Graded Exercise 6

Accessing and Manipulating Text Data in MongoDB

The Enron Corporation

“Enron Corporation was an energy and commodity services company based in Houston, Texas. It began business in 1985 and grew to be one of the ten largest companies in the United States, achieving a stock price of $90.75 in mid 2000. By November 2001, however, its stock price fell to less than $1 a share. Enron filed for bankruptcy on December 2, 2001, prompting investigations by the U.S. Securities and Exchange Commission and the Federal Energy Regulatory Commission.”

The Enron Scandal

“Enron’s fraudulent practices became a matter of public record and brought corporate accounting and auditing practices into question. The Enron scandal provides a lesson in business ethics, well documented in popular books and articles. Most of the executive employees of Enron and had nothing to do with the scandal, but their e-mail records became a matter of public record when the Federal Energy Regulatory Commission released them as part of its investigation. “

The Enron Email Corpus

“The Enron case data are Enron-centric—for non-Enron actors, we see only their communications with Enron executives. As one of the few sources of real e-mail data in the public domain, the Enron e-mail archive and network represent a substantial opportunity for research in text analytics, online communications, and social networks. The current Enron e-mail corpus, occupying more than two gigabytes of storage and 500 thousand files, contains folders for 158 executives and over 200 thousand e-mail messages. The e-mail network consists of more than 36 thousand nodes and 183 thousand links.”*

* Miller, Thomas W (2015). “Enron E-Mail Corpus and Network”. In *Web and Network: Data Science Modeling Techniques in Predictive Analytics.*

The Investigation

In Gr Ex 6, you will be engaged in a preliminary investigation and analysis of the corpus of thousands of Enron executive emails released into the public domain by FERC (Federal Energy Regulatory Commission). You will be working with the **enron** database in MongoDB. For this assignment you will be connecting to the database using Python.

A sample script is provided (**pymongo_jump_start_script_EA.py**), which is a slightly modified version of the **pymongo_jump_start_script.py** that can be found in the sscc_sps_tutorial_1_2_3_4.pdf file. Your first order of business is to make sure you can connect to the MongoDB using the script. For the script to run you need to have the pymongo module installed. If using Canopy, use the Package Manager to install the module if it is not already installed. Also make sure you are connected to the NU vpn before running the script.

The script prompts you for your netID and password in order to connect you to the **enron** database. Various queries are made on the database and a DataFrame object is created from one of those queries (a keyword search on the term “SilverPeak”).

