Project
Twitter Dataset Analysis and Modeling

Due date 21 November 2014 (firm)

Table of Contents

1. Overview

The aim of this project is for students to learn through hands on experience how to handle data and take it through typical big data processing steps: data storing, cleansing, query and visualization. For the advanced computer science student, it gives hands on system level experience of storing big data efficiently.

The project is divided into two tracks: an analysis track and system track.

You will choose one or the other of the tracks. You will state your preference at the beginning of the project.

You will work alone on the project. That is, project group size = 1. Projects are due 21 November 2014. The point of contact for the projects is associate instructor Yuan Luo who can be reached through Canvas. Discussion about the project will take place in a Canvas Discussion thread set up specifically for the project.

2. Project Tracks

2.1. Analysis Track (Track I)

2.1.1. Objective

The objective of this project track is to validate, label, analyze and visualize a Twitter dataset. You will be given a relatively small dataset to store in MongoDB [3] NoSQL store. The dataset consists of three collections, 1) profile, 2) networks, and 3) tweets, though only the user profile collection is used in this track.

2.1.2. Tasks

1) Reformat dataset. The raw txt file of user profiles is encoded in ISO-8859-1, a format MongoDB does not accept. The txt file will need conversion to UTF-8 format.
2) Ingest the reformatted data into MongoDB. A 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 values in the tsv file when converting to its json format.
3) Validate user locations and get geo-coordinates. The Twitter user profile allows user to input of arbitrary text to indicate their location. The user locations may not be recognizable. Use Google geocoding API to validate user locations, extract valid Latitude/Longitude of the user locations. The complete instruction of using Google geocoding API is here https://developers.google.com/maps/documentation/geocoding
4) Store user location coordinates into MongoDB. Done by updating user_profiles collection with necessary schema change.
5) Visualize selected user locations using Google Map.
   https://developers.google.com/maps/documentation/javascript/tutorial
   https://developers.google.com/chart/interactive/docs/gallery/map

2.2. System Track (Track II)

2.2.1. Objective

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 balancing the needs of the application, the performance characteristics of the database engine, and the data retrieval patterns.

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.

2.2.2. Tasks

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’s 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 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.

Note: Additional materials for this track can be found in 7.4.

3. Where Project Runs (for I and II)

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

You will first install VirtualBox on your own machine in order to host the VM.
https://www.virtualbox.org

Set up VirtualBox and VM:

1) Download the latest version of “VirtualBox platform packages” from
   https://www.virtualbox.org/wiki/Downloads
   “VirtualBox Extension Pack” and “Software Developer Kit” are optional.

2) Install VirtualBox on your machine.
   https://www.virtualbox.org/manual/ch01.html#intro-installing
3) Import VM to VirtualBox.
   a. Get VM image (I590FALL2014.ova), link will be provided separately
   b. Go to Virtual menu: File->Import Virtual Appliance, choose the I590FALL2014.ova file, 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.
   c. In the VirtualBox console, start "I590FALL2014" VM. You will login as user "mongodb" using password "datamanagementcourse". Change the password immediately by typing "passwd" into the terminal. See 7.1.1 to locate terminal.
   d. The default NAT network setup should be sufficient for you. At this point, you can access internet from the VM, but the VM is not accessible from the outside. Optionally, if you want to access the VM remotely, set up port forwarding. https://www.virtualbox.org/manual/ch06.html#natforward

See reference for MongoDB documents. http://www.MongoDB.org

4. Dataset (for I and II)

The Twitter dataset [1][2] we will use was downloaded from University of Illinois August 2014. It is a subset of Twitter data containing 284 million of the following relationships: 3 million user profiles and 50 million tweets. The dataset was collected at May 2011. The dataset is free and open to use but all uses of the dataset 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

Dataset makeup

- 284 million following relationships among 20 million users.
- 3 million user profiles, selected from 20 million users who have at least 10 relationships in the following network.
- At most 500 tweets for each user in 140 thousand users, who have their locations in their profiles among the 3 million users.

