
           'pos': 'ADJ',
           'morph': 'Case=Nom|Degree=Pos|Gender=Masc|Number=Sing',
           'lemma': 'stellvertretend',
           'dep': 'nk',
           'head': 74},
          {'id': 74,
           'start': 519,
           'end': 531,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Nom|Gender=Masc|Number=Sing',
           'lemma': 'Vorsitzender',
           'dep': 'pd',
           'head': 70},
          {'id': 75,
           'start': 532,
           'end': 535,
           'tag': 'ART',
           'pos': 'DET',
           'morph': 'Case=Gen|Definite=Def|Gender=Fem|Number=Sing|PronType=Art',
           'lemma': 'der',
           'dep': 'nk',
           'head': 76},
          {'id': 76,
           'start': 536,
           'end': 539,
           'tag': 'NN',
           'pos': 'NOUN',
           'morph': 'Case=Gen|Gender=Fem|Number=Sing',
           'lemma': 'CDU',
           'dep': 'ag',
           'head': 74},
            {'id': 77,
             'start': 539,
             'end': 540,
             'tag': 'NN',
             'pos': 'NOUN',
             'morph': 'Case=Gen|Gender=Fem|Number=Sing',
             'lemma': '/',
             'dep': 'ag',
             'head': 74},
            {'id': 78,
             'start': 540,
             'end': 562,
             'tag': 'NN',
             'pos': 'NOUN',
             'morph': 'Case=Gen|Gender=Fem|Number=Sing',
             'lemma': 'CSU-Bundestagsfraktion',
             'dep': 'ag',
             'head': 74},
            {'id': 79,
             'start': 562,
             'end': 563,
             'tag': '$.',
             'pos': 'PUNCT',
             'morph': '',
             'lemma': '.',
             'dep': 'punct',
             'head': 70}]}
```

What does the JSON representation of the short text contain? Which NLP applications are involved?

```
In [3]: # filter tokens tagged as nouns
        list(filter(lambda t: t.pos_ == 'NOUN', doc))
```

```
Out[3]: [Mai,
         Abitur,
         Ausbildung,
         Bankkaufmann,
         Angestellter,
         Studium,
         Politikwissenschaften,
         Mitglied,
         Vorsitzender,
         CDU-Kreisverbandes,
         Borken,
         Dezember,
         Mitglied,
         CDU-Bundespräsidiums,
         Mitglied,
         Bundestages,
         März,
         Staatssekretär,
         Bundesminister,
         Finanzen,
         März,
         Dezember,
         Bundesminister,
         Gesundheit,
         Dezember,
         Vorsitzender,
         CDU,
         /,
         CSU-Bundestagsfraktion]
```

For some NLP applications, spaCy provides nice visualizations: - named entities - syntax trees of dependency parsing

```
In [4]: from spacy import displacy

        displacy.render(doc, style="ent")
```

```
<IPython.core.display.HTML object>
```

```
In [5]: displacy.render(doc, style="dep")
```

```
<IPython.core.display.HTML object>
```

## 5.2 Machine Learning

The field of machine learning is too broad to be fully introduced here. Please, see Google's machine learning crash course. We'll focus on a couple of examples and introduce ML libraries written in or providing a Python API.

- scikit-learn: popular Python ML framework covering regression, classification and clustering using various approaches
- fastText: a library for text classification and word representation learning with Python bindings
- TensorFlow: ML framework with Python bindings focused on deep neural networks
- Keras: high-level API to Tensorflow
- PyTorch: competitor of Tensorflow
- Transformers: library to use, train and adapt transformer deep learning models

Before we begin to look into Python ML examples, few ML key terms: - label: something we want to predict - feature: variable in the input (eg. numeric value, words) - example: data to learn from during training (labeled example) or to predict the label for using a learned model - model: a model is trained on labeled input data and later used to make predictions ("infer" labels) for unlabeled examples - regression vs. classification: labels are continuous vs. categorical values

## 5.3 Linear Regression and Classification with Scikit-Learn

As an example for linear regression we take few trees from the tree cadastre used in notebook 2. We select a small subset of trees species to work with.

```
In [40]: %matplotlib notebook

         import pandas as pd
         import matplotlib.pyplot as plt

         tree_cadastre_file = 'data/KN_Baumkataster_2020.csv'
         df = pd.read_csv(tree_cadastre_file)

         df.rename(columns={'hoeheM': 'height (m)',
                            'kronendurchmesserM': 'treetop diameter (m)',
                            'stammumfangCM': 'trunk perimeter (cm)'},
                   inplace=True)
         df_all_trees = df

         # could try the top N trees
         #N = 6
         #top_trees = df['Name_lat'].value_counts().head(N).to_frame()
         #selected_trees = top_trees.index

         # instead, we choose 3 trees quite different in shape:
         # - birch : tall and high, thinner trunk
         # - lime tree : broad, thicker trunk
         # - apple tree : small, not tall
         selected_trees = ['Betula pendula', 'Tilia cordata', 'Malus domestica']

         metric_columns = ['trunk perimeter (cm)', 'treetop diameter (m)', 'height (m)']

         df = df[df['Name_lat'].isin(selected_trees)][['Name_lat', *metric_columns]]