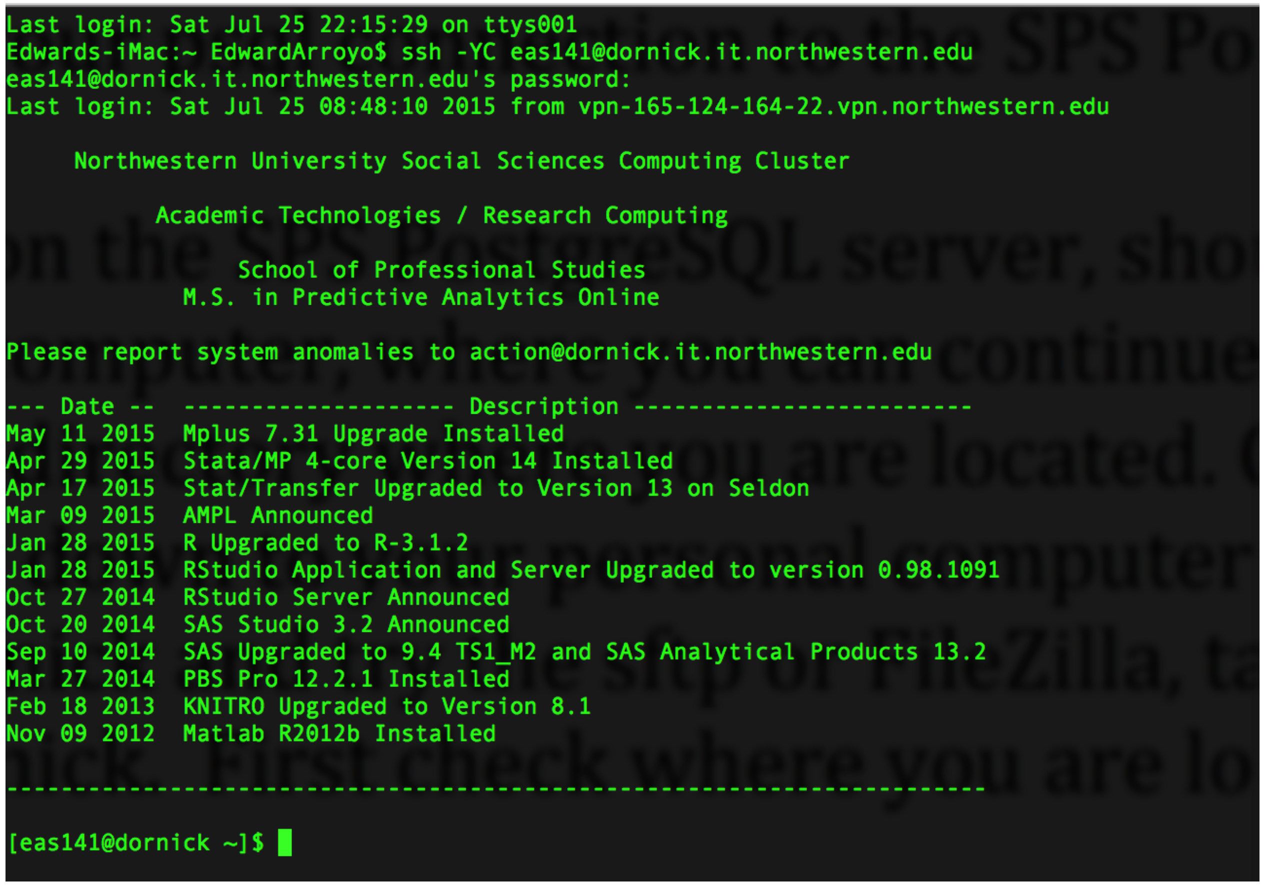
Using a MongoDB Document Collection

Just like you used the *psql* interactive terminal to connect to the SP PostgreSQL server in Gr Ex3, you can use the mongo shell to connect with the SPS MongoDB server.

The following slides goes over this procedure for connecting “manually” while discussing some of the features of MongoDB in the process. We then discuss the **pymongo_jump_start_script_EA.py** script in more details and the requirements for the assignment.

Connecting "Manually" to the MongoDB Server

Step 1: Use SSH to connect to Dornick as in Gr Ex 3:



```
Last login: Sat Jul 25 22:15:29 on ttys001
Edwards-iMac:~ EdwardArroyo$ ssh -YC eas141@dornick.it.northwestern.edu
eas141@dornick.it.northwestern.edu's password:
Last login: Sat Jul 25 08:48:10 2015 from vpn-165-124-164-22.vpn.northwestern.edu

     Northwestern University Social Sciences Computing Cluster

           Academic Technologies / Research Computing

                 School of Professional Studies
             M.S. in Predictive Analytics Online

Please report system anomalies to action@dornick.it.northwestern.edu

--- Date --  -------------------- Description -------------------------
May 11 2015  Mplus 7.31 Upgrade Installed
Apr 29 2015  Stata/MP 4-core Version 14 Installed
Apr 17 2015  Stat/Transfer Upgraded to Version 13 on Seldon
Mar 09 2015  AMPL Announced
Jan 28 2015  R Upgraded to R-3.1.2
Jan 28 2015  RStudio Application and Server Upgraded to version 0.98.1091
Oct 27 2014  RStudio Server Announced
Oct 20 2014  SAS Studio 3.2 Announced
Sep 10 2014  SAS Upgraded to 9.4 TS1_M2 and SAS Analytical Products 13.2
Mar 27 2014  PBS Pro 12.2.1 Installed
Feb 18 2013  KNITRO Upgraded to Version 8.1
Nov 09 2012  Matlab R2012b Installed

------------------------------------------------------------------------

[eas141@dornick ~]$
```

