Project A: Twitter dataset Analysis

Assigned: 11 Oct 2016
Due Date: 31 Oct 2016
Project group size: 1
Deliverables: project report, and additional evidence of work done
Submit where: Canvas
Discuss where: project specific thread on Piazza

In this project you will be taking a Twitter data set through a set of steps that could be seen as a manually executed data pipeline – the data pipeline concept taught in the first couple lessons of the course. You will use a subset of the Twitter data, just the user profile portion, 10,000 of them. You will invoke tools to clean the data and import it into the MongoDB database that is running on your Jetstream VM. From there you will invoke additional tools to geolocate the user profiles and display the geolocated user profiles visually.

The tutorials that the AIs have delivered over the first four weeks of the semester should have you well prepared to begin this project with ease. The tutorial material is available to you in the Files directory on Canvas under the directory.

Tutorial material:

Canvas>Files>Project Material	Generate SSH Key.pdf
Canvas>Files>Project Material	Jetstream Tutorial.pdf
Canvas>Files>Project Material	Transfer files into Instance.pdf

From this project you will have learned how to:

- Set up and use a cloud hosted virtual machine
- Use a MongoDB database
- Manipulate software tools that are given to you or that you find at cloud hosted sites to produce a specific visualized result

Students will work alone on this project. That is, project group size = 1. The point of contact for the projects is your Associate Instructor who can be reached through Edx/Piazza. Discussion about the project will take place in a Pizza Discussion group set up for the project.

You will turn in a report, and a couple other pieces of your work. This is described at the end of the project. You will have until Oct 31 2016 to complete the work.

1. Dataset

The dataset we will use in this project is a portion of a Twitter dataset that was created by researchers at the University of Illinois [1][2] from Twitter data that they collected May 2011 and cleaned for instance so that tweets have user profiles and following relationships don't have dangling followers (followers with no profiles).

We obtained the dataset August 2014. It is free and open for use but uses of the dataset beyond this classroom project must include this citation:

Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, Kevin Chen-Chuan Chang: Towards social user profiling: unified and discriminative influence model for inferring home locations. KDD 2012:1023-1031

The full dataset contains Twitter data of three types:

"Following" relationships	followers, followees relationships, 284 million of these relationships amongst 20 million users
User profiles	profiles for 3 million users selected from 20 million users who have at least 10 relationships in the following network.
Tweets	at most 500 tweets for each user in 140 thousand users, who have their locations in their profiles among the 3 million users.

For Project A, you will use just a subset of the user profiles dataset, containing 10,000 user profiles.

Data File	**File Size (approx.)**	**Location**	**Subject**	**Description**
users_10000.txt	730 KB	Canvas >> Files >> Project Materials	User Profiles	10,000 user profiles

2. Setting up your environment

Students are required to carry out your project in a virtual machine that you set up and use on Jetstream. If you prefer to work on another platform, you must demonstrate that your project runs on the Jetstream virtual machine.

2.1 Create and configure Virtual Machine (VM)

Through the tutorials, we have taken you through setting up your own VM for your use, and creating and copying your public key to your VM. If you have not yet set up your Jetstream virtual machine, see the following tutorials on Canvas. Giving your VM a copy of your public key is needed so that you can copy files to your VM. It is also needed if you plan to use a command line shell that is not the Atmosphere shell that Jetstream gives you through its web interface.

Canvas>>Files >>Jetstream Tutorial
Canvas>>Files>>Generate SSH Key.pdf

The VM you configure should be a small one. Do not use a larger VM for this project as it will eat up compute resources unnecessarily. Be sure to shut down your VM when you are done for the day. If you fail to, it will consume compute resources unnecessarily and reduce your quota.

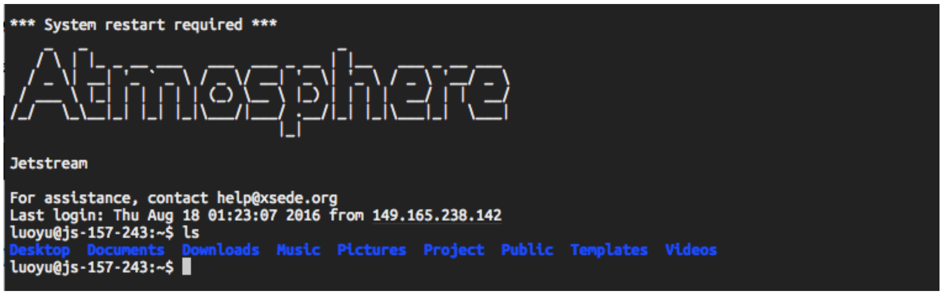
Name of image to use	'I535-I435-B669 Project A'
Image style	m1.tiny (1 CPU, 2 GB memory)

Inside your VM, you will find MongoDB already installed. You will need to disable the Access Control and Authentication in configuration file of MongoDB.