         # prepare a 3D plot to show how the trees are placed given the 3 metrics
         fig = plt.figure()
         ax = fig.add_subplot(projection='3d')

         for name, idx in df.groupby('Name_lat').groups.items():
             ax.scatter(*df.loc[idx, metric_columns].T.values, label=name)

         ax.set_xlabel(metric_columns[0])
         ax.set_ylabel(metric_columns[1])
         ax.set_zlabel(metric_columns[2])

         ax.legend()
         plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [41]: # linear regression: predict trunk perimeter and treetop diameter given the height

         import numpy as np
         from sklearn.linear_model import LinearRegression

         for tree in selected_trees:

             # select rows by tree in loop
             d = df[df['Name_lat']==tree].dropna()

             # convert metric cells to numpy arrays
             height = d[metric_columns[2]].values.reshape(-1,1)
             treetop_trunk = d[metric_columns[0:2]].values.reshape(-1,2)

             rgr = LinearRegression()
             rgr.fit(height, treetop_trunk)

             print(tree)
             for height in [2, 5, 10, 15, 20]:
                 print(height, rgr.predict(np.array([[height]])))
             print()
```

```
Betula pendula
2 [[18.50448984  1.13475045]]
5 [[40.26100411  2.42970328]]
10 [[76.52186123  4.587958  ]]
15 [[112.78271835   6.74621271]]
20 [[149.04357547   8.90446743]]

Tilia cordata
2 [[21.35878424  1.8773699 ]]
5 [[54.22855546  3.59077637]]
10 [[109.01150751   6.4464538 ]]
15 [[163.79445956   9.30213124]]
20 [[218.57741161  12.15780867]]

Malus domestica
2 [[24.58206233  1.4661875 ]]
5 [[67.07429191  4.07683903]]
10 [[137.89467456   8.42792492]]
15 [[208.71505721  12.77901081]]
20 [[279.53543986  17.1300967 ]]
```

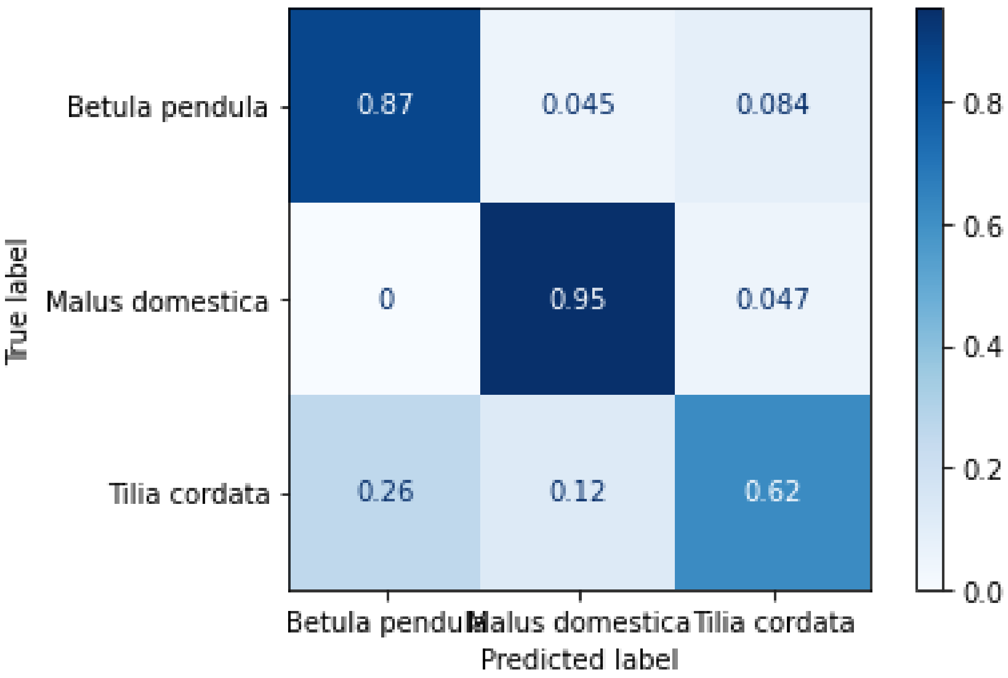
See also the scikit-learn documentation about linear models.

The neural network classifier used in the next example uses as input features the 3 metric columns and tries to predict the species of a tree. How are the results given that only 3 tree species are used. What if we use more or even all species?

