Workshop

Python Programming for Linguists

Exercises

202, 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

Before you start working on these exercises, have a look at the "**03 – New Tools and Syntax**" notebook, which introduces some very helpful new concepts.

I have also created an **opinionated notebook template** for you as a starting point! Of course, you don't have to use it when trying to solve these exercises. However, this will be the **starting point for the live session**.

Also, you can always have a look at the **2020 results** (video, notebook) if you get stuck. Most exercises have stayed the same! 😀

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 – Basic Collocation Analysis

Write a function that allows you to find collocates of a given word in a given text file. If you want to do this from scratch, you will have to implement a collocation/association measure of your choice.

Hint: Similarly to the n-gram exercise, the `NLTK` provides a fairly easy solution to generating collocations.

Exercise 13 – NLTK Stemming, Lemmatization, and WordNet

Use `NLTK` to stem and lemmatize the following words. Use the `PorterStemmer`, the `LancasterStemmer`, and the `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 14 – spaCy Tagging

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

Exercise 15 – 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 be able to produce the following output for the file `data/xml/bnc_style.xml` and the attribute `pos="VERB"`: *have, bought*

Exercise 16 – Web Scraping

Write a function that scrapes the text from a given website (e.g., Wikipedia). 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 17 – Putting Everything Together (Keyword Analysis)

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., simple maths or 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.