BIA660
Extra Credit LinkedIn Scraper

Assignment Description

This is an optional assignment for extra credit. It is an individual assignment (no teams).

- Create a python script to scrape the html version of 2000 Linkedin profiles. Each html should be saved in a separate file (so 2000 html files). IF YOU WANT TO WORK ON THIS ASSIGNMENT, EMAIL ME FIRST to express your interest. I will then reply with instructions on which Linkedin profiles to target.

- Create a second python script to parse the html of all 2000 profiles and create a single csv file that includes 1 line per person (so 2000 lines total). Each line should include the names of the companies that the person has worked for during their entire career (separated by commas).

Deliverables

-The deliverable is a zip file that includes:

The scraping script
The parsing script
A folder with the 2000 html files.
The csv file.

How to run:

First, edit the main() method on part1_loginandscrape.py to include your username, password and the link for the search you would like to scrape (in my case, one of the URLs was "https://www.linkedin.com/search/results/people/?keywords=intel&origin=SWITCH_SEARCH_VERTICAL")
Second, run part2_parseandwrite.py.
The code will output 2 lists saved as pickles, and a company_history.csv file.

My approach

Following the description, I created two files, one for obtaining the HTML files for each profile, and another for parsing it to obtain the information we need. Below I go into detail on the content of each file.

part1_loginandstore.py

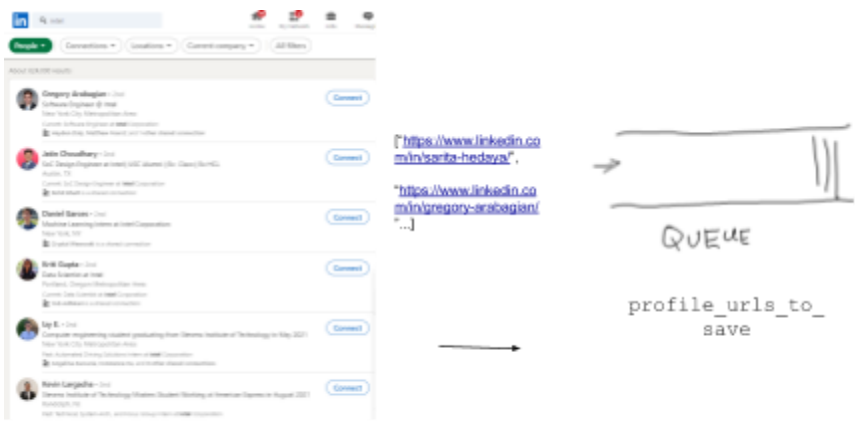
This file performs the following tasks:

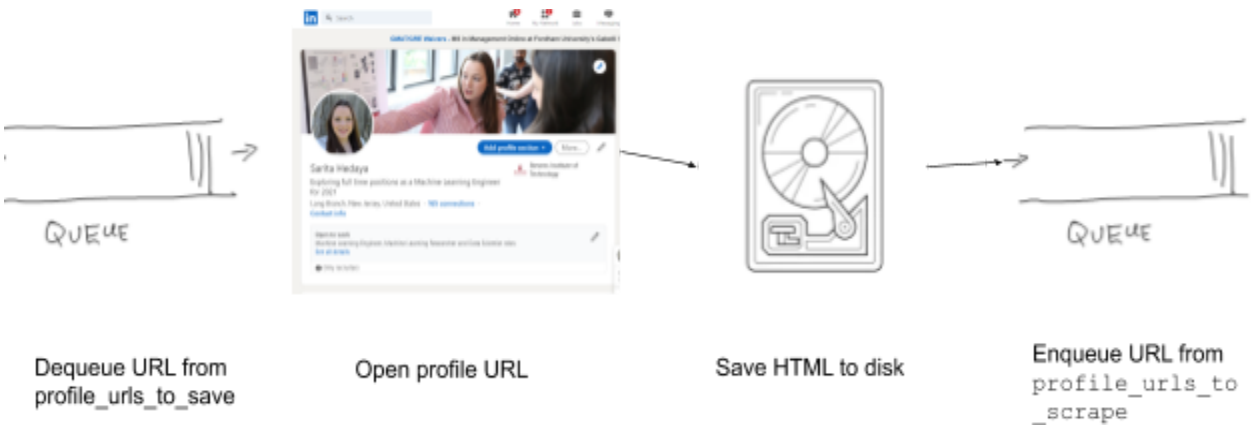
- Use selenium browser to open and login to linkedin.com
- Navigates to the search that we are interested in. In this case, people who worked for Intel at some point in their careers.
- Saves the html for the profiles to disk.

Implementation Details:

The implementation consists of keeping two queues: one for the URLs of the profiles to scrape, and another for those profiles that we already stored HTML files, but haven't yet parsed.

1. open the search results and scrape the URLs of the profiles for all of the pages on the range we selected when running main()
2. Store the URLs as strings in a list. We pickle that list and store in disk in case the code crashes, we don't have to repeat work.
3. One by one, dequeue the URLs from the queue, use Selenium to open the profile, and store the HTML to disk.
4. Enqueue the already saved URL string to another queue, for the profiles that we haven't scraped for employment history.





The actual parsing for employment history happens in the file for part2. The benefits of separating into two files in this way is that the second part won't have to enter the web at all, as it can work directly from the saved HTMLs. Therefore, I can run the html scraper during the time that I'm blocked out of LinkeinIn.com if that ever happens.

part2_scrapeandwrite.py

This code loads the HTML files one by one, parses for the company names we are interested, and saves them to csv. Specifically:

1. iterates through every html file I saved on the ./profiles directory, and opens them one by one.
2. Then it looks for the string "companyName". The first 7 appearances of that string (including quotes) are to be discarded. Starting from the 8th appearance, we look at the text within quotes that appears immediately after "companyName".
3. I save the company names into a set to avoid writing a company twice when the profile contains a promotion or a change of title within the same company (since we're only interested in naming the companies the person worked for).
4. I use csv writer to write the elements of the set to a csv.
5. Lastly I move the HTML file to another folder. I do this until I have parsed all of the HTMLs in the ./profiles directory.

Challenges with this assignment

The main issue encountered in this assignment is that when I used Chrome's "Inspect" tool like we learned in class, I easily found the tag containing the text I'm looking for. But when I look at the source code, the tag does not exist. Instead, there are some <code> and <script> tags containing JSON with the information mixed in. I tried the following approaches to extracting the company names:

1. Use a regular expression to find the text “CompanyName” and use whatever comes next that is between quotes. For example:

   ```
   "multiLocaleCompanyName":{"en_US":"Brigham and Women's
   Hospital"},"companyName":"Brigham and Women's
   Hospital","*company":
   ```

   Would return Brigham and Women's Hospital

2. Use a JSON parser like slimit. I've never worked with JSON before, so this was a learning experience.

Because of my limited experience with both JSON and regular expressions, I spent days trying both approaches to no avail.

Then I thought of using simple python string.find() to perform the searching I describe in the previous page, and I was surprised to find it was very fast even though it was searching into long HTMLs.

A second but less major challenge was that linkedin only has 100 pages of search results, and some of those results are not visible, so we are getting less than 1000 results for a search. I had to perform searches for people with the keyword “Intel”, as well as people who worked for “Intel Corporation”, “Intel Labs”, “Intel Internet of Things”, etc. while avoiding repeats.