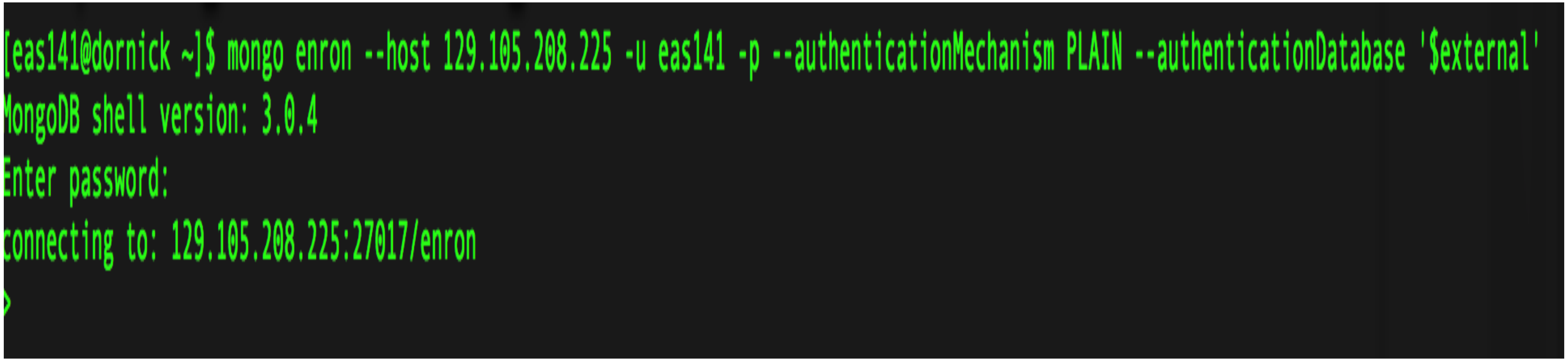
Set up Steps

Step 2: Create your working directory, switch to it and then start your script:

```
mkdir individual_assignment_3
cd individual_assignment_3
script individual_assignment_3.log
```

Connecting to the MongoDB Server

Step 3: Log on to the Mongo shell:



```
[eas141@dornick ~]$ mongo enron --host 129.105.208.225 -u eas141 -p --authenticationMechanism PLAIN --authenticationDatabase '$external'
MongoDB shell version: 3.0.4
Enter password:
connecting to: 129.105.208.225:27017/enron
>
```

Careful to type in the whole command on **one line**!

The MongoDB Database

MongoDB is an open-source document database which works on concept of *collection* and *document*. Related MongoDB (JSON-style) *documents* are grouped into *collections* and stored in *databases*. Each document in the **messages** collection of the **enron** database stores information related to a particular email message. You will be searching documents in this collection for occurrences of certain keywords.

Sample Document from Messages collection

```
{
            "_id" : ObjectId("4f16fc97d1e2d32371003e27"),
            "body" : "the scrimmage is still up in the air...\n\n\nwebb said that they didnt want to scrimmage...\n\nthe aggies are scrimmaging each
other... (the aggie teams practiced on \nSunday)\n\nwhen I called the aggie captains to see if we could use their field.... they \nsaid that it was tooo
smalll for us to use...\n\n\nsounds like bullshit to me... but what can we do....\n\nanyway... we will have to do another practice Wed. night.... and I
dont' \nknow where we can practice.... any suggestions...\n\n\nalso, we still need one more person...",
            "headers" : {
                        "Date" : "Tue, 14 Nov 2000 08:22:00 -0800 (PST)",
                        "From" : "michael.simmons@enron.com",
                        "Message-ID" : "<6884142.1075854677416.JavaMail.evans@thyme>",
                        "Subject" : "Re: Plays and other information",
                        "To" : "eric.bass@enron.com",
                        "X-From" : "Michael Simmons",
                        "X-To" : "Eric Bass",
                        "X-bcc" : "",
                        "X-cc" : ""
            },
            "mailbox" : "bass-e",
            "subFolder" : "notes_inbox"
}
```

Comparison to Relational Database Systems

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents

The Use command

The use command can be used to create a database or, if it already exists, return the existing database.

```
> use enron
switched to db enron
```

For more info about the **use** command go here.

Show Collections

```
> show collections
mbox
messages
pilot
system.indexes
```

Indexes in MongoDB

- Indexes are a special data structure that allow for the fast and efficient execution of queries.
- MongoDB provides text indexes to support text search of string content in documents of a collection.
- Text indexes can include any field whose value is a string or an array of string elements. To perform queries that access the text index, use the **$text** query operator. We will see examples of this later on.

Collection Methods

```
 db.messages.help()
DBCollection help
            db.messages.find().help() - show DBCursor help
            db.messages.count()
            db.messages.copyTo(newColl) - duplicates collection by copying all documents to newColl; no indexes are copied.
            db.messages.convertToCapped(maxBytes) - calls {convertToCapped:'messages', size:maxBytes}} command
            db.messages.dataSize()
            db.messages.distinct( key ) - e.g. db.messages.distinct( 'x' )
            db.messages.drop() drop the collection
            db.messages.dropIndex(index) - e.g. db.messages.dropIndex( "indexName" ) or db.messages.dropIndex( { "indexKey" : 1 } )
            db.messages.dropIndexes()
            db.messages.ensureIndex(keypattern[,options])
            db.messages.explain().help() - show explain help
            db.messages.reIndex()
            db.messages.find([query],[fields]) - query is an optional query filter. fields is optional set of fields to return.
                                                  e.g. db.messages.find( {x:77} , {name:1, x:1} )
            db.messages.find(...).count()
            db.messages.find(...).limit(n)
            …
```

