Lesson 9: Project
Twitter Dataset Analysis and Modeling

This project has two tracks, an analysis track (Track A) and a modeling track (Track B).   You choose one based on the track that best fits your background:

Track A Analysis Track does not assume deep familiarity with software programming. The learning objectives are to learn to:

- Set up VirtualBox and download a prepared virtual machine onto your computer,
- Build a software bundle that has your tools in it in the virtual machine,
- Run tools that are in the form of scripts on a dataset of 10,000 Twitter user profiles to store the data, clean the data, and visually plot the location of the users

Track B Modeling Track is geared for the student with a background in computer science; who has familiarity with database schema design, is comfortable with software programming, and who has had experience in performance evaluation of a software service. The learning objectives are to learn to:

- Set up VirtualBox and download a prepared virtual machine onto your computer,
- Design competing MongodB schemas for representing Twitter data,
- Carrying out performance evaluation of the two variant schemas

If you choose Track A, you will pay attention to only Track A and joint (Track A and Track B) items below. If you choose Track B, you will pay attention to Track B items and consult Track A items as needed.   Students will work alone on the project. That is, project group size = 1. The point of contact for the projects is the Associate Instructor who can be reached through Canvas. Discussion about the project will take place in a Canvas Discussion thread set up for the project.

Datasets (applies to both Tracks A and B)

The dataset we will use in this project is Twitter data created by researchers at the University of Illinois [1][2] from Twitter data collected May 2011. It was downloaded from University of Illinois August 2014. The dataset 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 following 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 Track A, you will use just the user profiles database, and a smaller one containing 10,000 user profiles. For Track B, you will use all three types of files. If you have a large enough computer, we urge you to process the full 50 million tweets (and tell us about it)! Download the data files under “Full Dataset”. If your computer memory isn’t big, then you will need a downsampled dataset, called “Partial Dataset” in the table.

Data File	**File Size (approx)**	**Location**	**Subject**	**Description**
10,000 Profiles Dataset (Track A)				
users_10000.txt	730 KB	Download from Canvas	User Profiles	10,000 user profiles
Full Dataset (Track B)				
TwitterCrawl-Aug2012-Network-Full	1.500 GB	Download from http://forward.cs.illinois.edu/datasets/UDI/UDI-TwitterCrawl-Aug2012-Network.zip	Following Network	200 million users following relationships
TwitterCrawl-Aug2012-Profiles-Full	100 MB	Download from http://forward.cs.illinois.edu/datasets/UDI/UDI-TwitterCrawl-Aug2012-Profiles.zip	User Profiles	3 million users’ profiles
TwitterCrawl-Aug2012-Tweets-Full	4.0 GB	Download from http://forward.cs.illinois.edu/datasets/UDI/UDI-TwitterCrawl-Aug2012-Tweets.zip	User Tweets	50 million tweets for 140,000 users
Partial Dataset (Track B for limited memory computers)				
TwitterCrawl-Aug2012-Network-Small	130 MB	Download from Canvas	Following Network	One tenth of full following network dataset

TwitterCrawl-Aug2012-Profiles-Small	45 MB	Download from Canvas	User Profiles	User profiles for reduced following network dataset
TwitterCrawl-Aug2012-Tweets-Small	800 MB	Download from Canvas	User Tweets	Tweets by users in reduced following network

Getting Project Environment Set Up (Tracks A and B)

Students frequently prefer to do projects on their own computers/laptops. To facilitate this, we have built a virtual machine (VM) image containing the Ubuntu operating system and all necessary software. If you are unable to locate your own compute resources on which to run this project, please contact the AI know and we will help you.

Steps to follow:

1) Install the VirtualBox[1] software on your own computer. This allows your computer to host the project virtual machine. Download the latest version of "VirtualBox platform packages" from https://www.virtualbox.org/wiki/Downloads VirtualBox will need about 180MB of space on your computer to install. When you click on the application (VirtualBox) to open it after you have installed it, it will have no Virtual Machines in it. This is OK. You will take care of that later. You do not need to install "VirtualBox Extension Pack" and "Software Developer Kit", which are extensions to Virtual Box. Installation should be easy, but if you need help, consult instructions at https://www.virtualbox.org/manual/ch01.html#intro-installing

Import the virtual machine to VirtualBox.

