Assignment 2.1

由 Wang, Ren-Jay创建, 最终由 Kim, Yongjin修改于 十月 08, 2018

Assignment 2.1 - Extending your web scraper

Overview

This week, you will be expanding on the data you scraped from last week to include several important new features.

> ⚠ **Programming Language**
> Continue working in the same language that you used last week, unless your moderator last week told you to switch languages. The one exception is for data visualization - see below for more details.

Being the superstar senior software engineer that you are, you have decided that although your work last week was impeccable, there are still some features you can add to make it more presentable. Specifically, the new requirements you would like to add are:

1. Analysis - you want to be able to answer some meaningful questions about your data
2. API Creation - you want the public to have access to your data
3. Visualization - you want your data to be understandable via some graphs and charts (Extra Credit)

Read the sections below for more detail!

> ⚠ **Non-Functional Web Scraper**
> If you were not able complete the web scraping from last week, you may use the data file in Part 0 (data.json). We have done our best to make this dataset as clean as possible; however, if you choose to use this data, it is up to you to work around any missing data or formatting issues you encounter. You will also need to compute the edges and their weights yourself.

Part 0 : External JSON support

We have provided a test JSON file, which stores the relevant data for actors and movies, but not for the edges. Here is the data file: data.json. Your job is to be able to parse this JSON file into your graph structure into both vertices and edges and be able to use it for each of the following 2 parts. This will allow us to test your code in section.

> ⚠ **Copying Code**
> Remember that you **must** cite any code snippets that you copy (from books, StackOverflow, etc). Remember, at least 80% of the code you turn in must be your own code.

Part I : Data Analysis

You have a client! Write code to help him answer the following questions. **Be sure to include graphs/charts/scatterplots along with the code you write to support your answer.**

- Who are the "hub" actors in your dataset? That is, which actors have the most connections with other actors? Two actors have a connection if they have acted in the same movie together.
- Is there an age group that generates the most amount of money? What does the correlation between age and grossing value look like?

You are also encouraged to perform your own analysis on your data, and may receive bonus points for interesting and/or well presented analysis. Note that you should be using the programming language you used last week for this part of the

assignment.

Part II : API Creation

You want to make your amazing data publicly accessible. To do this, you will write an API.

What is an API?

> In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it's a set of clearly defined methods of communication between various software components...Just as a graphical user interface makes it easier for people to use programs, application programming interfaces make it easier for developers to use certain technologies in building applications. By abstracting the underlying implementation and only exposing objects or actions the developer needs, an API reduces the cognitive load on a programmer. - Wikipedia, Application programming interface

APIs are used in many real world applications, and are common in software development, web development, personal projects, hacakthons and many other places. In this assignment, we are asking you to write a **web API**. In addition to making your data accessible, this will help give you a better understanding of how web structuring works.

Your API should include the following endpoints: GET, PUT, POST and DELETE.

- Get:
  - /actors?attr={attr_value} Example: /actors?name=”Bob”
    - Filters out all actors that don’t have “Bob” in their name (Should allow for similar filtering for any other attribute)
  - /movies?attr={attr_value} Example: /movies?name=”Shawshank&Redemption”
    - Filters out all actors that don’t have “Shawshank&Redemption” in their name (Should allow for similar filtering for any other attribute)
  - /actors/:{actor_name} Example: /actors/Bruce_Willis
    - Returns the first Actor object that has name “Bruce Willis” displays actor attributes and metadata
  - /movies/:{movie_name} Example: /movies/Shawshank_Redemption
    - Returns the first Movie object that has correct name, displays movie attributes and metadata
  - You should also be able to filter using boolean operators AND and OR, i.e. name=”Bob”|name=”Matt”, name=”Bob”&age=35
- Put:
  - /actors
    - Leverage PUT requests to update standing content in backend
    - curl -i -X PUT -H "Content-Type: application/json" -d ' {"total_gross":500}'http://localhost:4567/api/a/actors/Bruce_Willis
  - /movies
    - Leverage PUT requests to update standing content in backend
    - curl -i -X PUT -H "Content-Type: application/json" -d ' {"box_office":500}' http://localhost:4567/api/m/movies/Shawshank_Redemption
- Post:
  - /actors
    - Leverage POST requests to ADD content to backend
    - curl -i -X POST -H "Content-Type: application/json" -d'{"name":"Billy Joe"}' {API URL}/actors
  - /movies
    - Leverage POST requests to ADD content to backend
    - curl -i -X POST -H "Content-Type: application/json" -d'{"name":"Captain America"}' {API URL}/movies
- Delete:
  - /actors/:{actor_name}
    - Leverage DELETE requests to REMOVE content from backend
    - curl -i -X DELETE -H "Content-Type: application/json" {API URL}/actors/Bruce_Willis
  - /movies/:{movie_name}
    - Leverage DELETE requests to REMOVE content from backend
    - curl -i -X DELETE -H "Content-Type: application/json" {API URL}/movies/Shawshank_Redemption
- Your API should return a HTTP Code for each request. At a minimum, you should include 200 OK, 201 Created and 400 Bad Request.

