Workshop

Python Programming for Linguists

Exercises

2020, Ingo Kleiber

> There are multiple ways you can approach these exercises. However, it is best if you actually try to write some code! You can do this on **Google Colab** (in any notebook or in an empty one, e.g., "playground") or in your **own development environment** (see Video "*Setting Up Your Development Environment*"). If you do not have the time or resources, I want to encourage you to think about these problems, even without writing out some code.
>
> Please be aware that some of these exercises are very challenging for beginners. Please do **not feel disheartened** by them! You can always look at the **provided solutions** and use them as a starting point for your own exploration.

Python for (Corpus) Linguists / Live Session

Exercise 8 – Concordancer
Write a basic concordancer that can generate concordances based on a given file and a given search term. If you want to challenge yourself, try to format the concordances in KWIC format.

Exercise 9 – N-Grams
Write a function that produces all n-grams based on a given text file and an *n.*

Hint: The NLTK provides a fairly easy solution to generating n-grams.

Exercise 10 – Frequency Analysis
Write a script that generates a frequency table for a given text. The list should contain all types and their frequencies.

Hint: Have a look at Python's Counter capabilities.

Exercise 11 – Computing Basic Statistics
Write a script that generates the following statistics for a given search term and a set of text files (a corpus):

- Absolute and relative frequencies
- The mean frequency
- The standard deviation

Also try to plot the frequency distribution across files.

Exercise 12 – NLTK Stemming, Lemmatization, and WordNet

Use `NLTK` to Stem and Lemmatize the following words. Use the `PorterStemmer`, the `LancasterStemmer`, and `WordNetLemmatizer` and compare your results. What are the pros and cons of these approaches?

```
words = ['connection', 'become', 'caring', 'are',
'women', 'driving']
```

Of course, feel free to add more examples!

Since you already have `WordNet`, try to find the synonyms for *fantastic* using WordNet.

Exercise 13 – spaCy Tagging

Use spaCy to automatically tag/annotate a text file of your choice for PoS, NERs, and Universal Dependencies.

Exercise 14 – Parsing XML

Write a function that allows you to extract all elements with a given attribute from an XML file.

For example, the function should should be able to produce the following output for the file `data/xml/bnc_style.xml` and the attribute `pos= "VERB"`: *have, bought*

Exercise 15 – Web Scraping

Write a function that scrapes the text from a given website. The function should take a URL as its input and return the text present on the given website.

If you want to challenge yourself even further, try to remove boilerplate (everything that is not the main text) from the result.

Exercise 16 – Putting Everything Together

The ultimate goal of this exercise is to write a system which can perform basic (comparative) keyword analysis on two corpora.

1. Use your web scraper to build a small Wikipedia corpus of about three to five articles. Ideally, they will belong to a similar topic, e.g., politics.
2. Find a suitable reference corpus to compare your Wikipedia corpus with.
3. Use your new skills to generate frequency lists for both corpora.
4. Implement any keyness statistic (e.g., log-likelihood) and determine the keywords

Hint: To download the COCA sampler, run the following command in a Google Colab cell:

```
!cd python-programming-for-linguists/2020/data && sh download_coca.sh
```

This will download and extract the COCA sampler to your `/data/corpora/coca` folder.