2) Download the virtual machine that we provide to you. It is large (6.3GB so you are better off downloading it while on the IU network). The VM image is named "I590FALL2015.ova" and it is located in IU Box in a location we will share separately. You will need space on the disk of your computer to store so big a download.
3) Once download has completed, go to the File menu within your VirtualBox window and click: File->Import Virtual Appliance, choose the I590FALL2015.ova file.
4) Change CPU/Memory allocation to your own spec (default value is okay for the analysis track), then click "import". It will take several minutes to finish.
5) In the VirtualBox console, start "I590FALL2015" VM. Login into the VM with user "mongodb" using password "datamanagementcourse".

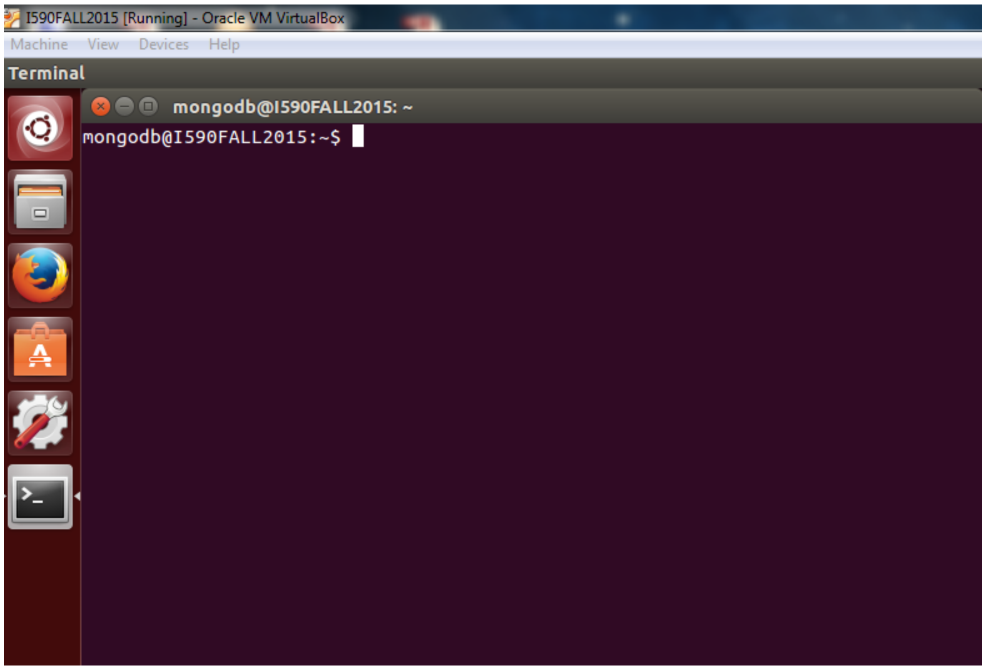
[1] https://www.virtualbox.org

6) Once you become familiar with working in the command line window you will want to change the password to your VM from "datamanagementcourse" to something that is yours. You will do this by typing
   `passwd`
   at the command line prompt.
7) The default network setup will work for this project. At this point, you will be able to access internet from the VM (needed to download other software), but the VM is not accessible from outside your computer. This will be fine for your project.

Building and Installing the Tools to Your VM (Track A)

Download the zipped tarball named I590-TwitterProjectCode.tar.gz to your VM. Do this by opening the Firefox browser, logging into Canvas, and click download on the file. Your file will go into the directory /home/mongodb/Downloads.

Open "Terminal" (sometimes called a command line window). Do this by clicking on the grey icon at the bottom of the icon list shown on left hand side of image below.


I590FALL2015 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Terminal
mongodb@I590FALL2015: ~

```
mongodb@I590FALL2015:~$
```

To show the current directory you are in type at the command line prompt type the command

```
pwd
```

which will return the current working directory, /home/mongodb. From there you will want to change the Projects directory, so type the following command in the window

```
cd Projects
```

Now you will move the tarball you downloaded from where it is at in the Downloads directory to the Projects directory by typing at the command line prompt:

```
mv ~/Downloads/* Projects
```

Over the next set of steps, you will install and build the tools you need for Track A.

