Homework-1-1

February 2, 2017

1 Stack Overflow

1.1 Introduction

In this assignment, we will look at some posts on Stack Overflow during the year of 2015 and measure the similarity of users by looking at the types of questions they answer. Do not delete the output of your code cells. This assignment is to be completed **INDIVIDUALLY** and it is due on **February 10 at 2:00 PM**. No late submission will be accepted.

Please update the README with your BU username.

1.2 Step 0. Preparation

Before we start working on the notebook, let's make sure that everything is setup properly. You should have downloaded and installed * Anaconda * Git

If you are working from the undergraduate lab (on a linux machine) these are both installed, but you need to follow the instructions from here.

1.3 Step 1. Getting the data

Let's make a sample request to retrieve some questions posted on Stack Exchange on the first day of 2015. Documentation of the Stack Exchange API can be found here.

```
In [ ]: import requests
        from datetime import datetime

        start_time = 1420070400 # 01-01-2015 at 00:00:00
        end_time   = 1420156800 # 01-02-2015 at 00:00:00

        response = requests.get("https://api.stackexchange.com/2.2/questions?pagesize=100" +
                                "&fromdate=" + str(start_time) + "&todate=" + str(end_time) +
                                "&order=asc&sort=creation&site=stackoverflow")
        print(response)
```

All dates in the Stack Exchange API are in unix epoch time. The format for the request string is specified here.

We can try to print the response that Stack Exchange returns.

```
In [ ]: print(response.text)
```

It is not possible to read the raw response. Instead, we need to decode the raw response as JSON and use the `json` library to print it.

```
In [ ]: import json
```

```
        json_response = response.json()

        print(json.dumps(json_response, indent=2))
```

Now we can easily see that the response consists of a list of question items. For each of these items, we get information about its attributes such as its `creation_date`, `answer_count`, `owner`, `title`, etc.

Notice that has_more is true. To get more items, we can request the next page.

1.4 Step 2. Parsing the responses

In this section, we practice some of the basic Python tools that we learned in class and the powerful string handling methods that Python offers. Our goal is to be able to pick the interesting parts of the response and transform them in a format that will be useful to us.

First let's isolate the creation_date in the response. Fill in the rest of the `print_creation_dates_json()` function that reads the response and prints the creation dates. Notice that a JSON object is basically a dictionary. **(5 pts)**

```
In [ ]: def print_creation_dates_json(response):
            """
            Prints the creation_date of all the questions in the response.

            Parameters:
                response: Response object
            """
```

Write the code that calls the `print_creation_dates_json()` function to print out all the creation dates of questions posted on the first day in 2015. Please be aware of Stack Exchange's rate limit. **(5 pts)**

```
In [ ]:
```

Due to time constraints, we have downloaded the data dump for Stack Overflow's posts in 2015. The link is only visible to BU students, so you must be logged in to your BU email. Note that the XML file is 10GB. If you don't have space on your computer, you can download it into `/scratch` on one of the machines in the undergrad lab or you can download it onto a USB. You may want to work with a subset of this data at first, but your solution should be efficient enough to work with the whole dataset. For example, if you call `read()` on the whole dataset, you will get a `MemoryError`.

Do not commit the data file. You may assume that we will place the data file in the same directory as your IPython Notebook, so provide a relative path when loading the data file.

Write a function to parse out the questions posted in 2015. These are posts with `PostTypeId=1`. Make a `pandas DataFrame` with 4 columns: `Id`, `CreationDate`, `OwnerUserId`, and the first tag in `Tags`. Print out the DataFrame and do not clear the output. **(10 pts)**

```
In [ ]:
```

1.5 Step 3. Putting it all together

We are now ready to tackle our original problem. Write a function to measure the similarity of the top 100 users with the most answer posts. Compare the users based on the types of questions they answer. We will categorize a question by its first tag. You may choose to implement any one of the similarity/distance measures we discussed in class. **(30pts)**

Note that answers are posts with `PostTypeId=2`. The ID of the question in answer posts is the `ParentId`.

You may find the sklearn.feature_extraction module helpful.

```
In [ ]:
```

Plot the distance of the top 100 users using a heatmap. **(10 pts)**

```
In [ ]:
```