Data Files

Data File	File Size	Download	Subject	Description
UDI-TwitterCrawl-Aug2012-Network.zip	1.5 GB	link	Following Network	200 million users following relationships.
UDI-TwitterCrawl-Aug2012-	96 MB	link	User	3 million users'

Profiles.zip			Profiles	profiles.
UDI-TwitterCrawl-Aug2012-Tweets.zip	4.1 GB	link	User Tweets	50 million tweets for 140 thousand users.

- **Usage**
    - For the system track, you will use the raw Twitter dataset above.
    - For the analysis track, you will use a smaller txt file of 10,000 user profiles.

5. Deliverable (for I and II)

1. You will set up a time to demo your project with the AI.
2. For the analysis track, you will:
   2.1. demo your project,
   2.2. dump [4] a copy of your modified dataset in MongoDB and turn that in,
   2.3. turn in brief report addressing any sources of help and answering following:
      2.3.1. How many locations were you able to validate? What is the remaining? What suggestions do you have for resolving it?
      2.3.2. 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 went large, how can you modify the query criteria to improve query performance?
3. For the system track, you will:
   3.1. demo your project,
   3.2. submit scripts/code showing how you organize the data in MongoDB for each of the data organization you use,
   3.3. turn in brief report containing the following information:
      3.3.1. Description of your solution with necessary tables and diagrams.
      3.3.2. Important design decisions, preferably explained using diagrams, and reason explaining why they are better than other alternatives.
      3.3.3. Strengths and weakness of one data model over the other.

6. 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] Dump MongoDB db http://docs.MongoDB.org/manual/reference/program/mongodump

7. Appendix (for I and II)

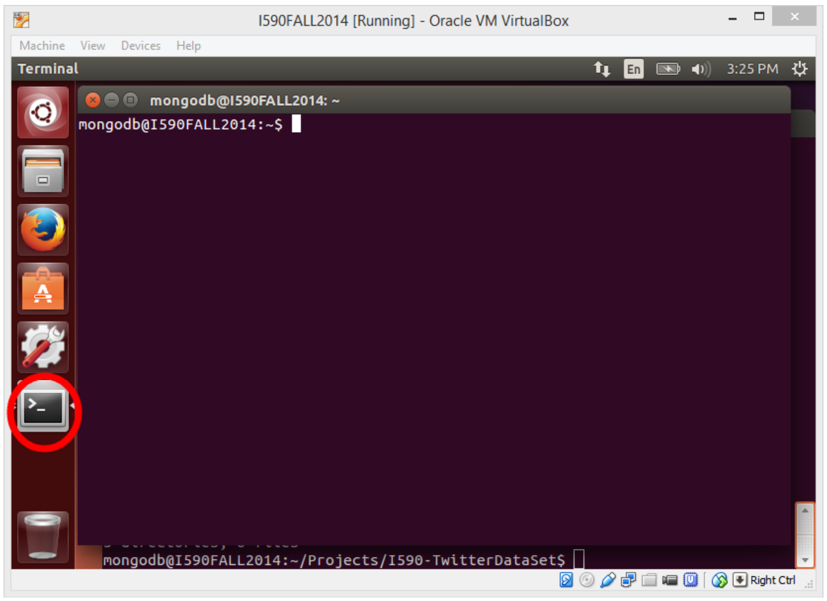
7.1. Project Code

7.1.1. Build the code

1. Download the code tarball (I590-TwitterDataSet.tar.gz) to the VM you installed on your laptop/desktop machine. Place the tarball in the following directory.

```
/home/mongodb/Projects
```

2. Open "Terminal", circled in the figure below.


I590FALL2014 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Terminal
En
3:25 PM
mongodb@I590FALL2014: ~
Right Ctrl

```
mongodb@I590FALL2014:~$
```

```
mongodb@I590FALL2014:~/Projects/I590-TwitterDataSet$
```

3. To show the current directory and change directory to the Projects directory, type the following command in the Terminal

```
pwd
cd Projects
```

4. Extract the code package, and go to the code directory (your project base directory)

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

5. Before building and deploy the code, check the build.properites, make sure it reads like this

```
# $Id: build.properties
# @author: Yuan Luo (yuanluo@indiana.edu)
# Configuration properties for building I590-TwitterDataset
project.base.dir=/home/mongodb/Projects/I590-TwitterDataSet
java.home=/usr/bin
```