```
In [43]: import sklearn

         from sklearn.neural_network import MLPClassifier
         from sklearn.model_selection import train_test_split

         # split data into train and test data (80% resp. 20% of the data)
         train, test = train_test_split(df.dropna(), test_size=0.2)

         cls = MLPClassifier(alpha=1, max_iter=1000)

         x_train = train[metric_columns].values.reshape(-1,3)
         y_train = train[['Name_lat']].values.reshape(-1,1).ravel()

         x_test = test[metric_columns].values.reshape(-1,3)
         y_test = test[['Name_lat']].values.reshape(-1,1).ravel()

         cls.fit(x_train, y_train)

         # print results for predictions on test data
         y_predicted = cls.predict(x_test)
         print(
             f"Classification report for classifier {cls}:\n"
             f"{sklearn.metrics.classification_report(y_test, y_predicted)}\n"
         )
```

```
Classification report for classifier MLPClassifier(alpha=1, max_iter=1000):
                 precision    recall  f1-score   support

 Betula pendula       0.80      0.87      0.83       154
Malus domestica       0.81      0.95      0.88       106
  Tilia cordata       0.82      0.62      0.70       131

       accuracy                           0.81       391
      macro avg       0.81      0.81      0.80       391
   weighted avg       0.81      0.81      0.80       391
```

```
In [44]: # print a confusion matrix: which tree species are predicted better? which ones are confused more often?
         %matplotlib inline

         sklearn.metrics.ConfusionMatrixDisplay.from_predictions(y_test, y_predicted, normalize='true', cmap='Blues')

Out[44]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd734e35df0>
```



## 5.4 Text Classification with fastText

fastText is a software library for text classification and word representation learning. See the fastText tutorials for

- text classification
- word representation learning

We will now follow the fastText text classification tutorial (cf. documentation of the Python module "fasttext") to train and apply a text classifier.

The fastText tutorial uses the StackExchange cooking data set. We will use the Kaggle Toxic Comment Classification Challenge data set. In order to download the data set, you need to register at Kaggle.com. Note: Kaggle is a good place to look and learn how other researchers and engineers tried to solve various ML problems.

After the data set is downloaded and unpacked into the folder `data/kaggle-jigsaw-toxic`, you should see the tree files `train.csv`, `test.csv` and `test_labels.csv` in the mentioned folder.

```
In [6]: import pandas as pd

        df_train = pd.read_csv('data/kaggle-jigsaw-toxic/train.csv')

        #df.head()

In [7]: labels = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']

        df_train[labels].mean()
```

```
Out[7]: toxic            0.095844
        severe_toxic     0.009996
        obscene          0.052948
        threat           0.002996
        insult           0.049364
        identity_hate    0.008805
        dtype: float64
```

Only 10% of the comments are toxic. What does it mean for building a classifier?

```
In [8]: # tokenize the comments
        import string

        from nltk.tokenize import TweetTokenizer

        tweet_tokenizer = TweetTokenizer(reduce_len=True)

        def tokenize(text):
            global tweet_tokenizer
            words = tweet_tokenizer.tokenize(text)
            words = filter(lambda w: w != ''
                                     and w not in string.punctuation, words)
            words = map(lambda w: w.lower(), words)
            return ' '.join(words)

        tokenize("You're a hero! http://example.com/index.html")
```

```
Out[8]: "you're a hero http://example.com/index.html"
```

```
In [9]: # write data to fastText train file

        train_file = 'data/kaggle-jigsaw-toxic/train.txt'

        def write_line_fasttext(fp, row):
            global labels
            line = ''
            for label in labels:
                if row[label] == 1:
                    if line:
                        line += ' '
                    line += '__label__' + label
            if line:
                line += ' '
            else:
                line += '__label__none '
            line += tokenize(row['comment_text'])
            fp.write(line)
            fp.write('\n')

        with open(train_file, 'w') as fp:
            df_train.apply(lambda row: write_line_fasttext(fp, row), axis=1)
```

```
In [10]: !pip install fasttext
```

```
Requirement already satisfied: fasttext in ./.venv/lib/python3.9/site-packages (0.9.2)
Requirement already satisfied: pybind11>=2.2 in ./.venv/lib/python3.9/site-packages (from fasttext) (2.9.0)
Requirement already satisfied: setuptools>=0.7.0 in ./.venv/lib/python3.9/site-packages (from fasttext) (44.1.1)
Requirement already satisfied: numpy in ./.venv/lib/python3.9/site-packages (from fasttext) (1.22.1)
```

```
In [11]: # train a model

         import fasttext

         model = fasttext.train_supervised(input=train_file, wordNgrams=2, minCount=2)
```

```
In [38]: model.predict(tokenize("This is a well-written article."))
         # model.predict(tokenize("Fuck you!"), k=5)
```

```
Out[38]: (('__label__obscene',
           '__label__insult',
           '__label__toxic',
           '__label__severe_toxic',
           '__label__identity_hate'),
          array([0.41425505, 0.32454872, 0.18850109, 0.06784596, 0.00485881]))
```

```
In [13]: # looking into the underlying word embeddings

         model.get_nearest_neighbors('idiot', k=20)
```

```
Out[13]: [(0.9997887015342712, 'stupid'),
          (0.9996612668037415, 'moron'),
          (0.9996005296707153, 'jerk'),
          (0.9994908571243286, 'arrogant'),
          (0.999340832233429, 'stupidity'),
          (0.999234139919281, 'pathetic'),
          (0.9992112517356873, 'coward'),
          (0.9991973042488098, 'fool'),
          (0.9991273880004883, 'ignorant'),
          (0.9991151690483093, 'disgusting'),
          (0.9990187883377075, 'idiotic'),
          (0.9989598393440247, 'jackass'),
          (0.9988994598388672, 'fascist'),
          (0.9988807439804077, 'morons'),
          (0.9988119602203369, 'fat'),
          (0.9987836480140686, 'hell'),
          (0.998756468296051, 'bloody'),
          (0.9987396001815796, 'sucked'),
          (0.9987378716468811, 'anal'),
          (0.9987097382545471, 'losers')]
```

