Lab 9. Using MongoDB

In this lab, you will learn the MongoDB basics, including data insertion, query, projection and index. You will also learn simple analysis with MongoDB. The MongoDB lecture slides have been posted at here.

I. Installation

Install MongoDB software (See also MongoDB installation)

On Ubuntu 14.04 (the virtual machine), the installation is quite simple. Type the following commands line by line in the terminal (the >> sign indicates a line), which will install all necessary components of MongoDB for you.

```
>> sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

>> echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-
org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list

>> sudo apt-get update

>> sudo apt-get install -y mongodb-org
```

Now, MongoDB will be listening at your local port: http://localhost:27017, waiting for the connection from other applications.

Install PyMongo Package

PyMongo package is a driver for MondoDB, which makes it possible to communicate (insert, query, update, etc) with MongoDB with Python language. Type the following command in the terminal.

```
sudo pip install pymongo
```

Note: Anytime when a password is asked, please type *cis4340*

You can learn about the PyMongo package by reading the documentation at following link.

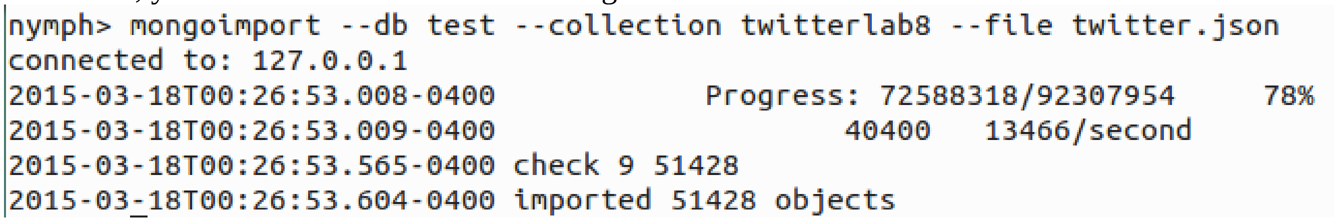
II. Load Twitter data into MongoDB

Download the twitter.json file from the course website. Since each line of the twitter.json file is in JSON format, we can use mongodbimport tool to load the data into MongoDB in batch mode.

1. Open the terminal at the location where you put the twitter.json file.
2. Type the command

```
mongoimport --db test --collection twitterlab9 --file twitter.json
```

3. By doing so, you inserted all tweets into a **collection** named twitterlab9 in **database** called test.
If succeed, you will be able to see the following screen shot.



```
nymph> mongoimport --db test --collection twitterlab8 --file twitter.json
connected to: 127.0.0.1
2015-03-18T00:26:53.008-0400            Progress: 72588318/92307954     78%
2015-03-18T00:26:53.009-0400                      40400    13466/second
2015-03-18T00:26:53.565-0400 check 9 51428
2015-03-18T00:26:53.604-0400 imported 51428 objects
```

III. Connection, query, projection, index with PyMongo

Now, in your Python applications, you can extract data or insert data to MongoDB through PyMongo package. For the definition of concepts like query, projection, index, please look at the lecture slides.

1. Open Spyder.
2. Type the following to connect to the MongoDB.

```
import pymongo
from pymongo import MongoClient
client = MongoClient(max_pool_size=200)
db = client['test']
```

3. Type the following to show one example (or one document) of twitterlab9 collection.

```
db.twitterlab9.find_one()
query = {'user.geo_enabled' : True}
db.twitterlab9.find_one(query)
```

By default, find_one() with no parameters passed will return an arbitrary example; if a condition like {'user.geo_enabled':True} is given, it will return an example that satisfy the condition. The query specify a condition on the truncated field of a tweet, which means the field ' user.geo_enabled == True'.

Remember that a query is defined using a Python dictionary, which is a key-value pair. The key is the field you are looking at, the value is the condition you want to specify on the field.
Q. What is the functionality of the find_one() function?   Where does the retrieved user come from?

4. The following commands will extract all of the tweets that satisfy a specified condition.

```
query = {'user.geo_enabled':True}
results1 = list(db.twitterlab9.find())
results2 = list(db.twitterlab9.find(query))
```

Q. What is the functionality of find() function? What's the purpose of each line? What's is the data type of the results1 and results2 variables (use function `type(results2)` to figure out)?

