Web Scraping

Dr. Garrett Dancik

Overview

- Web scraping is the process of retrieving web pages and extracting relevant data from them
- Why (examples)?
  - Search engines collect data to index web pages
  - Collecting weather and climate data for research
    - https://www.sciencedirect.com/science/article/pii/S0168169909002348
  - For businesses and consumers to keep track of products
  - For research in economics
    - https://www.aeaweb.org/articles?id=10.1257/jep.30.2.151
  - Understanding the housing / rental market
    - https://journals.sagepub.com/doi/full/10.1177/0739456X16664789
  - To create a facial recognition database of > 3 billion images scraped from Facebook, YouTube, Venmo, etc
    - https://www.nytimes.com/2020/01/18/technology/clearview-privacy-facial-recognition.html

Legal / ethical considerations

- Legally, web scraping can be a gray area, but you should
  - Respect each website's terms of service
  - Respect each site's *robots.txt* file
    - A web site's robots.txt file lets robots (crawlers and web scrapers) know what behavior is permissible and what is not
    - https://www.promptcloud.com/blog/how-to-read-and-respect-robots-file
    - Eastern does not have one: http://www.easternct.edu/robots.txt
    - From Travelocity: https://www.travelocity.com/robots.txt
  - Use a web site's Application Programming Interface (API) if available.
  - Do not overload the web server of the page you are visiting. Have your scraper pause/sleep if making multiple requests
  - Identify yourself in the HTTP request header

Steps for web scraping

- Identify a page you want to scrape
- Understand the structure of the page (e.g., by using the Web Inspector)
- Write a script that does the following:
  - Retrieve a web page
    - From a file using *open*
    - Use the Python *requests* module to retrieve a page from its URL
  - Extract information from the web page
    - Using Python's *BeautifulSoup* library