6. To build and deploy the code, simply type the following command in your project base directory.

```
ant
```

7.1.2. Understand the code

The code is written in Java. Here's a tree structure of the code package. Comments are delimited by parenthesis.

```
I590-TwitterDataSet
├──── 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 needed for finding
│              and updating the documents in MongoDB)
├──── lib (third party dependency library jars)
├──── log (empty directory, put your log files here)
├──── src (source code)
│     ├──── 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 configuration file (config.properties)

```
#MongoDB hostname and port.
#We recommend that you leave blank; default values will be used (localhost, 27017)
MongoDB.hostname=
MongoDB.port=

#Geocoding Web Service basics
geocode.protocol=https
geocode.baseUrl=maps.googleapis.com
geocode.serviceUrl=/maps/api/geocode/

#Geocoding Web Service output format, default value is json
geocode.outputformat=json

#Google Geocoding authentication. See 7.2 for more details.
#https://developers.google.com/maps/documentation/geocoding
#Option 1: Anonymous: Leave all the following parameters blank.
#Option 2: Simple API Client Key.
```

```
geocode.clientKey=
#Option 3: Google OAuth 2.0 clientID and Secret Key.
geocode.clientID=
geocode.clientSec=
```

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)
```

A sample geocode that added by the code is given below

```
{
"geocode" : {
                "formatted_address" : "Noel N5, Kitimat-Stikine D, BC V0J, Canada",
                "location" : { "lat" : 57.4755555, "lng" : -132.3597222 }
              }
}
```

The equivalent MongoDB operation to the code, but with different query criteria, is shown below.

```
$ 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 } } }
```

The code 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

7.1.3. Run the code

Reformat given twitter dataset from ISO-8859-1 to UTF-8 format, run following reformatting script that is in your bin directory.

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

Add the following line to the newly reformatted Twitter data file as first line (headerline). Make sure that you use tab to split the fields.

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

To import the data, 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>
```

Now you can run the query and update the user profile collection. The location of the configuration file and query criteria file can be found in the code package tree structure in 7.1.2. The <db name> and <collection name> were defined in the previous step. The code will exit when the Google geocoding query number reaches the daily limit. See Google Geocoding details in 7.2.

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

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

If you make any change to the src or lib directory (change any existing code, add any new Java class file and additional third party jar files), you can rebuild and deploy with ant. Here’s a script to help you run any new Java class.

```
./bin/ClassRun.sh <Main Class> <Arguments>
```

For example, the the main method of mongodb.MongoDBOperations Java class takes four arguments:

- args[0]: Configuration File.
- args[1]: DB Name.
- args[2]: Collection Name.
- args[3]: Query Criteria in JSON format.

The corresponding command for running mongodb.MongoDBOperations Java class through ClassRun.sh is,

```
./bin/ClassRun.sh mongodb.MongoDBOperations <configuration file> <db name> <collection
name> <query criteria file>
```

7.2. Google Geocoding

A simple but workable sample of code for geocoding is found in your source tree src/google/GeoCodingClient.java. The code allows you to specify an authentication option in the configuration file in the config directory. Google currently provides three authentication options. Make necessary changes in the configuration file to enable one of them.

- Option 1: Anonymous. It allows ~2500 geocoding queries per day. Leave all authentication configuration parameters blank.
- Option 2: Simple API client key. Provide clientKey in the configuration file. It allows ~2500 geocoding queries per day, providing usage tracking capability from google development console. https://developers.google.com/maps/documentation/geocoding
- Option 3: Google OAuth 2.0 authentication. If you are a business user, you can issue 100,000 queries per day. https://developers.google.com/maps/documentation/business/webservices

We recommend that you enable either option 1 or option 2. For larger dataset, the code will need to be run multiple times. You can run over 4 days to finish 10,000 geocoding.

7.3. Visualization

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 users profile in it.

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

7.4. Additional materials for system track

7.4.1. Write your code

Programming Language: Your code can be extended from the Java code introduced in 7.1, or 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.

Measure 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
```

7.4.2. Possible issues

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.