Python for Informatics

LESSON 8

Visualizing Data

- With this lesson, we will look at three complete Python applications that integrate the technologies we have learned so far.
- The technologies we will integrate are:
  - *The Core Python Language*
  - *Network Programming*
  - *Database Programming*

Visualizing Data

- The ***geoload.py*** application acquires ***Google geocoding*** information to process the geographic locations of user-entered university names, which are then placed upon a ***Google map***.
- This application can be downloaded from here:

  www.py4inf.com/code/geodata.zip

Visualizing Data

- The free ***Google geocoding web service*** is rate-limited.
- You are allowed to make only 2500 requests per day.
- To work within the imposed limitations, you may need to start and stop your application, thereby gathering the data together incrementally.

Visualizing Data

- Our user-entered "survey" data comes to us in the form of the ***where.data*** file.

- The ***where.data*** file is a text file containing a series of university names, with one university name per line.

Visualizing Data

Northeastern University
University of Hong Kong,
Illinois Institute of Technology,
Bradley University
Technion
Viswakarma Institute, Pune, India
UMD
Tufts University
Monash University
.
.
.

Visualizing Data

- After reading each university name line by line, we will retrieve the ***Google geocode*** information, and then store that information in ***geodata.sqlite***.
- Before we ask the ***geocode web service*** for any location information, we first check to see if we already have that information stored in our ***geodata.sqlite database***.

Visualizing Data

- In this way, our ***geodata.sqlite database*** serves as a local cache, ensuring that we don't ask for data that we already have.
- If we ever want to start over from scratch, we can just delete the ***geodata.sqlite database***, and re-run the ***geoload.py*** program.

Visualizing Data

- More succinctly, this is what the ***geoload.py*** program does:
  - Read input line from the ***where.data*** file.
  - For each line, check to see if it is already in the ***geodata.sqlite database***.
  - If it is not in the ***database***, call the ***geocode web service*** to retrieve the data, and then store it in the ***database***.

Visualizing Data

Found in database Northeastern University

Found in database University of Hong Kong

Found in database Illinois Institute of Technology

.

.

.

Resolving Kokshetau Institute of Economics and Management

Retrieving http://maps.googleapis.com/maps/api/geocode/json?sensor=false&address=Kokshetau+Institute+of+Economics+and+Management

Retrieved 1956 characters {   "results" : [

{u'status': u'OK', u'results': [{u'geometry': {u'location_type': u'APPROXIMATE', u'bounds': {u'northeast': {u'lat': 53.3444028, u'lng': 69.4638061}, u'southwest': {u'lat': 53.2533834, u'lng': 69.35711859999999}}, u'viewport': {u'northeast': {u'lat': 53.3444028, u'lng': 69.4638061}, u'southwest': {u'lat': 53.2533834, u'lng': 69.35711859999999}}, u'location': {u'lat': 53.2948229, u'lng': 69.4047872}}, u'formatted_address': u'Kokshetau 020000, Kazakhstan', u'place_id': u'ChIJVy5JP2CUTEIRE1RU19YiGYg', u'address_components': [{u'long_name': u'Kokshetau', u'types': [u'locality', u'political'], u'short_name': u'Kokshetau'}, {u'long_name': u'Zerendi District', u'types': [u'administrative_area_level_2', u'political'], u'short_name': u'Zerendi District'}, {u'long_name': u'Akmola Region', u'types': [u'administrative_area_level_1', u'political'], u'short_name': u'Akmola Region'}, {u'long_name': u'Kazakhstan', u'types': [u'country', u'political'], u'short_name': u'KZ'}, {u'long_name': u'020000', u'types': [u'postal_code'], u'short_name': u'020000'}], u'partial_match': True, u'types': [u'locality', u'political']}]}

Visualizing Data

- Since our ***where.data*** input file only has a few hundred lines, we shouldn't hit the daily rate limit of the ***geocode web service***.
- If you need to process a file that is substantially larger, then you will need to control how much you process per day.

Visualizing Data

- You can halt the running of the program in the ***Canopy IDE*** by selecting "Restart Kernel…" in the "Run" menu.
- A more elegant way of restricting the number of your ***API calls*** is to modify the program to use the count variable with some logic to stop after having issued a certain number of requests.

Visualizing Data

- After loading some data into your ***geodata database***, you can visualize the data by running the ***geodump.py*** program.
- ***geodump.py*** reads the ***geodata.sqlite database*** and writes the file ***where.js***, providing location, latitude, and longitude in ***JavaScript*** code format.

Visualizing Data

- After running the ***geodump.py*** program, if you look at the ***where.js*** file in a text editor you will see that ***myData*** is simply a variable that holds a ***list*** of ***lists***. JavaScript syntax is quite similar to Python syntax, so it should seem familiar and comfortable to you.