```
In [14]: # save the model
         model_file = 'data/kaggle-jigsaw-toxic/model.bin'

         model.save_model(model_file)
```

```
In [15]: df_test = pd.read_csv('data/kaggle-jigsaw-toxic/test.csv')
         df_test_labels = pd.read_csv('data/kaggle-jigsaw-toxic/test_labels.csv')

         # join both tables
         df_test = df_test.merge(df_test_labels, on='id')

         # skip rows not labelled / not used
         df_test = df_test[df_test['toxic'] != -1]

         test_file = 'data/kaggle-jigsaw-toxic/test.txt'

         # write test set for fastText
         with open(test_file, 'w') as fp:
             df_test.apply(lambda row: write_line_fasttext(fp, row), axis=1)
```

### 5.4.1 Model Validation

See also: precision and recall

```
In [16]: model.test(test_file)

Out[16]: (63978, 0.9302416455656632, 0.8239308903132917)
```

```
In [17]: res_per_label = model.test_label(test_file)

         for label in res_per_label.items():
             print(label)
```

```
('__label__threat', {'precision': nan, 'recall': 0.0, 'f1score': 0.0})
('__label__identity_hate', {'precision': nan, 'recall': 0.0, 'f1score': 0.0})
('__label__severe_toxic', {'precision': 0.3076923076923077, 'recall': 0.07629427792915532, 'f1score': 0.1222707423580786
('__label__insult', {'precision': 0.75, 'recall': 0.005252407353370295, 'f1score': 0.010431758910460736})
('__label__obscene', {'precision': 0.9448275862068966, 'recall': 0.1113519371444053, 'f1score': 0.1992244304411052})
('__label__toxic', {'precision': 0.5900198468953786, 'recall': 0.683415435139573, 'f1score': 0.6332927571515521})
('__label__none', {'precision': 0.9737822400397347, 'recall': 0.9508270546462284, 'f1score': 0.9621677518863543})
```

```
In [37]: # in case the fastText command-line tool is installed: it has a nice output formatter
         !fasttext test-label \
            data/kaggle-jigsaw-toxic/model.bin \
            data/kaggle-jigsaw-toxic/test.txt
```

```
F1-Score : 0.962168  Precision : 0.973782  Recall : 0.950827   __label__none
F1-Score : 0.633293  Precision : 0.590020  Recall : 0.683415   __label__toxic
F1-Score : 0.199224  Precision : 0.944828  Recall : 0.111352   __label__obscene
F1-Score : 0.010432  Precision : 0.750000  Recall : 0.005252   __label__insult
F1-Score : 0.122271  Precision : 0.307692  Recall : 0.076294   __label__severe_toxic
F1-Score : 0.000000  Precision : --------  Recall : 0.000000   __label__identity_hate
F1-Score : 0.000000  Precision : --------  Recall : 0.000000   __label__threat
N        63978
P@1        0.930
R@1        0.824
```

## 5.5 Transformer Language Models and the Transformers Library

Transformer language models are used to address a couple of NLP tasks – text classification, text generation, translation and more. Hugging Face's transformers library provides an powerful and easy to learn interface to use transformers. Hugging Face also offers a large repository of transformer models shared by a growing community of researchers and organizations. For more details exceeding the examples below, see the transformers course.

Transformers can be "fine-tuned" to a specific task, see training of transformers. Adding a task-specific head to a transformer pre-trained on large amounts of training data (usually 100 GBs or even TBs of text) saves resources spent for training and can overcome the problem of not enough training data. Manually labelling training data is expensive and naturally puts a limit on the amount of training data. But even if the vocabulary in the training data is limited, there's a good chance that the pre-trained transformer has seen the unknown words in the huge data used for pre-training.

```
In [19]: !pip install transformers
         !pip install tensorflow
         !pip install "transformers[sentencepiece]"
```