2.2 Set up VM with Pipeline Tools and Dataset

You will need on your virtual machine cleaning and import tools that you will be needing for this project. To get started you will need a project directory (another name for folder) in which to work. From within your VM type the below command to create Project Directory. This project directory will be used for your tools and dataset.

```
mkdir Project
```



```
*** System restart required ***

Jetstream

For assistance, contact help@xsede.org
Last login: Thu Aug 18 01:23:07 2016 from 149.165.238.142
luoyu@js-157-243:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Project  Public  Templates  Videos
luoyu@js-157-243:~$
```

When you log into your VM, you will be in the parent folder */home/your_username.* To move to the folder, project, type

```
cd Project
```

To move back up a level in the directory hierarchy, type

```
cd ..
```

Once you have a project directory set up in your VM, you'll need to get the dataset and tools into that directory.

Tools (the software)	I590-TwitterProjectCode.tar.gz
Dataset	users_10000.txt
Location of both	Canvas>Files>Project Material
Tutorial on getting tools and dataset into VM	Canvas>Files>Project Material>Transfer files into Instance.pdf

2.3 Build Pipeline Tools in VM so they can run

You will need to build the pipeline tools before you can use the. That's because the tools are scripts that invoke Java code, and you'll need to build the Java code (which means compiling the Java code into executable form.)

The two files for both tools and data ('I590-TwitterProjectCode.tar.gz' and 'users_10000.txt') should be in your Project directory at this point. Your command prompt should be in that directory too. To verify that your command prompt is there, type 'pwd' and you will see the current directory: */home/your_username/Project*

Extract the tools from their zipped and tarred package

```
tar -zxf I590-TwitterProjectCode.tar.gz
cd I590-TwitterProjectCode
```

Notice that the second command had you change to a deeper directory. Verify that you are in the directory */home/your_username/Project/I590-TwitterProjectCode* by typing at the command line

```
pwd
```

From this command you should see: */home/username/Project/I590-TwitterProjectCode.* Before building and deploy the code, take a look at the configuration file 'build.properties' by typing

```
cat build.properties
```

Inspect the file to ensure it looks as it does below. We suggest you use the *VI* editor to open the file. A quick cheat sheet for VI is below. You will need to edit both lines (for location of `project.base.dir` and `java.home`) as is appropriate for your VM. Hint: you may just need to replace "username" with your login name.

```
# $Id: build.properties
# @author: Yuan Luo
# Configuration properties for building I590-TwitterProjectCode
project.base.dir=/home/username/Project/I590-TwitterProjectCode
java.home=/usr/bin
```

VI editor cheat sheet

type *vi filename*	open file
type *i*	Insert or add text
hit <Esc>, type *:wq*	save and quit
hit <Esc>, type :q!	Force quit without save
hit <Esc>, type :w	save

Now you are ready to build the Java software. Your cursor should still be in the directory */home/username/Project/I590-TwitterProjectCode*

```
ant
```

To check to see if the build was successful by typing at the command line

```
ls build
```

If you see a new directory called "class", then you know the build was successful. Great work! If not, please ask the AI's for help.

2.4 Using the Tools: Running the Data Pipeline

You will be the master orchestrator of your data pipeline. In other words, you will be manually invoking each tool instead of having the pipeline invoke each in turn automatically.

Your pipeline has three tools. While they look like simple scripts, when you open one of them up, you will see that they make calls to Java code. That's why you had to build the software. All your tools are in the directory called "bin".