The db.collection_name.count(**query**) method

- db.collection_name.count(**query**), where query is the selection criteria, returns the count of documents that would match a find( ) query.
- The db.collection_name.count(**query**)method does *not* perform **find( )** operation but instead counts and returns the number of results that match the **query**.
- It is similar to the SQL count( ) method in relational databases:

```
SELECT COUNT(*) FROM table_name;
```

The find( ) and findOne methods

To query data from MongoDB collection, you need to use MongoDB's **find()** method:

*db.collection_name.find(**query**)*

To return *one* item use:

*db.collection_name.findOne(**query**)*

To return **n** items, for some numer **n**, use:

*db.collection_name.find(query) .limit(**n**)*

Some MongoDB Resources

- https://docs.mongodb.com/manual/reference/operator/query/text/
- http://www.tutorialspoint.com/mongodb/mongodb_create_database.htm
- http://www.querymongo.com/ (Converts SQL queries to MongoDB queries)

The “Jump-Start” Script

Your starting point for this assignment is a slightly revised version of the “jump-start” script from the tutorial: **pymongo_jump_start_script_EA.py**.

Note, in particular, that I replaced "Silverpeak" by "\"Silverpeak\"" in the search for documents containing that term. This was not strictly necessary in this case. However, if a phrase contains punctuation (such an email address) the punctuation is used a delimiters in the search. So to match on such phrases you must enclose them in escaped double quotes (\") (cf. https://docs.mongodb.com/manual/reference/operator/query/text/).

Using pymongo to connect to MongoDB

```
from pymongo import MongoClient
⋮
try:
    client = MongoClient("129.105.208.225")
    client.enron.authenticate(my_netid, my_password, source='$external',
    mechanism='PLAIN')
    print('\nConnected to MongoDB enron database\n')
    success = True
except:
    print('\nUnable to connect to the enron database’)
```

Number of documents in message collection

```
# work with documents in the messages collection
workdocs = client.enron.messages

# inquire about the documents in messages collection
print('\nNumber of documents: ', workdocs.count())
```

```
Connected to MongoDB enron database
Number of documents:  501513
```

Find “Silverpeak” documents in MongoDB

```
#print('How many documents contain the string <Silverpeak>?')
print(workdocs.find({'$text':{'$search':'Silverpeak'}}).count())
```

```
How many documents contain the string <Silverpeak>?
27
```

Getting one document in MongoDB

```
print('\nOne such document:\n', workdocs.find_one())
```

```
One such document: {u'body': u"the scrimmage is still up in the air...\n\n\nwebb said that they didnt want to scrimmage...\n\nthe aggies  are scrimmaging each other... (the aggie teams practiced on \nSunday)\n\nwhen I called the aggie captains to see if we could use their field.... they \nsaid that it was tooo smalll for us to use...\n\n\nsounds like bullshit to me... but what can we do....\n\n\nanyway... we will have to do another practice Wed. night....    and I dont' \nknow where we can practice.... any suggestions...\n\n\nalso,  we still need one  more person...", u'headers': {u'X-cc': u'', u'From': u'michael.simmons@enron.com', u'X-bcc': u'', u'To': u'eric.bass@enron.com', u'X-From': u'Michael Simmons', u'Date': u'Tue, 14 Nov 2000 08:22:00 -0800 (PST)', u'X-To': u'Eric Bass', u'Message-ID': u'<6884142.1075854677416.JavaMail.evans@thyme>', u'Subject': u'Re: Plays and other information'}, u'_id': ObjectId('4f16fc97d1e2d32371003e27'), u'subFolder': u'notes_inbox', u'mailbox': u'bass-e'}
```

Load the MongoDB data into a frame…

```
# flatten the nested dictionaries in selectdocs
# and remove _id field
list_of_emails_dict_data = []
for message in selectdocs:
tmp_message_flattened_parent_dict  = message
tmp_message_flattened_child_dict  = message['headers']
del tmp_message_flattened_parent_dict['headers']
del tmp_message_flattened_parent_dict['_id']
tmp_message_flattened_parent_dict.\
update(tmp_message_flattened_child_dict)
list_of_emails_dict_data.\
append(tmp_message_flattened_parent_dict.copy())
# we can use Python pandas to explore and analyze these data
# create pandas DataFrame object to begin analysis
enron_email_df_2  = pd.DataFrame(list_of_emails_dict_data)
```

“Flattening” Nested Dictionaries

We have seen this before in Graded Exercise 3 where you first had to “flatten” a nested dictionary (of hotel reviews) in order to load the data into a DataFrame for further analysis. You don’t really need to understand how this flattening code works (and might even have some ideas for doing it differently yourself). You just need to modify the query to produce the data you are interested in in this assignment, load the data into a DataFrame, and do a bit of analysis using pandas.