```
Requirement already satisfied: transformers in ./.venv/lib/python3.9/site-packages (4.15.0)
Requirement already satisfied: numpy>=1.17 in ./.venv/lib/python3.9/site-packages (from transformers) (1.22.1)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in ./.venv/lib/python3.9/site-packages (from transformers) (0.10
Requirement already satisfied: regex!=2019.12.17 in ./.venv/lib/python3.9/site-packages (from transformers) (2022.1.18)
Requirement already satisfied: requests in ./.venv/lib/python3.9/site-packages (from transformers) (2.27.1)
Requirement already satisfied: sacremoses in ./.venv/lib/python3.9/site-packages (from transformers) (0.0.47)
Requirement already satisfied: tqdm>=4.27 in ./.venv/lib/python3.9/site-packages (from transformers) (4.62.3)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in ./.venv/lib/python3.9/site-packages (from transformers) (0
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.9/site-packages (from transformers) (21.3)
Requirement already satisfied: pyyaml>=5.1 in ./.venv/lib/python3.9/site-packages (from transformers) (6.0)
Requirement already satisfied: filelock in ./.venv/lib/python3.9/site-packages (from transformers) (3.4.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in ./.venv/lib/python3.9/site-packages (from huggingface-hub<1
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in ./.venv/lib/python3.9/site-packages (from packaging>=20.0->tr
Requirement already satisfied: charset-normalizer~=2.0.0 in ./.venv/lib/python3.9/site-packages (from requests->transfor
Requirement already satisfied: certifi>=2017.4.17 in ./.venv/lib/python3.9/site-packages (from requests->transformers) (
Requirement already satisfied: idna<4,>=2.5 in ./.venv/lib/python3.9/site-packages (from requests->transformers) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./.venv/lib/python3.9/site-packages (from requests->transformers
Requirement already satisfied: click in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers) (8.0.3)
Requirement already satisfied: joblib in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers) (1.1.0)
Requirement already satisfied: six in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers) (1.16.0)
Requirement already satisfied: tensorflow in ./.venv/lib/python3.9/site-packages (2.7.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.43.0)
Requirement already satisfied: keras<2.8,>=2.7.0rc0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: wrapt>=1.11.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.13.3)
Requirement already satisfied: six>=1.12.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: protobuf>=3.9.2 in ./.venv/lib/python3.9/site-packages (from tensorflow) (3.19.3)
Requirement already satisfied: h5py>=2.9.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in ./.venv/lib/python3.9/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in ./.venv/lib/python3.9/site-packages (from tensorf
Requirement already satisfied: wheel<1.0,>=0.32.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (0.34.2)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in ./.venv/lib/python3.9/site-packages (from tensorflow) (2.0)
Requirement already satisfied: astunparse>=1.6.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: libclang>=9.0.1 in ./.venv/lib/python3.9/site-packages (from tensorflow) (12.0.0)
Requirement already satisfied: numpy>=1.14.5 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.22.1)
Requirement already satisfied: gast<0.5.0,>=0.2.1 in ./.venv/lib/python3.9/site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in ./.venv/lib/python3.9/site-packages (from tensorflow) (4.0.1)
Requirement already satisfied: opt-einsum>=2.3.2 in ./.venv/lib/python3.9/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: tensorboard~=2.6 in ./.venv/lib/python3.9/site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: termcolor>=1.1.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: absl-py>=0.4.0 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.0.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in ./.venv/lib/python3.9/site-packages (from tensorflow) (1.1.
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in ./.venv/lib/python3.9/site-packages (from tensorfl
Requirement already satisfied: google-auth<3,>=1.6.3 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2.6->tens
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2
Requirement already satisfied: requests<3,>=2.21.0 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2.6->tensor
Requirement already satisfied: werkzeug>=0.11.15 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2.6->tensorfl
Requirement already satisfied: setuptools>=41.0.0 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2.6->tensorf
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in ./.venv/lib/python3.9/site-packages (from tensorboard
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in ./.venv/lib/python3.9/site-packages (from tensor
Requirement already satisfied: markdown>=2.6.8 in ./.venv/lib/python3.9/site-packages (from tensorboard~=2.6->tensorflow
Requirement already satisfied: cachetools<5.0,>=2.0.0 in ./.venv/lib/python3.9/site-packages (from google-auth<3,>=1.6.3
Requirement already satisfied: pyasn1-modules>=0.2.1 in ./.venv/lib/python3.9/site-packages (from google-auth<3,>=1.6.3-
Requirement already satisfied: rsa<5,>=3.1.4 in ./.venv/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: requests-oauthlib>=0.7.0 in ./.venv/lib/python3.9/site-packages (from google-auth-oauthli
Requirement already satisfied: importlib-metadata>=4.4 in ./.venv/lib/python3.9/site-packages (from markdown>=2.6.8->ten
Requirement already satisfied: zipp>=0.5 in ./.venv/lib/python3.9/site-packages (from importlib-metadata>=4.4->markdown>
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in ./.venv/lib/python3.9/site-packages (from pyasn1-modules>=0.2.1->
Requirement already satisfied: idna<4,>=2.5 in ./.venv/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboar
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./.venv/lib/python3.9/site-packages (from requests<3,>=2.21.0->t
Requirement already satisfied: charset-normalizer~=2.0.0 in ./.venv/lib/python3.9/site-packages (from requests<3,>=2.21.
Requirement already satisfied: certifi>=2017.4.17 in ./.venv/lib/python3.9/site-packages (from requests<3,>=2.21.0->tens
Requirement already satisfied: oauthlib>=3.0.0 in ./.venv/lib/python3.9/site-packages (from requests-oauthlib>=0.7.0->go
Requirement already satisfied: transformers[sentencepiece] in ./.venv/lib/python3.9/site-packages (4.15.0)
Requirement already satisfied: requests in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (2.27.
Requirement already satisfied: filelock in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (3.4.2
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece])
Requirement already satisfied: tqdm>=4.27 in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (4.6
Requirement already satisfied: numpy>=1.17 in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (1.
Requirement already satisfied: pyyaml>=5.1 in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (6.
Requirement already satisfied: sacremoses in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (0.0
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in ./.venv/lib/python3.9/site-packages (from transformers[sen
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in ./.venv/lib/python3.9/site-packages (from transformers[senten
Requirement already satisfied: regex!=2019.12.17 in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece
Requirement already satisfied: sentencepiece!=0.1.92,>=0.1.91 in ./.venv/lib/python3.9/site-packages (from transformers[
Requirement already satisfied: protobuf in ./.venv/lib/python3.9/site-packages (from transformers[sentencepiece]) (3.19.
Requirement already satisfied: typing-extensions>=3.7.4.3 in ./.venv/lib/python3.9/site-packages (from huggingface-hub<1
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in ./.venv/lib/python3.9/site-packages (from packaging>=20.0->tr
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./.venv/lib/python3.9/site-packages (from requests->transformers
Requirement already satisfied: idna<4,>=2.5 in ./.venv/lib/python3.9/site-packages (from requests->transformers[sentence
Requirement already satisfied: certifi>=2017.4.17 in ./.venv/lib/python3.9/site-packages (from requests->transformers[se
Requirement already satisfied: charset-normalizer~=2.0.0 in ./.venv/lib/python3.9/site-packages (from requests->transfor
Requirement already satisfied: click in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers[sentencepiece
Requirement already satisfied: six in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers[sentencepiece])
Requirement already satisfied: joblib in ./.venv/lib/python3.9/site-packages (from sacremoses->transformers[sentencepiec
```