Reformat.sh	Converts the data encoding to another encoding
Import_mongodb.sh	Imports twitter user profiles into mongoDB
QueryAndUpdate.sh	Query Google Geo API

Below is the directory structure of the software package. Comments are delimited by parenthesis.

```
I590-TwitterProjectCode
├── bin          (Contains scripts (executables); generated after code deployment)
├── build         (build directory, generated during the code compile time)
│ ├── classes    (class files that generated by java compiler)
│ │ ├── google
│ │ ├── mongodb
│ │ └── util
│ └── lib        (core jar file for scripts in bin)
├── config       (configuration file: config.properties)
├── data         (empty directory, put your data here)
├── input        (query criteria file, query.json, needed for finding docs in MongoDB)
├── lib          (third party dependency library jars)
├── log          (empty directory, log files go here)
├── src          (source code, broken down in next diagram)
│ ├── google
│ ├── mongodb
│ └── util
└── templates  (template files and deploy script. Deploy script generates
                platform-dependent scripts and outputs them to bin during code
                deployment)
```

And the "src" directory, broken down in more detail is organized as follows:

```
src
├── google
│ └── GeoCodingClient.java   (return geocoding results from Google)
├── mongodb
│ ├── Config.java               (extract parameters from configuration file)
│ └── MongoDBOperations.java (select documents that satisfy given query criteria;
│                               update docs by adding geocode)
└── util                        (utility classes)
├── Base64.java                (encoder/decoder)
├── PropertyReader.java       (help class for reading .properties files)
└── UrlSigner.java             (OAuth 2.0 help class)
```

The tools you have are simple but powerful. As said earlier, most are in the form of scripts that invoke Java code. You will execute the analysis as a series of four steps, described below.

Data pipeline task 1: reformat the data

The raw txt file of user profiles is encoded in ISO-8859-1 format. This is a format that the MongoDB NoSQL store does not accept, a common problem. So you will need to convert the txt file into the UTF-8 format that MongoDB accepts. You need to do this before you can store the Twitter user profiles into the MongoDB database.

Reformat the user profile twitter dataset from ISO-8859-1 to UTF-8 format by running the following reformatting script that is in your bin directory. Name the output file

```
./bin/reformat.sh <input file> <output file>
```

(User should *move* *users_10000.txt* data file from Project directory to the */I590-TwitterPorjectCode* directory:

```
mv users_10000.txt I590-TwitterPorjectCode/
```

And then the sample command could be: *./bin/reformat.sh users_10000.txt user_10000.tsv*)

Use *vi editor* to open the file you created. Add the following line as the first line to the newly reformatted Twitter data file (it becomes the "headline", something MongoDB understands). Be sure that you use tabs to split the fields.

```
user_id user_name friend_count follower_count status_count
favorite_count account_age user_location
```

Data pipeline task 2: Import the data into MongoDB

The tab-separated values (tsv) file could be imported directly into MongoDB, however, it has no structure. Adding a header line (i.e., field names for each field) allows MongoDB to give structure to each record. The internal format for MongoDB object is a binary form of JSON (www.json.org).

To import the converted user profile data into MongoDB, run the script ./bin/import_mongodb.sh. The script accepts four parameters:

<db name>	name of database into which data should go
<collection name>	name of collection in database into which data should go
<import file type>	import file type
<file name>	name of tsv file that has data for import

For example:

```
./bin/import_mongodb.sh projectA profile tsv user_10000.tsv
```

Data Pipeline Task 3: Query and Update the User Profile Collection

The Twitter user profile permits a Twitter user to input arbitrary text as their location meaning user locations can be anything. Through the QueryAndUpdate.sh tool you will access the Google geocoding API to validate user locations and extract valid Latitude/Longitude of the user locations. If you are interested in what the Google geocoding API does, take a look here https://developers.google.com/maps/documentation/geocoding

You are now ready to run geolocation on the user profiles in MongoDB to add lat/lon geolocation information to the user profiles. You will need the configuration file and query criteria file which can be found on your VM by following the code package tree structure that we gave you above. The <db name> and <collection name> are the same as how you named these files in the previous step. Note that the tool will exit when the Google geocoding query number reaches the daily limit.

Simple but workable software for doing the geocoding is QueryAndUpdate.sh. It's a script, but you should peek at the Java code that the script invokes to see how it works. The Java code is at src/google/GeoCodingClient.java (see tree structure above). QueryAndUpdate.sh allows you to specify an authentication option in the configuration file that you can find in the config directory (see tree structure above). While Google provides three authentication options, you will use the anonymous user option:

- Anonymous user: limited to making 2500 (or little more) geocoding queries per day. To use this option, leave all authentication configuration parameters blank.

