

# Data Mining 2022/2023

## Lab exercises

### 1 Python

I assume that you know the basics of Python. If you have only a little experience in Python, exercises in this section will guide you through the process of creating a convenient programming environment. At the end of this section you should be able to write and run Python code in Jupyter Notebook and also know basic Jupyter Notebook commands. You should also know how to install new packages and switch environments.

**Exercise 1** — Download and install [Anaconda](#). This is distribution of the Python and R programming languages for applications related to data science (see [wiki](#)). Read a quick [user guide](#) and [conda-cheatsheet](#) to learn how to create and switch environments and install new packages.

**Exercise 2** — Using Anaconda run Jupyter Notebook and see *Help* → *User Interface Tour, Keyboard Shortcuts*. Learn how to create, move and run cells. Learn how to get information about objects and methods and how to print their source code (e.g. can use *tab*, *shift+tab*, *shift+tab+tab* or write *'?* and *'??'* before a method name).

**Exercise 3** — Recall what [data structures](#) are available in Python. Pay special attention to list comprehensions as often they help to write more readable and more efficient code.

**Exercise 4** — Install and familiarize yourself with `numpy`, `pandas`, `matplotlib` packages.

- a) Read [official numpy user guide](#).
- b) Read [official pandas user guide](#).
- c) Read [official matplotlib user guide](#).

### 2 Word-count problem (deadline: 3rd lab)

**Exercise 5** — Find the source of your favorite book and save it in UTF-8 format. Load the book and split it into single words. For example you can use a construction like: (5p)

```
with open("Catch_22.txt", encoding="UTF-8") as f:
    words = [word
    for line in f
    for word in line.split() ]
```

Change all words to lower case, remove punctuation and remove stop-words. You may try constructions like:

```
from string import punctuation
words = [word.lower().translate(str.maketrans('', '', punctuation)) for
        ... ]

filtered_words = [w for w in words if not w in stopwords]
```

List of stop-words you can find in the Internet (e.g. [here](#)). You may also try using [stemming procedure](#) to reduce the different forms of a given word to a common form:

```
from stemming.porter2 import stem
filtered_words = [stem(word) for word in filtered_words]
```

Next, convert the obtained list of words into a list of pairs `(word, 1)` of the type `(String, Int)`:

```
pairs = [(w,1) for w in filtered_words]
```

Group the list of pairs by different words and count the total number of occurrences of each word to get pairs (word, occurrences). For example, you may use `groupby` method. However note that `groupby` method requires that input list is sorted by keys and such sorting might be computationally costly for large lists. Try to think up and implement a more efficient way of aggregating occurrences that does not require sorting.

```
from itertools import groupby
pairs.sort()
word = lambda pair : pair[0]
grouped_pairs = [(w, sum(1 for _ in g)) for w, g in groupby(pairs, key=word)]
```

Sort obtained list of pairs by the second component in decreasing order:

```
occurrences = lambda pair: pair[1]
grouped_pairs.sort(key=occurrences, reverse=True)
```

Remove some of the initial elements (most common words) and save the result to a text file. Build a word-cloud from the obtained list. You can use the service <http://www.wordclouds.com/>.



**Exercise 6** — This is the continuation of the previous task. (5p)

1. Divide your book into chapters. Treat each chapter as a document.
2. Split each documents into words (use lower case, stemming, etc.).

3. Determine the tf-idf weights of all words in all documents:

$$tf-idf(t, d, D) = tf(t, d) \times idf(t, D),$$

where  $t$  denotes a term (word),  $d$  denotes a document and  $D$  denotes the collection of all documents. Term frequency  $tf(t, d)$  is the number of times a term  $t$  appears in document  $d$ . Inverse document frequency  $idf(t, D)$  is often defined as

$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}.$$

There are packages that make it easy to find tf-idf weights, but try to implement the appropriate procedure yourself.

4. For each document separately build a word cloud using obtained tf-idf weights.

**Exercise 7** — Write a function that takes a word as an input and use tf-idf weights to create the list of chapters of your book most matching to that word (i.e. it should return a list of chapters sorted according to appropriate tf-idf weights). (5p)

**Exercise 8** — For each given word in your book make a list of five most common words that appear directly after considered word (but ignore stop-words). Use this summary to generate a random paragraph that resembles a paragraph of your book. (5p)

J.L.