```
In [20]: from transformers import pipeline

         p = pipeline('fill-mask', model='bert-base-german-cased')
```

```
Some weights of the model checkpoint at bert-base-german-cased were not used when initializing BertForMaskedLM: ['cls.se
- This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model trained on another task or wit
- This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a model that you expect to be exac
```

```
In [21]: for s in p("Er arbeitet als [MASK]."): print(s)
```

```
{'sequence': 'Er arbeitet als Rechtsanwalt.', 'score': 0.09919334203004837, 'token': 6143, 'token_str': 'Rechtsanwalt'}
{'sequence': 'Er arbeitet als Trainer.', 'score': 0.07836302369832993, 'token': 3674, 'token_str': 'Trainer'}
{'sequence': 'Er arbeitet als Journalist.', 'score': 0.0628521665930748, 'token': 10486, 'token_str': 'Journalist'}
{'sequence': 'Er arbeitet als Anwalt.', 'score': 0.05725342780351639, 'token': 6938, 'token_str': 'Anwalt'}
{'sequence': 'Er arbeitet als Schauspieler.', 'score': 0.05046413466334343, 'token': 5607, 'token_str': 'Schauspieler'}
```

```
In [22]: pipeline_fill_mask = pipeline('fill-mask', model='bert-base-german-cased')

         def fill_mask(cloze):
             global pipeline_fill_mask
             for s in pipeline_fill_mask(cloze):
                 print('%-20s\t%.5f' % (s['token_str'], s['score']))
```

```
Some weights of the model checkpoint at bert-base-german-cased were not used when initializing BertForMaskedLM: ['cls.se
- This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model trained on another task or wit
- This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a model that you expect to be exac
```

```
In [23]: fill_mask("Er arbeitet als [MASK] in einer Klinik.")
```

```
Arzt                        0.61843
Angestellter                0.04225
Koch                        0.03064
Assistent                   0.02001
Mediziner                   0.01900
```

```
In [24]: fill_mask("Er arbeitet als [MASK] in einer Lungenklinik.")
```

```
Arzt                        0.69560
Angestellter                0.03423
Chemiker                    0.02711
Facharzt                    0.02113
Mediziner                   0.02024
```

```
In [25]: fill_mask("Er arbeitet als [MASK] bei BMW.")
```

```
Ingenieur                    0.18871
Berater                      0.17160
Manager                      0.15090
Geschäftsführer              0.07775
Trainer                      0.04951
```

```
In [26]: fill_mask("Er arbeitet als [MASK] an der Universität Konstanz.")
```

```
Professor                    0.74687
Dozent                       0.11445
Hochschullehrer              0.08565
Wissenschaftler              0.00667
Assistent                    0.00427
```

```
In [27]: fill_mask("Sie arbeitet als [MASK] an der Universität Konstanz.")
```

```
Professor                    0.52318
Lehrerin                     0.09859
Dozent                       0.08542
Professur                    0.04144
Richterin                    0.02292
```

```
In [28]: fill_mask("Sie ist wirklich [MASK].")
```

```
schön                        0.11005
jung                         0.06098
glücklich                    0.05704
toll                         0.05053
gut                          0.03495
```

```
In [29]: fill_mask("Er ist wirklich [MASK].")
```

```
gut                          0.05452
glücklich                    0.05183
da                           0.03765
jung                         0.03233
tot                          0.03229
```

```
In [30]: help(pipeline)
```

```
Help on function pipeline in module transformers.pipelines:

pipeline(task: str, model: Optional = None, config: Union[str, transformers.configuration_utils.PretrainedConfig, NoneTy
    Utility factory method to build a :class:`~transformers.Pipeline`.

