MP3 Write-Up and Reflection

Nabih Estefan - March 2020

The Wikipedia Philosophy Chain

Project Overview

There is a very interesting fact about Wikipedia that many people do no know about. If you were to click on the first link on any Wikipedia page (excluding links to notes on the bottom of the page) and then keep doing this for every consequent Wikipedia link, you will inevitably reach the page for Philosophy. Knowing this, I hoped to create a program that would receive a Wikipedia page from the user and return a file (which it also printed) that showed the user the steps taken, from that initial page, to reach the Philosophy page, and how many steps it took. The easiest way to do this was reading the `html` code for the page, although easy might be an overstatement, because (as you might expect from an open-source thing) the `html` code is not completely standardized.

Implementation

There were many ways I could have decided to build my project. The one I chose to do is a recursive search through the link until the function reaches a base case, the Philosophy Wikipedia Page, the end of what I called *The Philosophy Chain*. The major component of the recursive case is the calling of the function `findFirst()` which is destined to find the first link in any given Wikipedia link. This function is where I had to decide the biggest part of how my code was going to work. I either had to choose to use the `BeautifulSoup` or the `Wikipedia` imports to look through the web page. I eventually decided to use `BeautifulSoup` because it gave me more control over the `html` while the `Wikipedia` one was more general and only gave me more complications.

After this I had complete control of the `html` and was able to find the most common course to reach the first link, and the places I had to loop through to reach it. I had to check that it was in the first paragraph and that the classes and types of text were the appropriate ones, and then return it. But after returning it, I ran into my second biggest choice, the one I doubted the most and the one I went back and forth about: *Do I store this path? If I do, how do I store it?* The way I ended up solving this problem was by having a global variable called steps that recorded every link used throughout the path. This worked, but I could still make it better. I decided that, if I had already found the path once, there was no reason to find it again, so I decided to, instead of saving the paths to a list that was lost when the program finished, I would save them to a file which would always be there, which I finally did. I doubled back on this decision when I tried to solve links that looped though certain paths, but the complication for this was too high, and I ended up ditching it and going back to files.

Results
I was, after many attempts and workarounds, to finish my program. It is a very interesting program called with a single line:

```
python3 text_mining.py "name"
```

where name is `"Apple Inc"` or `"Latin"` or any other Wikipedia webpage you want to search. This command will run for a while and then will create a file and prints its contents. The file will be names `Philosophy Chain name` (again with name being the Wikipedia page where the recursive call will start) and will include three parts.

The first and last parts of the file are base lines that say "Starting at `link` these are the steps to reach the Wikipedia Philosophy Webpage" (with link being the starting link) and the last one is "These were the `x` steps to reach the Wikipedia Philosophy Webpage" (with x being the number of times it had to "click" on the first link to reach the base case). The middle part is the part that defers and it is the list of links followed by the program.

Here are a couple of examples from the text outputted by the program:

```
Starting at https://en.wikipedia.org/wiki/Water, these are the steps
to reach the Wikipedia Philosophy Webpage
 https://en.wikipedia.org/wiki/Water
 https://en.wikipedia.org/wiki/Inorganic_compound
 https://en.wikipedia.org/wiki/Chemical_compound
 https://en.wikipedia.org/wiki/Chemical_substance
 https://en.wikipedia.org/wiki/Matter
 https://en.wikipedia.org/wiki/Classical_physics
 https://en.wikipedia.org/wiki/Physics
 https://en.wikipedia.org/wiki/Ancient_Greek_language
 https://en.wikipedia.org/wiki/Greek_language
 https://en.wikipedia.org/wiki/Modern_Greek
 https://en.wikipedia.org/wiki/Dialect
 https://en.wikipedia.org/wiki/Latin
 https://en.wikipedia.org/wiki/Help:IPA/Latin
 https://en.wikipedia.org/wiki/International_Phonetic_Alphabet
 https://en.wikipedia.org/wiki/Alphabet
 https://en.wikipedia.org/wiki/Letter_(alphabet)
 https://en.wikipedia.org/wiki/Symbol
 https://en.wikipedia.org/wiki/Idea
 https://en.wikipedia.org/wiki/Philosophy
These were the 19 steps to reach the Wikipedia Philosophy Webpage
```

Another example:

```
Starting at https://en.wikipedia.org/wiki/Post-it Note, these are the
steps to reach the Wikipedia Philosophy Webpage
 https://en.wikipedia.org/wiki/Post-it Note
 https://en.wikipedia.org/wiki/Adhesion
 https://en.wikipedia.org/wiki/Interface_(matter)
 https://en.wikipedia.org/wiki/Outline_of_physical_science
 https://en.wikipedia.org/wiki/Natural_science
 https://en.wikipedia.org/wiki/Branch_of_science
 https://en.wikipedia.org/wiki/Science
 https://en.wikipedia.org/wiki/Latin
 https://en.wikipedia.org/wiki/Help:IPA/Latin
 https://en.wikipedia.org/wiki/International_Phonetic_Alphabet
 https://en.wikipedia.org/wiki/Alphabet
 https://en.wikipedia.org/wiki/Letter_(alphabet)
 https://en.wikipedia.org/wiki/Symbol
 https://en.wikipedia.org/wiki/Idea
 https://en.wikipedia.org/wiki/Philosophy
These were the 15 steps to reach the Wikipedia Philosophy Webpage
```

Alignment

One of the questions that sparked my interest when I heard about *The Wikipedia Philosophy Chain* was why it happened. Did all links eventually get to Philosophy because of some special reason, or was it just pure luck it happened? Well, after playing around for a while with this program (and getting banned for using the Wikipedia `html` code for a weekend.) I have concluded that it is not pure luck, it is, in fact, caused by an internal Wikipedia agreement on pronunciations. Yes, pronunciations lead to the International Phonetic Alphabet (IPA), which always ends up leading to Philosophy.

Of the more general questions for Mini-Project 3, I decided that I wanted to use recursion in text processing, and find patterns that occur in different web pages and their text. I ended up doing this very nicely, I started by choosing `html` which was very useful in helping me organize the web pages, and then used recursion and loops to find the patterns that I wanted, which was very nice.

On the other hand, I feel that the tools at my disposal, added with the unruliness of Wikipedia `html` limited the results I could get. I really wanted to be able to find a way to tell the program that, if the first link of a page has already been used (i.e. its going into a loop) then it should look for the next link. This was almost impossible using `BeautifulSoup` and `html`. On the same topic, I tried to find a way to go around the IPA route, and seek another route, but eventually they all ended looping (or going to links way down in the page) and the tools I had made ti very difficult for me to go around this problem.

Reflection

In my opinion, my project went very well, I was able to find *The Philosophy Chain* in most Wikipedia web pages I wanted to, and was able to find patterns and reasons why this exists. Personally, I feel that, initially I thought my project was going to be too easy, but as time went on, I found details that made every case harder, to eventually get my project to the difficulty level it reached, where the tools at my disposal don't let me complete it appropriately. Looking back, I wish I knew the depth this would go to, so I would be able to look into other tools for `html` parsing and find a more suitable one for my work.