Your output to all endpoints should be in JSON format. You are **not required** to put your data on the internet. You can test your API on localhost using POSTMAN.

Here are some recommended resources to get you started:

- Python: Beginner - Flask, Designing an API with flask, Advanced - Django
- Javascript: Node.js, Designing an API with Node.js
- Ruby: Sinatra, Designing an API with Sinatra

Again, note that you should be using the programming language you used last week for this part of the assignment.

Part III: Data Visualization (Extra Credit)

We would like you to visualize the movie/actor data you scraped from last week. The overall look of this is up to you, but the visualization should **at a minimum:**

- Show each actor and movie clearly
- Show the connections between each movie and actor clearly
- Show the age of each actor and the total grossing value of each movie clearly or on demand
- Be readable

Since there are a lot of vertices in your graph, you may choose to visualize a portion of your graph rather than all of it.

To help you get started, here are some recommended libraries; however, you are free to use whichever libraries you would like as long as it does not trivialize the assignment:

- Python: matplotlib, networkx, graph-tool (available for GNU/Linux, MacOS X), igraph, Plotly (online tool, requires API key)
- Javascript: visjs

We realize that Ruby does not have well documented graph visualization support, so you may use any language you want for graph visualization this week. You can use your function from last week to convert your data into a JSON file, and then write a visualization class in the language of your choice that accepts a JSON file.

As mentioned above, if you were not able to scrape data from last week, you may use this data for your visualization.

Testing

As usual, we require that you write extensive unit tests for each part of this assignment. Since it is difficult to write unit tests for your visualization, you should submit a manual test plan with relevant screenshots (Extra Credit). For your API, if the tools you are using support unit testing we expect you to use them. For example,

- Python: Flask, Django
- Javascript: Mocha and Chai
- Ruby: minitest

If you cannot find an appropriate unit testing support for the language you are using, you should contact your moderator who may approve a manual test plan.

无标签

Summary

Table of Contents

Reading

[On the Criteria To Be Used in Decomposing Systems into Modules]

Submission

This assignment is due at the **beginning of your discussion section the week of October 15, 2018**. Please be sure to submit in Gitlab, and ask your moderator or TA before the deadline if you have any questions.

Objectives

- Learn about data visualization
- Practice data cleaning/preparation
- Practice basic data analysis techniques and algorithms
- Learn how to create and test a web API

Resources

- Python
- Ruby
- Javascript
- JSON Homepage
- JSON Wikipedia
- Graphs

Grading

Category	Weight	Scoring	Requirement
Basic Preparation	2	0-1	Ready to go at the start of section
Cleverness	2	0-2	The hardest points on the rubric
Code Submission	4	0-2	Submitted correct content on time and to the correct location in the repository
Decomposition	4	0-2	Project is adequately decomposed to different classes and methods
Documentation	4	0-2	Comments for each class and each function are clear and are following style guides
Effort	2	0-2	Perform considerable amount of work
Naming	2	0-2	Variable names and method names are readable and are following the naming conventions of the language you chose
Overall Design	5	0-2.5	Have nice approaches and structures in overall
Participation	5	0-2.5	Interact with the group 2 times (ask a question, make a comment, help answer a question, etc.)
Presentation	4	0-2	Present the code clearly
Requirements - Data Analysis	5	0-2.5	• 0 points: Lacks any Data Analysis • 1 point: Provides some form of Data Analysis over provided questions; lacks support through code or graphs/charts/scatterplots; or uses a different language from 2.0 for code not related to visualization • 2 points: Answers all provided questions and provides support through code or graphs/charts/scatterplots • 2.5 points: Adds on extra analysis that provides valuable insight into the data beyond what is required, or provides well presented analysis.

Requirements - API Creation	5	0-2.5	• 0 points: Lacks any form of API • 1 point: Covers some of the required endpoints, but not all required functionality or does not output to JSON format; Implements API in a different language from 2.0 • 2 points: Covers all required endpoints, outputs to JSON format • 2.5 points: Extends API to support more endpoints than the required ones, outputs to JSON format.
Requirements - External JSON support	3	0-1	• 0 points: Lacks support for importing provided JSON • 1 point: Support importing provided JSON and calculating any information necessary for internal data structures as provided JSON is as minimal as possible and does not contain weights on edges.
Testing - API & Data analysis	5	0-2.5	• 0 points: No unit tests covering API • 1 point: Sub 80% code coverage or unit tests fail to cover all API related functionality or Data Analysis code • 2 points: 80%+ code coverage on all API related code and Data Analysis code, API unit tests use a smaller test JSON • 2.5 points: 95% code coverage on all API related code and Data Analysis code, API unit tests use a smaller test JSON
Live Testing - Data analysis	2	0-1	• 0 points: Incorrect or missing answers for analysis questions for the test JSON file. • 1 point: Provide correct answers to analysis questions for the test JSON file.

Requirements - Data Visualization (Extra Credit)	4	0-2	• 0 points: Lacks any Data Visualization • 1 point: Provides some form of Data Visualization, but does not meet the minimum requirements • 2 points: Provides enough Data Visualization to meet the minimum requirements • 2.5 points: Provides well-designed Data Visualization and manual test plan.
Total	53		