ARIA Week 9 Tutorial: Accessing & Transporting Data

Week 9 - March 26th, 2012

PART 2: CREATING API REQUESTS

SUMMARY:
We're going to build API requests for Twitter to learn more about creating requests and parsing the responses.

In this tutorial, we will use ScraperWiki along with some very very basic Python, to create requests and parse them.

REQUIREMENTS:
You will need a ScraperWiki account to build & edit your own scrapers as well as to save your work. If you don't have one, go to https://scraperwiki.com/ and click "Log in" to create an account.

CREATING YOUR FIRST SCRAPER

ScraperWiki is a web application that allows developers to scrape web pages, applications, Excel spreadsheets, PDFs and other information sources for specific information.

[Please Note: many websites or applications, like Ryanair.com, do not allow scraping as it violates their terms of use. Please ensure that you're adhering to the sites' terms and conditions as stated. Everything in this tutorial is fine, just make sure you comply if you go beyond this tutorial.]

1) Take a look at https://scraperwiki.com/scrapers/aras_election_data/

Examine the datastore table to see the categories and data retrieved.
Click "Download" to download the data as a JSON or CSV file and examine it.
Click the column headings to sort the data. Are there repeats in the "from_user" category?

2) Click the "Copy" button. This will create another version of this scraper which you now own.

This code is in Python. Can you understand what it is doing?

Here's a short run down:

- Import libraries needed (scraperwiki so we can save it to a database, simplejson so we can read the results from Twitter, and urllib2 which lets us call the Twitter API).
- Create a query. I broke these out so it would be easier to read. I am using the search terms "aras11 OR aras OR president' to search for people talking about the Aras election in Ireland. I use Geoinfo to limit my search results to people I expect are located in Ireland. I also limit my search to English and 15 pages of results.
- Then we construct the query & send it to Twitter
- Last we take the results, and for each tweet I save the id, text, and user information to a sqlite database.

3) Click the "Run" button to see what hte results look like in the console window below. It may take a while so you can click "Stop" to cancel the search.

4) Now you're going to make your own scraper. Pick a topic to search for, something like Eurovision or a topic of your choice.

Update the query information so that you are searching Twitter for the topic you chose.

5) Run the scraper & take a look at the results in the "Console" window at the bottom. Are the results as you expected?

6) When you're finished, click the "Save Scraper" button.

7) Give your scraper a description and save it to be public.

[BONUS - VISUALISE RESULTS]
8) Now we need to create a web friendly view of this data. On your scraper's main page (the one with the data blurbs and your name and the scraper description), scroll down and click on the "Visualise" button.

9) It will ask you for a language choice. Choose "Python"

10) Your code should look like this. ScraperWiki lets us combine HTML & code here to make it faster & easier to show a view of the data.

```
import scraperwiki
scraperwiki.sqlite.attach("YOUR SCRAPER NAME HERE")

data = scraperwiki.sqlite.select(
    '''* from YOUR SCRAPER NAME HERE.swdata
    order by id desc limit 10'''
)

print "<table>"
print "<tr><th>ID</th><th>Tweet</th><th>User</th>"
for d in data:
    print "<tr>"
    print "<td>", d["id"], "</td>"
    print "<td>", d["text"], "</td>"
    print "<td>", d["from_user"], "</td>"
    print "</tr>"
print "</table>"
```

If it's not working for you, take a look at this one for more information: https://scraperwiki.com/views/wikipediavisualiser/

[BONUS]

If you've gotten this far, step through the ScraperWiki intro tutorial here for more details on how to use ScraperWiki and how to parse API requests and responses:

https://scraperwiki.com/docs/python/python_intro_tutorial/