Extract the code from its zipped and tarred package, and go to the code directory (your project base directory)

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

Verify that you are in the directory /home/mongodb/Projects/I590-TwitterProjectCode by typing at the command line

```
pwd
```

This command will show you the directory you are currently working in, and should confirm that you are in the directory /home/mongodb/Projects/I590-TwitterProjectCode

Before building and deploy the code, take a look at the configuration file "build.properties" by typing at the command line "

```
cat build.properties
```

The file should look as it does below. If it does not, bring this to the attention of the course AI who will help you.

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

Build and deploy the code by typing the following command from the current working directory (/home/mongodb/Projects/I590-TwitterProjectCode).

```
ant
```

→ Once you reach this point, you can check to see if the build was successful by typing at the command line

```
ls build
```

If you see a classes directory, then the build was successful. Great work!

What the Tools Are (Track A)

You will be using three tools. They look like simple scripts, but 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 bin directory.

First tool converts the way the data is encoded. It is called "reformat.sh". Second tool imports twitter user profiles into mongoDB. It is called "import_mongodb.sh". Third tool is used to query the Google Geo API. It is named "QueryAndUpdate.sh".

Below is the tree structure of the code package. Comments are delimited by parenthesis.

```
I590-TwitterProjectCode
 ├── bin (Contains scripts (executables); generated after the code deployment)
 ├── build (This is a build directory, generated during the code compile time)
 │   ├── classes (.Class files that generated by java compiler)
 │   │   ├── google
 │   │   ├── mongodb
 │   │   └── util
 │   └── lib (contains core jar file for scripts in bin)
 ├── config (contains a configuration file: config.properties)
 ├── data (empty directory, put your data here)
 ├── input (contains a query criteria file, query.json, that is needed for finding
 │          and updating the documents 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. The deploy script generates
                platform-dependent scripts and output them to bin during code deployment)
```

The tree structure of the source code is given below.

```
src
```

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

Running the Tools (Track A)

The tools we give you are simple but powerful. As said earlier, most are in the form of scripts but the scripts hide complexity that you can take look at because you have the code! You will execute the analysis as a series of four steps, described below.

1. Reformat tool: Reformat.sh

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>
```

Double click on the file you created to bring the file up in the "gedit" editor. Add the following line as the first line to the newly reformatted Twitter data file (it becomes the "headerline", 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
```

2. Import tool: import_mongodb.sh

The tab-separated values (tsv) file can be imported directly into MongoDB, however, proper headerlines (fields) must be defined so that MongoDB can give structure to the data when converting it to its internal format (which happens to be JSON (www.json.org) though you don't need to know that).

Next you will import the data into MongoDB. To do so, run the following script which exists in your bin directory. Note that <import file type> is tsv. Choose your own <db name> and <collection name>.

```
./bin/import_mangodb.sh <db name> <collection name> <import file type>
<import file>
```

3. Query and Update the User Profile Collection: QueryAndUpdate.sh

The Twitter user profile permits a Twitter user to input arbitrary text as their location meaning user locations may be fabricated. 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 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>
```

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 you how to enter your MongoDB database and then query a record. The example shows a query of a user profile user_id:100008949, then an update of the record to add the geolocation information, then another query of the user profile for the same user showing that the update was successful.

```
$ mongo
> use twitter_data
switched to db twitter_data
users > db.users.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.users.update({user_id:100008949},{$set: {geolocation :{formatted_address: "El Paso, TX,
USA", location:{lat: 31.7775757, lng:-106.6359219}}}})
>
> db.users.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

4. Visualization

Visualize selected user locations using Google Map.
https://developers.google.com/maps/documentation/javascript/tutorial

To visualize your now corrected user profile dataset, you will need to dump your mongoDB contents into a file.   Take a look at the sample html code here https://developers.google.com/chart/interactive/docs/gallery/map, make a similar html file, with the user profiles in it.

Visualization API https://developers.google.com/chart/interactive/docs/reference

Deliverables (Track A)

Once you complete the project, you will submit the following through Canvas:

A dump of a copy of your modified dataset in MongoDB

A written report that addresses all additional sources of help that you consulted and answers the following questions:

1) How many locations were you able to validate? What is the remaining number? What suggestions do you have for resolving the remainder?

2) The sample query criteria included in the input directory of the code package will let MongoDB find every document that has not been updated with coordinates and update it. What are the alternative solutions of query criteria? If the collection was large, what suggestions do you have to modify the query criteria to improve query performance?