What's is the size of the *results1* and *results2* variable? What is stored in *result1* and *result2*? What's is the 10th element in result1 and result2?

5. The following commands specify more complex queries using the Python dictionary, e.g. nested document query, and query with operators.

```
query1 = {'geo': {'$ne': None}}
query2 = {'truncated': True, 'source': 'web'}
query3 = {'user.followers_count': {'$gt' : 400 } }
query4 = {'user.followers_count': {'$gt' : 200, '$lt': 300}}
query5 = {'entities.hashtags' : {'$size': 1}}
query6 = {'entities.hashtags.text': 'NCIS'}
query7 = {'retweeted_status': {'$exists':0}}
query8 = {'text': {'$regex': '[Bb]asketball'}}
```

Q. Study the document to get familiar with the operators in MongoDB from here.
What is the condition specified by each query? What is the difference between a field name and an operator name? How many tweets will you get by each query?

6. We can specify the fields that will be shown in the retrieved tweets, by specifying a Python dictionary, which is again a key-value pair, with the key specifying the field you are looking at, the value = 1 to show the field, and value = 0 not to show the field.

```
projection = {'text': 1 }
db.twitterlab9.find_one( query8, projection )
results3 = list(db.twitterlab9.find( query8, projection))
```

Q. What's the purpose of each line? What's the type of results3? What is the 10th element in results3? What's the difference of results3 if you set `projection = {'text':1, '_id': 0}`?

7. We can build an index on a field of a tweet to accelerate the queries executed on the field, (Please refer to the MondoDB tutorial pp. 37 for the definition of index.)

Before we create index, let's evaluate the execution time for query3 by typing the following commands.

```
import time

start_time = time.time()
query3 = {'user.followers_count': {'$gt' : 400 } }
print(db.twitterlab9.find(query3).count())
print("--- %s seconds ---" % (time.time() - start_time))
```

Now, let's build an index on the 'user.followers_count' field.

```
db.twitterlab9.create_index('user.followers_count')
```

Then, evaluate the execution time again using exactly the same code before.

Pay attention that index takes space, so you should be careful how many indexes you create.
Q. What's difference in the execution time before you create the index and after? What's the size of index you created (you can use the command `db.command('collStats', 'twitterlab9')` and get the size from 'indexSizes' field)?

IV. Analysis with MongoDB.

Consider the task of counting the tweets posted by different user languages in the twitterlab9 collection. We can decompose the task into two steps:
1). Find all distinct user languages in the collection. (The 'user.lang' field)
2). For each distinct user id, count the number of tweets published by him/her

```
””” 1). Find all distinct user languages in the collection. (The
'user.lang' field) ”””
dist_langs = db.twitterlab9.distinct('user.lang')
print(len(dist_langs))

”””2). For each distinct user language, count the number of tweets
published by this language”””
counts = [db.twitterlab9.find({'user.lang': lang}).count() for lang
in dist_langs]
print(counts)
```

You have a more fancy and more efficient way to finish the task.

```
results4 = db.twitterlab9.aggregate(
         {'$group':
            {'_id':  '$user.lang',   'count':  { '$sum' :  1} }
         })
results4 = results4['result']
```

The aggregate function first divide the tweets into groups, each of which has the same user.lang. Then for each group, count the number of tweets in the group. For the aggregation pipeline of MongoDB, please look at the MongoDB lecture slide pp. 39 .

Q. Can you evaluate the execution time of both ways and tell the difference? What's the data type of results4 (use `type(results4)`)? What do the elements in results4 mean?

V. Assignments and submissions

Task1 (60). Answer all the questions in the previous sections.
Task2 (10). There is another field 'user.time_zone', show the number of tweets that come from different time zones.

Task3 (15). Find the user that created the most tweets. How many tweets did he/she create? What were those tweets about? The following code will help you.

```
results5 = db.twitterlab9.aggregate(
   {'$group':
       {'_id':  '$user.id',   'count':  { '$sum' :  1} }
   })
results5 = results5['result']
```

Task4 (15). Extract the number of friends for each unique user. Then plot the histogram of the number of friends. Which user has the most friends? How many users have more than 100 friends?

Submit your scripts, writeups through blackboard. Your submission will be graded based on both.