Programming for LT Applications

Artur Kulmizev

February 15, 2021

This assignment package will be rolled out in three parts, each corresponding to a lab section: **Web Scraping**, **Pre-processing and Visualization**, and **Modular Programming**. This document will be updated with each part accordingly, which will have separate exercises associated with it. The goal of this assignment is to get to you to write practical applications in Python, using the concepts we covered in the lectures. You should hand in a single tarball[1], named according to the following format: `assignment4.tar.gz`. The directory structure of the tarball should be the following:

```
A3_0_assignment_3
├── twitter_query
│   ├── README.txt
│   └── twitter_query.py
├── wiki_query
│   ├── README.txt
│   └── wiki_query.py
├── term_plotter
│   ├── README.txt
│   └── term_plotter.py
└── tagger
    ├── tagger
    │   ├── __init__.py
    │   ├── pre_process.py
    │   └── model.py
    ├── README.txt
    ├── run.py
    └── config.yaml
```

In other words, the examiner should easily be able to navigate to a particular exercise folder, and call the exercise script directly via e.g.

```
$ python script.py ARG1 ARG2
```

This code should run out of the box[2] and produce the desired results. You can approach the problems in any way that you want (using classes, functions, etc.), but the code has to be executable from the corresponding exercise script file and return the desired output. You should be also mindful in making your code clean and interpretable, commenting where necessary. The `PEP 8` style guide[3] can give you some tips in this regard, but you are by no means required to use it. Each exercise folder should also include a `README.txt` file that explains how to run your code, as well as how you went about writing it (e.g. did you make a class?).

[1] https://www.howtogeek.com/362203/what-is-a-tar.gz-file-and-how-do-i-open-it/

[2] We shouldn't get errors when running a simple query.

[3] https://www.python.org/dev/peps/pep-0008/

Lab 1: Web Scraping

Finding good data is a crucial first step in many NLP applications. When ready-made corpora are not available, we often revert to scraping relevant texts from the internet. In the web scraping lecture, we learned a bit about web scraping: how websites are built upon HTML files, how we can parse these files to extract information, among other things. If you recall, we discussed two distinct scenarios that might arise when scraping the web. In the first scenario, you might have access to a website's API, which you can use to interact with the website directly. Otherwise, you might have to do the hard work yourself: understand the structure of the website data and parse it accordingly. We'll explore these situations in this lab, where you'll build two tools: one for extracting Tweets for a specified user on Twitter, and another for extracting the descriptions of concepts from Wikipedia.

Extracting Tweets

Your task: Write a program that takes a Twitter user's handle as an argument and returns the user's most recent Tweet(s), as well as the time of their posting. Your script should also accept an optional integer N, which, if specified, returns the user's N-most recent Tweets. If not specified, it should return only the single most recent Tweet. For example:

```
$ python3 twitter_query.py jack 3
11:28 PM, Feb 21, 2020: I've discovered the way to make slack grind
to a crawl and crash repeatedly
3:29 AM,  Feb 21, 2020: Talked #bitcoin Africa with #NiiQuaynor,
@PindarWong, and @moneyball! Earth globe europe-africa
11:25 PM, Feb 20, 2020: I just talked with @leslieberland's mom
on the phone AMA
```

You should work with the `Tweepy`[4] library for navigating the Twitter API. In order to do so, you will need a set of credentials: a consumer key and secret, and an access token and access token secret. These can be accessed via a JSON file found at `/local/kurs/advprog/module3/credentials.json`. You can decide how to handle replies, retweets, images, emoji, etc., but make sure to document your decision in doing so. Your script should be called `twitter_query.py`.

Mining Wikipedia

Your task: Write a program that takes a search term as an argument and then prints the content of the Wikipedia page for that term. For example:

```
$ python3 wiki_query.py ankylosaurus
Ankylosaurus is a genus of armored dinosaur. Its fossils have been
found in geological formations dating to the very end of the Cretaceous
Period, about 68-66 million years ago, in western North America,
making it among the last of the non-avian dinosaurs. It was named by
Barnum Brown in 1908, ...}
```

Make sure that your program uses at least one `try-except` clause (you can choose which exceptions to try to catch). Also ensure that you can search for terms with non-`ASCII` characters. For example, check that your program works with the search term

[4] http://docs.tweepy.org/en/latest/index.html

"Miloš_Zeman". You might want to use the libraries `urllib` and `beautifulsoup` (see the lecture notebook and online documentation), but feel free to approach the problem however you want. Your script should be called `wiki_query.py`.