    Pipelines are made of:

        - A :doc:`tokenizer <tokenizer>` in charge of mapping raw textual input to token.
        - A :doc:`model <model>` to make predictions from the inputs.
        - Some (optional) post processing for enhancing model's output.

    Args:
        task (:obj:`str`):
            The task defining which pipeline will be returned. Currently accepted tasks are:

            - :obj:`"audio-classification"`: will return a :class:`~transformers.AudioClassificationPipeline`:.
            - :obj:`"automatic-speech-recognition"`: will return a
              :class:`~transformers.AutomaticSpeechRecognitionPipeline`:.
            - :obj:`"conversational"`: will return a :class:`~transformers.ConversationalPipeline`:.
            - :obj:`"feature-extraction"`: will return a :class:`~transformers.FeatureExtractionPipeline`:.
            - :obj:`"fill-mask"`: will return a :class:`~transformers.FillMaskPipeline`:.
            - :obj:`"image-classification"`: will return a :class:`~transformers.ImageClassificationPipeline`:.
            - :obj:`"question-answering"`: will return a :class:`~transformers.QuestionAnsweringPipeline`:.
            - :obj:`"table-question-answering"`: will return a :class:`~transformers.TableQuestionAnsweringPipeline`:.
            - :obj:`"text2text-generation"`: will return a :class:`~transformers.Text2TextGenerationPipeline`:.
            - :obj:`"text-classification"` (alias :obj:`"sentiment-analysis" available): will return a
              :class:`~transformers.TextClassificationPipeline`:.
            - :obj:`"text-generation"`: will return a :class:`~transformers.TextGenerationPipeline`:.
            - :obj:`"token-classification"` (alias :obj:`"ner"` available): will return a
              :class:`~transformers.TokenClassificationPipeline`:.
            - :obj:`"translation"`: will return a :class:`~transformers.TranslationPipeline`:.
            - :obj:`"translation_xx_to_yy"`: will return a :class:`~transformers.TranslationPipeline`:.
            - :obj:`"summarization"`: will return a :class:`~transformers.SummarizationPipeline`:.
            - :obj:`"zero-shot-classification"`: will return a :class:`~transformers.ZeroShotClassificationPipeline`:.

        model (:obj:`str` or :obj:`~transformers.PreTrainedModel` or :obj:`~transformers.TFPreTrainedModel`, `optional`)
            The model that will be used by the pipeline to make predictions. This can be a model identifier or an
            actual instance of a pretrained model inheriting from :class:`~transformers.PreTrainedModel` (for PyTorch)
            or :class:`~transformers.TFPreTrainedModel` (for TensorFlow).

            If not provided, the default for the :obj:`task` will be loaded.
        config (:obj:`str` or :obj:`~transformers.PretrainedConfig`, `optional`):
            The configuration that will be used by the pipeline to instantiate the model. This can be a model
            identifier or an actual pretrained model configuration inheriting from
            :class:`~transformers.PretrainedConfig`.

            If not provided, the default configuration file for the requested model will be used. That means that if
            :obj:`model` is given, its default configuration will be used. However, if :obj:`model` is not supplied,
            this :obj:`task`'s default model's config is used instead.
        tokenizer (:obj:`str` or :obj:`~transformers.PreTrainedTokenizer`, `optional`):
            The tokenizer that will be used by the pipeline to encode data for the model. This can be a model
            identifier or an actual pretrained tokenizer inheriting from :class:`~transformers.PreTrainedTokenizer`.

            If not provided, the default tokenizer for the given :obj:`model` will be loaded (if it is a string). If
            :obj:`model` is not specified or not a string, then the default tokenizer for :obj:`config` is loaded (if
            it is a string). However, if :obj:`config` is also not given or not a string, then the default tokenizer
            for the given :obj:`task` will be loaded.
        feature_extractor (:obj:`str` or :obj:`~transformers.PreTrainedFeatureExtractor`, `optional`):
            The feature extractor that will be used by the pipeline to encode data for the model. This can be a model
            identifier or an actual pretrained feature extractor inheriting from
            :class:`~transformers.PreTrainedFeatureExtractor`.

            Feature extractors are used for non-NLP models, such as Speech or Vision models as well as multi-modal
            models. Multi-modal models will also require a tokenizer to be passed.

            If not provided, the default feature extractor for the given :obj:`model` will be loaded (if it is a
            string). If :obj:`model` is not specified or not a string, then the default feature extractor for
            :obj:`config` is loaded (if it is a string). However, if :obj:`config` is also not given or not a string,
            then the default feature extractor for the given :obj:`task` will be loaded.
        framework (:obj:`str`, `optional`):
            The framework to use, either :obj:`"pt"` for PyTorch or :obj:`"tf"` for TensorFlow. The specified framework
            must be installed.