System Track (Track B)

The objective of this project track is to design at least two different data models for the Twitter data that you think would be interesting to evaluate. Then ingest and query a large Twitter dataset against MongoDB. You will conduct a performance evaluation and analysis of both ingest, update and query of these data models. Consult Track A for additional information as needed.

System Track Tasks (Track B)

Download and install VirtualBox. Download the VM provided for the class and install it following instructions given in the section “Getting Project Environment Set Up”. This VM will give you a pre-installed instance MongoDB to use.

Data in MongoDB has a flexible schema. Unlike relational databases, where you must determine and declare a table’s schema before inserting data, MongoDB’s collections (equivalent of an RDBMS table) do not enforce document structure. This flexibility facilitates the mapping of documents to an entity or an object. Each document can match the data fields of the represented entity, even if the data has substantial variation. The key challenge in data modeling is to balance the needs of the application, the performance characteristics of the database engine, and the data retrieval patterns.

In this track, you will:

1) Design two different data models. See http://docs.MongoDB.org/manual/data-modeling/ for data modeling reference.

2) Ingest all Twitter dataset (profile, networks, and tweets) into both data models of MongoDB, using the data models you defined in step 1

3) Perform the following query/aggregation/update operations against the data for both data models.
    a. Return all the user IDs from the tweets, which contain keyword KEYWORD in their text fields. Set the KEYWORD to a high-frequency word (e.g., “good”) first, then set it to a low-frequency word (e.g., “qwertyuiopasdfghjkl”) and run the query again. That is, running two queries for each data model.
    b. Return cumulated retweet counts of all tweets, each of which has at least one hashtag.
    c. Select a user/users who has/have the largest number of followers, find all the followers in the network dataset, and return all the names of the followers (if these names can be retrieved from the profile dataset).
    d. Add a follower to a user, update all the necessary collections.

4) Performance Evaluation:
    a. What is being measured?
        i. You will measure response time of each operation in 2) and 3), for both data models that you designed.
    b. How to measure?
        i. Each operation response time can be measured at the MongoDB client side. Write your MongoDB client code that implements all the operations in 2) and 3). Wrap each of the operations with start and finish timestamps.
        ii. To grab the timestamp, we recommend that you embed the timestamp related code in the same process/thread of your MongoDB client code. Although you can measure timestamps by invoking linux commands (/bin/date) before and after invoking your MongoDB client code, the response time will be less accurate for faster operations. However, the /bin/date command is still acceptable in this project.

Choice of Programming Language: Your code can be extended from the Java code introduced for Track A or can be written in any preferred language that has MongoDB API support. See the list of the MongoDB supported language here http://api.mongodb.org. If you decide to use Java for this track, you can use the Java code and make necessary changes. The Java code base provides a general script to run your java Class, making you code easily deliverable.

Measuring Time: We recommend that you embed the timestamp related code in the same process/thread of your MongoDB client code.

If you plan to use Java, here's a sample code snippet to grab time in milliseconds.

```
long startTime = System.currentTimeMillis(); //Get starting timestamp
//Your MongoDB client code
long endTime = System.currentTimeMillis(); //Get ending timestamp
System.out.println("Total Execution Time: "+(endTime-startTime));
```

If you plan to use JavaScript, timestamp n is the number of milliseconds since 1970/01/01.

```
var d = new Date();
var n = d.getTime();
```

For any of other languages, refer to its manual. However, in this project, you are allowed to use linux command to capture timestamps. Here is an example of capturing timestamps in nanoseconds since 1970/01/01.

```
/bin/date +%s%N
```

Depending on how large memory your VM is allocated and how your code is written, you may experience memory outage during 1) ingest; 2) query operations. If you encounter any out of memory issue, change your code to divide the dataset into several subsets and perform operations against these subsets sequentially, while keeping parallelization in any one subset.

Deliverables (Track B)

Once you complete the project, you will submit the following through Canvas:

1) The scripts/code you wrote showing how you organize the data in MongoDB for each of the data organization you use
2) A brief report containing the following information:
   a) Description of your solution with necessary tables and diagrams
   b) Important design decisions, preferably explained using diagrams, and reason explaining why they are better than other alternatives.
   c) Strengths and weakness of one data model over the other.

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