Visualizing Data

- After the ***geodump.py*** program has generated the ***where.js*** file, you can then open the ***where.html*** file with your ***web browser***.
- The ***where.html*** file contains a mixture of ***HTML*** and ***JavaScript*** code.
- The ***JavaScript*** code accesses the formatted ***geocode data*** in the ***where.js*** file.

Visualizing Data

- For our second project, we are going to perform the operations of a simple search engine.
- We will use a spider within a small subsection of the web (a website), and then apply a simplified version of Google's page rank algorithm to identify the connectivity values of the various nodes with the web subsection.

www.py4inf.com/code/pagerank.zip

Visualizing Data

- By running the ***spider.py*** program, we crawl a specified web site, and store a series of web pages, along with the links that interconnect them, into an ***sqlite database*** named ***spider.sqlite***.

Visualizing Data


```
%run "C:\UCSD\PythonForInformatics\code\pagerank\spider.py"

Enter web url or enter: http://extension.ucsd.edu/
['http://extension.ucsd.edu']

How many pages:8
1 http://extension.ucsd.edu 51
28 http://extension.ucsd.edu/studyarea/index.cfm?vAction=saDetail&vStudyAreaID=8 43
20 http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=INFO-
70007&vStudyAreaID=10 23
54
http://extension.ucsd.edu/programs/index.cfm?vAction=certDetail&vCertificateID=186&v
StudyAreaID=8 30
8 http://extension.ucsd.edu/international/index.cfm 23
76 http://extension.ucsd.edu/programs/
http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31419&vStudyAreaID=8  0
62 http://extension.ucsd.edu/about/index.cfm?vAction=instructorBio&personid=23 58
69 http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31415&vStudyAreaID=8 23

How many pages:
```

Visualizing Data

- Over successive runs, the spider uses the database to ensure that it does not crawl any pages that it has already crawled.
- When restarted, the program chooses a random web page that hasn't been crawled and begins crawling.
- With successive runs the program expands the crawled net of pages in an additive fashion.

Visualizing Data

- Running the ***spdump.py*** program will provide an output dump of the ***spider.sqlite*** data.

```
%run "C:\UCSD\PythonForInformatics\code\pagerank\spdump.py"
(7, None, 1.0, 1, u'http://extension.ucsd.edu')
(7, None, 1.0, 8, u'http://extension.ucsd.edu/international/index.cfm')
(3, None, 1.0, 54,
u'http://extension.ucsd.edu/programs/index.cfm?vAction=certDetail&vCertificateID=186&
vStudyAreaID=8')
(2, None, 1.0, 62,
u'http://extension.ucsd.edu/about/index.cfm?vAction=instructorBio&personid=23')
(1, None, 1.0, 20,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=INFO-
70007&vStudyAreaID=10')
(1, None, 1.0, 28,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=saDetail&vStudyAreaID=8')
(1, None, 1.0, 69,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31415&vStudyAreaID=8')
(1, None, 1.0, 76, u'http://extension.ucsd.edu/programs/
http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31419&vStudyAreaID=8 ')
8 rows.
```

Visualizing Data

- Running the ***sprank.py*** program calculates and updates page ranks on the crawled pages within the ***spider.sqlite*** database.
- Just specify how many iterations to perform—the more iterations, the sooner the page rank values will converge.

```
How many iterations:8
1 0.857142857143
2 0.365079365079
3 0.153439153439
4 0.0890652557319
5 0.0487948265726
6 0.0216539290613
7 0.00909595662682
8 0.00417510396934
[(1, 2.683622923334856 7), (69, 0.11046334400243 87), (8, 1.7859320225753 66), (20, 0.8923754000914493), (54, 0.3367245846669715)]
```

Visualizing Data

- Re-running the ***spdump.py*** program will show the updated page rank information.

```
%run "C:\UCSD\PythonForInformatics\code\pagerank\spdump.py"
(7, 1.0, 2.6836229233348567, 1, u'http://extension.ucsd.edu')
(7, 1.0, 1.7859320225575366, 8, u'http://extension.ucsd.edu/international/index.cfm')
(3, 1.0, 0.3367245846669715, 54,
u'http://extension.ucsd.edu/programs/index.cfm?vAction=certDetail&vCertificateID=186&
vStudyAreaID=8')
(2, 1.0, 0.2985063252552965, 62,
u'http://extension.ucsd.edu/about/index.cfm?vAction=instructorBio&personid=23')
(1, 1.0, 0.8923754000914493, 20,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=INFO-
70007&vStudyAreaID=10')
(1, 1.0, 0.8923754000914493, 28,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=saDetail&vStudyAreaID=8')
(1, 1.0, 0.1104633440024387, 69,
u'http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31415&vStudyAreaID=8')
(1, 1.0, 1.0, 76, u'http://extension.ucsd.edu/programs/
http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=EDUC-
31419&vStudyAreaID=8 ')
8 rows.
```

Visualizing Data

- If you continue to re-run the ***sprank.py*** program, it will refine the page rank calculations.
- You can always re-run the ***spider.py*** program to crawl some more, and then re-run ***sprank.py*** to process the newly crawled pages.
- A search engine typically runs both the crawling and the ranking operations continuously.
- Running the ***spreset.py*** program will clear the page rank data in the ***spider.sqlite database***.