This means you will need to run your tool 4 times over 4 days to finish geocoding all 10,000 user profiles. This workaround is simpler than the other authentication options.

```
./bin/QueryAndUpdate.sh <configuration file> <db name>
<collection name> <query criteria file> <log file>
```

For example

```
./bin/QueryAndUpdate.sh config/config,properties projectA profile
input/query.json test1.log
```

A sample of the geocode information that is added by the Google geocoding service is given below

```
{
"geocode" : {
              "formatted_address" : "Noel N5, Kitimat-Stikine D,

              BC V0J, Canada",
              "location" : { "lat" : 57.4755555,
              "lng" : -132.3597222 }
            }
}
```

The below example shows how to query for a record from within MongoDB. The example queries for user profile user_id:100008949. From there it updates the record to add the geolocation information. Finally, another query is issued showing that the update was successful.

```
$ mongo
> use projectA
switched to db projectA

> db.profile.find({user_id:100008949})
```

```
{ "_id" : ObjectId("5415fc01d77bc408f1397df5"), "user_id" :
NumberLong(100008949), "user_name" : "esttrellitta",
"friend_count" : 264, "follower_count" : 44, "status_count" :
6853, "favorite_count" : 0, "account_age" : "28 Dec 2009 18:01:42
GMT", "user_location" : "El Paso,Tx." }
>

>db.profile.update({user_id:100008949},{$set: {geolocation
:{formatted_address: "El Paso, TX, USA", location:{lat:
31.7775757, lng:-106.6359219}}}})
>

> db.profile.find({user_id:100008949})
{ "_id" : ObjectId("5415fc01d77bc408f1397df5"), "user_id" :
NumberLong(100008949), "user_name" : "esttrellitta",
"friend_count" : 264, "follower_count" : 44, "status_count" :
6853, "favorite_count" : 0, "account_age" : "28 Dec 2009 18:01:42
GMT", "user_location" : "El Paso,Tx.", "geolocation" : {
"formatted_address" : "El Paso, TX, USA", "location" : { "lat" :
31.7775757, "lng" : -106.6359219 } } }
```

QueryAndUpdate.sh uses the find method to query the MongoDB. A sample query criteria used in the find method is this:

```
{
"geocode": {"$exists": false}
}
```

Additional reference for the query criteria is here:
http://docs.MongoDB.org/manual/core/crud-introduction/#query

To check for results, use database commands to query MongoDB directly.
http://docs.mongodb.org/manual/reference/command

Data Pipeline Step 4: Visualization

The final step in the data pipeline is to visualize selected user profiles and their geo-locations using Google Maps. You will select a subset of user profiles (no less than 50) and plot them.

To visualize the geo-location corrected user profile dataset, you will need to export the user names and long/lat coordinates to a csv file and reformat (again!!) it to conform with the format that Google chart JavaScript lib can use. An example of such a format is here below. Note that the first row in the format "arrayToDataTable" gives the field names. This should help you get the lat/long in right places. You will hand in a screen shot of the visualization of your data

```
var data = google.visualization.arrayToDataTable([
      ['Lat', 'Long', 'Name'],
      [37.4232, -122.0853, 'Work'],
      [37.4289, -122.1697, 'University'],
      [37.6153, -122.3900, 'Airport'],
      [37.4422, -122.1731, 'Shopping']
      ]);
```

The original TSV dataset contains 10,000 profiles. After inserting and updating these profiles, most now have real world geo-locations. Your final task is to query MongoDB to extract 50 user profiles with their geo-location information, and plot the geolocations a map.

You will create an html file, which has 50 user profiles with their geo-locations in the 'arrayToDataTable' data structure as above. You then open this html file on your browser to visualize the result.

For more information, see:
https://docs.mongodb.com/manual/reference/program/mongoexport/
https://developers.google.com/chart/interactive/docs/gallery/map
https://developers.google.com/maps/documentation/javascript/tutorial
Visualization API https://developers.google.com/chart/interactive/docs/reference

3 Deliverables

Submit the following through Canvas:

1. The exported portion of your MongoDB dataset in tab separated value form. The dataset will include only those profiles that you chose to plot.
2. The html file that underlies the Google map picture of your selected region.
3. A written report that:
    a. Lists all sources of help that you consulted.
    b. Answers the following questions:
        i. How many locations were you able to validate (i.e., geolocate)? What is the remaining number? Give suggestions for resolving those that you were not able to resolve.
        ii. List ways in which you think this pipeline could be improved, including other tools that could be used.

References

[1] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, Kevin Chen-Chuan Chang: Towards social user profiling: unified and discriminative influence model for inferring home locations. KDD 2012:1023-1031
[2] https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012
[3] MongoDB Reference http://docs.MongoDB.org/manual/reference
[4] Instructions to dump the MongoDB db http://docs.MongoDB.org/manual/reference/program/mongodump