Some sample analysis…

```
# We can modify the query to get all the emails. It takes a while…
selectdocs =list(workdocs.find())

# set missing data fields
enron_email_df_2.fillna("", inplace=True)

#Add a column containing the number of emails in each "To" field
enron_email_df_2['To Size']=enron_email_df_2['To'].map(lambda addr_lst:
len(addr_lst.split()))
```

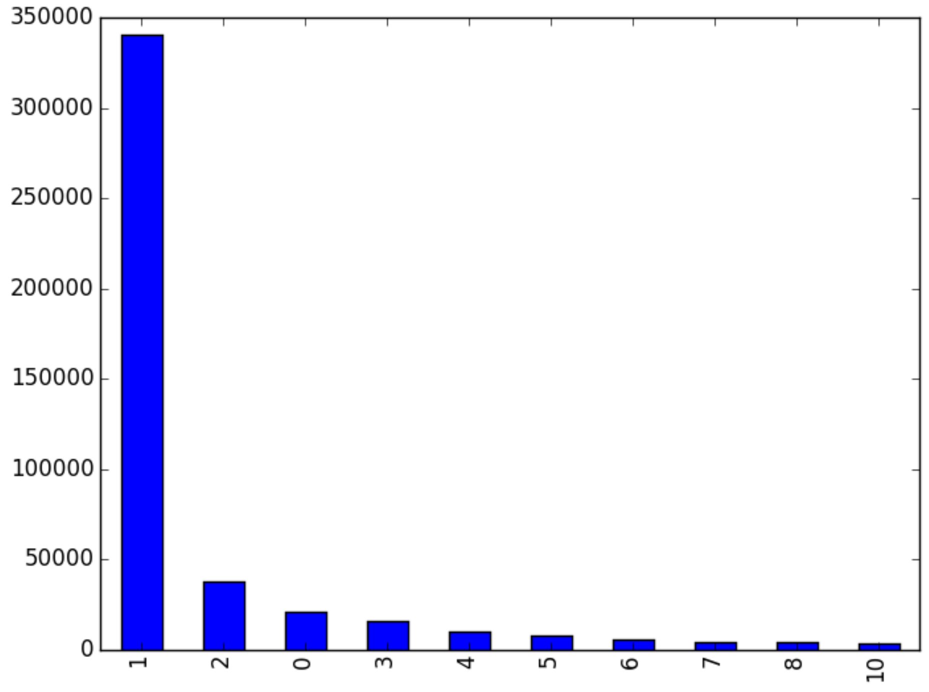
Some Sample Analysis (continued)…

```
enron_email_df_2['To Size'].value_counts(ascending=False)
```

```
1      340226
2       37608
0       21349
3       15867
4       10137
          ⋮
276         1
558         1
392         1
354         1
1031        1
361         1
```

Bar graph of the ten most common…

```
enron_email_to_frequencies[:10].plot(kind='bar')
```



Graded Exercise 6 Instructions

The first part of the assignment

- Import the messages that include Key Lay's email address into a list of dicts.
- "Flatten" the nested dicts as needed and import it into a DataFrame object.

Modify the given script so that you search emails containing Ken Lay’s email address: klay@enron.com. Note that search is case *insensitive*. This is something you will need to deal with in the analysis part of assignment

Graded Exercise 6 Instructions (Cont’d)

Then, do the following:

1. Determine how many of the messages are "To:" Ken Lay, "From:" Ken Lay, and on which Ken Lay was cc'd or bcc'd. Get a count for each of these.

You can use Boolean filtering on the DataFrame object to get a new DataFrame object of messages sent to Ken Lay and then use count( ) to count how many records there are. Similar remarks apply to messages sent to Ken Lay for which he is cc’d or bcc’d.

Graded Exercise 6 Instructions (Cont’d)

2. Who did Lay send the most emails to? Who did he receive the most from?

You can use the ”groupby” method with DataFrame objects from #1.

3. Did the volume of emails sent by Lay increase or decrease after bankruptcy was declared?

You can use filtering to divide the records into two DataFrames: those date before and those data after BANRUPTCY. Count the records...

3. How many of these email messages mention Arthur Andersen, Enron's accounting firm?

Just look for his name in the body of the emails...

Two helpful pandas string methods

- str.upper()
- str.contains()

http://pandas.pydata.org/pandas-docs/stable/api.html#string-handling

These methods can be applied to any DataFrame column you happen to be analyzing.