Visualizing Data

- To prepare our data for visualization, just run the ***spjson.py*** program.
- ***spjson.py*** reads the database and writes formated page rank data for our top ranked pages.

Visualizing Data

- To visualize our page rank data, use your ***browser*** to open the ***force.html*** file.
- If you recalculate the page ranking, just rerun ***spjson.py***, and then refresh your ***browser*** to pull the newly updated data from the ***spider.json*** file.

Visualizing Data

- You can click and drag any node.
- You can double-click on a node and get the ***URL*** associated with that node.

Visualizing Data

- Our third and last project for this lesson is an application that allows us to visualize email data.
- This application can be downloaded from here:

www.py4inf.com/code/gmane.zip

Visualizing Data

- To complete this project, we will utilize a free email list archiving service named www.gmane.org.
- The ***gmane*** service is popular with open source projects because it provides a free, easily searchable archive of their email threads.

Visualizing Data

- ***gmane*** does not enforce a rate limit of any kind.
- They only ask that you do not overwhelm their service, and that you request only what you need.
- This means you should add delays to access requests, and that you spread long tasks over an extended time period.

Visualizing Data

- The ***README.txt*** file contains instructions on how to download a ***content.sqlite*** database file that contains pre-spidered email data.
- By using the pre-spidered data, you avoid having to spend 4 or 5 days running the spidering software to produce your initial data set.

Visualizing Data

- To do your own spidering, you run the ***gmane.py*** program.
- The ***gmane.py*** program uses the Sakai developer list repository by default.
- If you want to use a different repository, just change the value of the default, hard-coded ***URL***, and delete the ***content.sqlite*** file so you start out with a fresh, new database.

baseurl = "http://download.gmane.org/gmane.comp.cms.sakai.devel/"

Visualizing Data

- ***gmane.py*** is coded to run slowly and responsibly.
- Processing a complete respository may take several hours and require many restarts.
- After you have populated your ***content.sqlite database***, you can periodically re-run ***gmane.py*** to update your database with the most recently received emails.

Visualizing Data

- Occasionally, you may notice that gmane.org is missing an email message.
- In such cases, the message was either deleted or simply lost.
- To enable your processing to continue past this obstruction, just manually add a ***row*** to ***content.sqlite*** with the missing message ***id***—all other ***columns*** should be left blank.

Visualizing Data

- The ***content.sqlite database*** is not ***compressed*** or ***normalized***, and is thereby highly inefficient for servicing queries.
- By leaving ***content.sqlite*** in a crudely inefficient form, it makes it easier for humans to inspect, analyze, troubleshoot, and correct any spidering problems.

Visualizing Data

- ***index.sqlite*** has two ***tables*** that map ***domain name*** and ***email addresses*** that might change during the timespan over which the data set is gathered.
- Note, for example, that the following addresses are for the same person, at different times in his career:

s-githens@northwestern.edu
sgithens@cam.ac.uk
swgithen@mtu.edu

Visualizing Data

- To resolve this problem, we add two entries to the ***Mapping table*** in the ***content.sqlite database***:

  s-githens@northwestern.edu -> swgithen@mtu.edu
  sgithens@cam.ac.uk -> swgithen@mtu.edu

- Similarly, the ***DNSMapping table*** allows for the mapping of multiple ***DNS names*** to a single ***DNS name***. For example:

  iupui.edu -> indiana.edu

Visualizing Data

- By successively running the ***gmodel.py*** program, and inserting mappings as necessary, your ***index.sqlite*** data will look progressively cleaner.
- To perform some simple analysis on the ***index.sqlite database***, run the ***gbasic.py*** program.
- You will note that working with the optimized ***index.sqlite*** is vastly more efficient than ***content.sqlite***.

Visualizing Data

- Now you are ready for two fun visualizations: ***gword.py*** and ***gline.py***.
- Running ***gword.py*** will generate the ***gword.js*** Javascript file, which can be visualized by opening ***gword.htm*** in your browser. You should see a ***word frequency cloud***.
- Running ***gline.py*** will generate the ***gline.js*** Javascript file, which can be visualized by opening ***gline.htm*** in your browser. You should see a ***graph depicting email participation by organizations over time***.