Stat 222 Project 3: Twitter
Getting started with Python and Twitter

1 Webservices

Many data source are available on the Internet. You've probably used a web browser interface to search through some of this data and even download it to your computer. As you may have noticed, this manual process is labor intensive, error prone, and hard to document.

To allow programmatic and automatic interaction with these data stores, many website serve this information via a documented application programmer interface (APIs).[1] These webservice APIs provide a simple mechanism to create new functionality on top of existing webcontent.

While there are several ways to implement webservices, Representational State Transfer (REST) has gained widespread popularity. REST is more of a style than a standard. A system designed in the REST style is called RESTful. RESTful systems typically use HTTP requests to read and post data using the standard HTTP verbs (GET, POST, PUT, DELETE, etc.). Often some form of authentication is necessary to communicate with a webservice. It is common to use OAuth for this purpose.

2 Data serialization

Normally when you use a webbrowser to view a webpage, your browser handles the HTTP communication for you. You just specify what you wish to view in the form of a URI such as `http://example.org/absolute/URI/with/absolute/path/to/resource.txt`. At this point your browser communicates with the webserver and requests the resource. This resource is typically provided to your webbrowser as an HTML document, which webbrowsers know how to render.

Similiarly, you will use HTTP to communicate with the Twitter webservice. However rather than wishing to know how a webpage should look, you will be interested in retrieving data in a form that is amenable to further processing.

Data serialization is the process of encoding data structures and objects in a format that can be used to store this information on disk or transmit it over the web. For example, you may recall that in R you can use the Rdata format

[1] See Appendix.

to save R objects to disk and reload them later. For webservices, JSON and XML are standard formats. To better understand this, let's briefly look at data serialization more generally.

Python object

First let's create a Python object.

```
>>> mydict = {
...     "parents": [{"name": "mom",
...                  "number": "555-123-4567"},
...                 {"name": "dad",
...                  "number": "555-123-4567"}],
...     "colleagues": {"name": "advisor",
...                    "number": "555-123-4567"}
... }
```

And let's print the results:

```
>>> mydict
{'colleagues': {'name': 'advisor', 'number': '555-123-4567'}, 'parents': [{'name': 'mom', 'r
>>> import pprint
>>> pprint.pprint(mydict)
{'colleagues': {'name': 'advisor', 'number': '555-123-4567'},
 'parents': [{'name': 'mom', 'number': '555-123-4567'},
             {'name': 'dad', 'number': '555-123-4567'}]}
```

XML

How does this object look if we convert it to XML?[2]

```
>>> from dict2xml import dict2xml
>>> print(dict2xml(mydict))
<colleagues>
  <name>advisor</name>
  <number>555-123-4567</number>
</colleagues>
<parents>
  <name>mom</name>
  <number>555-123-4567</number>
</parents>
<parents>
  <name>dad</name>
  <number>555-123-4567</number>
</parents>
```

[2]This functionality is not part of the standard library. And should not be used in practice.

JSON

What if we convert it to JSON?

```
>>> import json
>>> print(json.dumps(mydict, indent=4, sort_keys=True))
{
    "colleagues": {
        "name": "advisor",
        "number": "555-123-4567"
    },
    "parents": [
        {
            "name": "mom",
            "number": "555-123-4567"
        },
        {
            "name": "dad",
            "number": "555-123-4567"
        }
    ]
}
```

YAML

What if we convert it to YAML?

```
>>> import yaml
>>> print(yaml.dump(mydict))
colleagues: {name: advisor, number: 555-123-4567}
parents:
- {name: mom, number: 555-123-4567}
- {name: dad, number: 555-123-4567}
```

Questions

Looking over the output of the above formats you should notice several things.

- Which of the formats uses the largest number of characters?
- Which uses the fewest?
- Which looks most like Python?

Saving JSON and CSV

For this project you will be querying the Twitter webservice and will be getting responses in the JSON format. After you get your response, you will want to

save it to disk. I recommend that you use the JSON format as your main data storage format for this project.

```
>>> import json
>>> with open("data.json", "w") as outfile:
...     json.dump(mydict, outfile, indent=4, sort_keys=True)
...
```

If you decide that you would like to use R for part of your analysis or for creating figures, I recommend saving the information you want to work with in R as a CSV file. Your JSON file will have nested and non-homogeneous structure, which is not possible to directly store using CSV. So you will need to first decide how to

Now you can go ahead an save your list of tuples as a CSV file.

3 Example: US Senate tweets

- connect to Twitter
    - make some queries
    - work with JSON
    - list
    - dictionary
    - string
    - list comprehension
    - saving CSV
    - work in R
    - pca / clustering?
    - ex. senators
    - foreach senator
    - get name
    - get tweets
    - make term document
    - project labeled senators onto 1st and 2nd pcs

Appendix

Links

Webservices

- https://dev.twitter.com/overview/documentation
- https://developers.facebook.com/docs/graph-api
- https://developers.google.com/youtube/getting_started
- http://en.wikipedia.org/w/api.php
- http://www.mediawiki.org/wiki/API:Main_page
- https://developer.github.com/v3/

Serialization

- http://en.wikipedia.org/wiki/Serialization
- http://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats
- http://www.json.org/xml.html
- http://yaml.org/
- http://www.drdobbs.com/web-development/after-xml-json-then-what/240151851
- http://www.cowtowncoder.com/blog/archives/2012/04/entry_473.html