DSCI 558: Building Knowledge Graphs

Homework 1: Crawling

Released: Aug 28th, 2020

Due: Sep 6th, 2020 @ 11:59pm

Ground Rules

This homework must be done individually. You can ask others for help with the tools, however, the submitted homework has to be your own work.

Summary

In this homework, you will implement a web crawler to collect webpages and extract data from the Internet Movie Database (IMDb) website (https://www.imdb.com/).
A web crawler is a program/bot that systematically browses the World Wide Web (WWW), typically for the purpose of web indexing (web spidering). It starts with a list of seed URLs to visit, and as it visits each webpage, it finds the links in that web page, and then visits those links and repeats the entire process. You are required to use Scrapy (https://scrapy.org), a free and open-source web-crawling python library.

Task 1 (4 points)

Crawl at least **5000** webpages of **Comedy movies/shows** in IMDb using Scrapy. Extract and generate the following attributes for each webpage:

`id`	unique id for the webpage
`url`	url of the webpage
`timestamp_crawl`	timestamp of the crawling event
`title`	see Figure 1 if attribute doesn't exist, set as empty value (string or list. According to type)
`genres`*	
`languages`*	
`release_date`	
`budget`	
`gross`	
`runtime`	

`*` attribute holds a list of strings, not a single string value

Store your crawled data into a JSON-Lines (`.jl`) file. In this file format, each line is a valid JSON object (dictionary) that holds the attributes listed above for a single crawled webpage. You can check the attached file `sample.jl` to understand the format (note that `genres` and `languages` are a list of values). While crawling, please make sure you obey the website's politeness rules (i.e. sleep time between requests) in order to avoid getting banned.

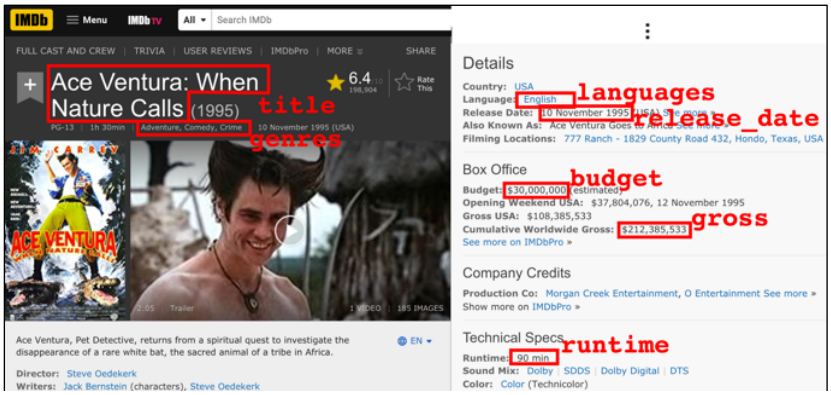

Figure 1: An example movie/show webpage with the required attributes

Task 2 (4 points)

Similar to the previous task, crawl at least **5000** webpages of cast (actors and actresses) in IMDb using Scrapy. Extract and generate the following attributes for each cast webpage:

`id`	unique id for the webpage
`url`	url of the webpage
`timestamp_crawl`	timestamp of the crawling event
`name`	see Figure 2 if attribute doesn't exist, set as empty string
`date_of_birth`	
`place_of_birth`	
`date_of_death`	
`place_of_death`	
`mini_bio`	

Similarly, store your crawled data into a JSON-Lines (`.jl`) file.

Figure 2: An example cast webpage with the required attributes

Task 3 (2 points)

Answer the following questions (no more than 2 sentences for each question) **for each** of the previous tasks (Tasks 1 and 2):

3.1. What is the seed URL(s) you used?
3.2. How did you manage to only collect movie/show or cast pages?
3.3. Did you need to discard irrelevant pages? If so, how?
3.4. Did you collect the required number of pages? If you were not able to do so, please describe and explain your issues.

Submission Instructions

You must submit (via Blackboard) the following files/folders in a single `.zip` archive named `Firstname_Lastname_hw01.zip`:

- `Firstname_Lastname_hw01_report.pdf`: pdf file with your answers to Task 3
- JSON-Lines files containing the data you crawled using Scrapy for Tasks 1 and 2:
  - `Firstname_Lastname_hw01_scrapy_title.jl`: Generated data from Task 1
  - `Firstname_Lastname_hw01_scrapy_cast.jl`: Generated data from Task 2
- `source`: This folder includes all the code you wrote to accomplish Tasks 1 and 2 (i.e. your Scrapy crawler, seed files, your script/program to eliminate unwanted pages and store webpages into JSON-Lines format, etc…)

Additional Notes

1. We also provide you with a script called `post_processing.py`, you can use this script to validate the structure of your JSON-Lines files in Task 1:
   a. Prerequisites: python 3 and the `ujson` package (`pip install <package>`)
   b. Usage: The script takes one argument `<path>` which is the path of your `jl` file, processes the file and prints a message to let the user know if it is valid.
   `python post_processing.py /path/to/file/sample.jl` will print:
   ```
   Process: (0/2)...
   Process: (1/2)...
   Finished processing, looks good, found 2 entries.
   ```
2. It is your responsibility to validate the structure of the outputs in task 2 prior to submitting.