            If no framework is specified, will default to the one currently installed. If no framework is specified and
            both frameworks are installed, will default to the framework of the :obj:`model`, or to PyTorch if no model
            is provided.
        revision(:obj:`str`, `optional`, defaults to :obj:`"main"`):
            When passing a task name or a string model identifier: The specific model version to use. It can be a
            branch name, a tag name, or a commit id, since we use a git-based system for storing models and other
            artifacts on huggingface.co, so ``revision`` can be any identifier allowed by git.
        use_fast (:obj:`bool`, `optional`, defaults to :obj:`True`):
            Whether or not to use a Fast tokenizer if possible (a :class:`~transformers.PreTrainedTokenizerFast`).
        use_auth_token (:obj:`str` or `bool`, `optional`):
            The token to use as HTTP bearer authorization for remote files. If :obj:`True`, will use the token
            generated when running :obj:`transformers-cli login` (stored in :obj:`~/.huggingface`).
            revision(:obj:`str`, `optional`, defaults to :obj:`"main"`):
        model_kwargs:
            Additional dictionary of keyword arguments passed along to the model's :obj:`from_pretrained(...,
            **model_kwargs)` function.
        kwargs:
            Additional keyword arguments passed along to the specific pipeline init (see the documentation for the
            corresponding pipeline class for possible values).

    Returns:
        :class:`~transformers.Pipeline`: A suitable pipeline for the task.

    Examples::

        >>> from transformers import pipeline, AutoModelForTokenClassification, AutoTokenizer

        >>> # Sentiment analysis pipeline
        >>> pipeline('sentiment-analysis')

        >>> # Question answering pipeline, specifying the checkpoint identifier
        >>> pipeline('question-answering', model='distilbert-base-cased-distilled-squad', tokenizer='bert-base-cased')

        >>> # Named entity recognition pipeline, passing in a specific model and tokenizer
        >>> model = AutoModelForTokenClassification.from_pretrained("dbmdz/bert-large-cased-finetuned-conll03-english")
        >>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
        >>> pipeline('ner', model=model, tokenizer=tokenizer)
```

```
In [31]: p = pipeline('sentiment-analysis')

         p("I'm happy.")

No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english (https://huggingface.co/distilbert-b


Out[31]: [{'label': 'POSITIVE', 'score': 0.9998724460601807}]

In [32]: p("I'm sad.")

Out[32]: [{'label': 'NEGATIVE', 'score': 0.9994174242019653}]

In [33]: p("I'm not happy.")

Out[33]: [{'label': 'NEGATIVE', 'score': 0.9998021721839905}]

In [34]: import transformers

         p = pipeline('ner', aggregation_strategy=transformers.pipelines.AggregationStrategy.SIMPLE)

         p("""We would like to belatedly welcome Ulrich Glassmann of the Europa-Universität
           Flensburg (#EUF), who is currently a guest at the Cluster. Ulrich has just decided
           to extend his stay until the end of June, welcome news indeed!""")

No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll03-english (https://huggingface.co/dbmdz/bert-


Out[34]: [{'entity_group': 'PER',
           'score': 0.9996402,
           'word': 'Ulrich Glassmann',
           'start': 35,
           'end': 51},
          {'entity_group': 'ORG',
           'score': 0.8913957,
           'word': 'Europa - Universität Flensburg',
           'start': 59,
           'end': 89},
          {'entity_group': 'ORG',
           'score': 0.988505,
           'word': 'EUF',
           'start': 92,
           'end': 95},
          {'entity_group': 'ORG',
           'score': 0.6957305,
           'word': 'Cluster',
           'start': 130,
           'end': 137},
          {'entity_group': 'PER',
           'score': 0.9996954,
           'word': 'Ulrich',
           'start': 139,
           'end': 145}]
```

```
In [35]: p = pipeline('translation', model='facebook/wmt19-de-en')


         p("""Nicht nur unterschiedliche Berechnungen bereiten Kopfzerbrechen.
           Bei der Eigenwahrnehmung zeigt sich: In Deutschland gibt es massive
           Missverständnisse über Ausmaß und Art von Ungleichheit.""")

Out[35]: [{'translation_text': 'It is not only different calculations that cause headaches. Self-perception shows that i
```

```
In [36]: p = pipeline('translation', model='facebook/wmt19-en-de')


         p("""We would like to belatedly welcome Ulrich Glassmann of the Europa-Universität
           Flensburg (#EUF), who is currently a guest at the Cluster. Ulrich has just decided
           to extend his stay until the end of June, welcome news indeed!""")

Out[36]: [{'translation_text': 'Mit Verspätung begrüßen wir Ulrich Glassmann von der Europa-Universität Flensburg (# EUF
```

For text generation capabilities of transformers, see the demo page. Or the nice example models fine-tuned on tweets: https://huggingface.co/huggingtweets

Or run in the console:

```
p = pipeline('text-generation', model='distilgpt2')
p("In Germany there are massive misunderstandings about the extent and type of inequality.")
```