Storing Tweets

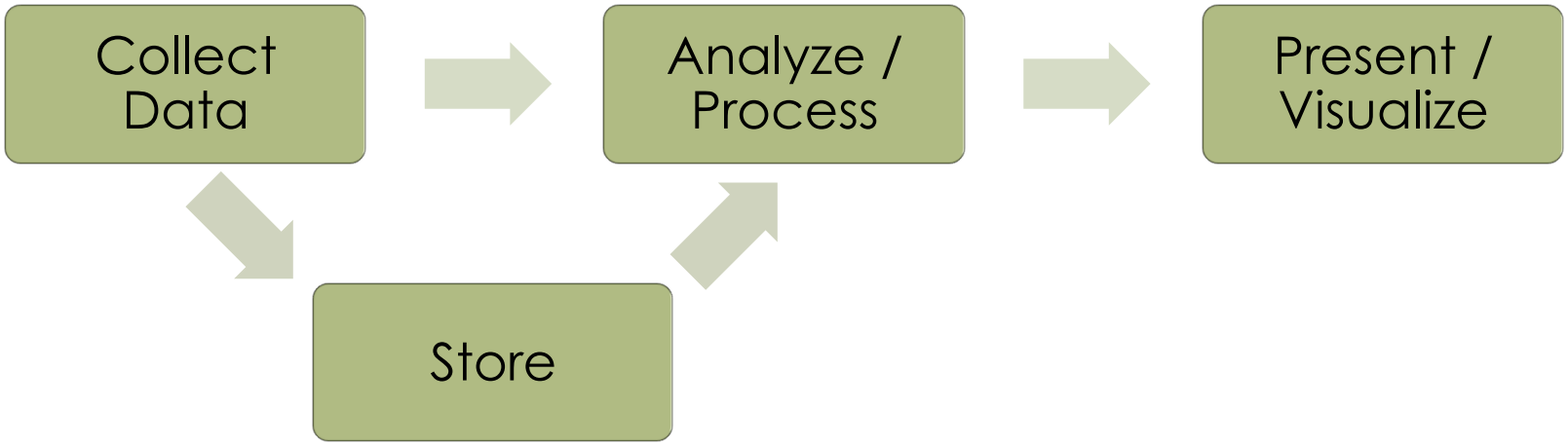
Pickling, MongoDB

Connecting to MySQL through Python

Outline

- Storing or Streaming
- Simple Storage: Pickling in Python
- Intro to Mongo DB
- Connecting and using MySQL

Processing pipeline



- Streaming vs Storing

Simple Storage

- Tweet data response – JSON
  - This is text
  - What does this look like?

Simple Storage

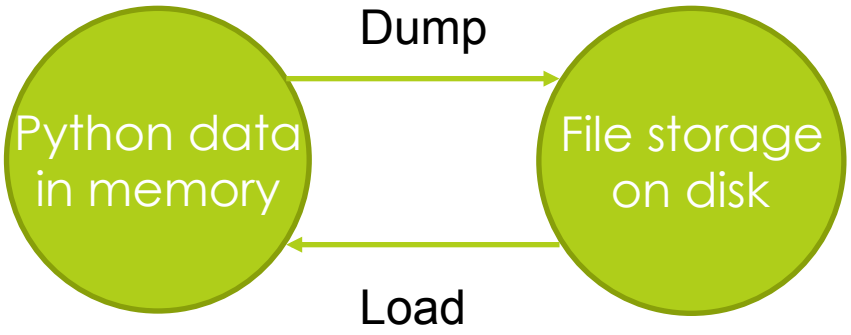

```
{
    "created_at": "Wed Jan 15 05:34:43 +0000 2014",
    "id": 423327363959492610,
    "id_str": "423327363959492610",
    "text": "RT @nflnetwork: Anyone up for some Football?\n\n@Seahawks vs.
@49ers. Week 14. Right now. #NFLReplay http://t.co/12bqr3He9f",
    "user": {
        "followers_count": 87,
        "friends_count": 400,
        "id": 245167474,
        "id_str": "245167474",
        "location": "Dallas,Tx",
        "name": "La Troy Woodruff",
        "screen_name": "LockDown_22",
        "verified": false
    }
}
```

Simple Storage

- Tweet data response – JSON
  - This is text
  - What does this look like?
- JSON is more useful as a python dictionary
  - Using the json module
  - import json
  - Sample read a JSON text file and convert to python dictionary

Storage

- Loading and dumping



Read A JSON File (example 1)

```
import sys, json, pickle, cPickle

def read_json_file(fname=""):
    jsdict = dict()
    fdoc = ""
    if fname:
        fin = open(fname, "r")
        for line in fin:
            fdoc=fdoc+line
        fin.close()
        jsdict = json.loads(fdoc)
    else:
        print "Must supply a filename"
    return jsdict
```

Read A JSON File (example 2)

```
import sys, json, pickle, cPickle

def read_json_jsonload (fname=""):
    jsdict = {}
    if fname:
        fin = open(fname, "r")
        jsdict = json.load(fin)
    else:
        print "Must supply a filename"
    return jsdict
```

Write a JSON File

- What would it take to save a JSON file?

Write a JSON File

- What would it take to save a JSON file?
  - JSON is just text
  - We've had prior samples of reading/writing text

Alternatives to JSON?

- JSON
  - Easy to read, edit – it's just text
  - Slow to load, read and convert to dict (python dictionary)
- Pickle
  - Raw data, dictionary
  - Quick to load, no conversion
- Both are built into Python, simple
  - More sophisticated DBs require more work

Save data (pickle)

```
import sys, json, pickle, cPickle

def pickle_dictionary(fname="",d={}):
    if fname:
        fout = open(fname, "w")
        pickle.dump(d,fout)
        #cPickle.dump(d,fout)
    else:
        print "Must supply a filename"
    return
```

Load data (unpickle)

```
import sys, json, pickle, cPickle

def unpickle_dictionary(fname=""):
    d = {}
    if fname:
        fin = open(fname, "r")
        d = pickle.load(fin)
        #d = cPickle.load(fin)
    else:
        print "Must supply a filename"
    return d
```

Little demo

- Pickling and unpickling

Intro to Using MongoDB

SQL is...

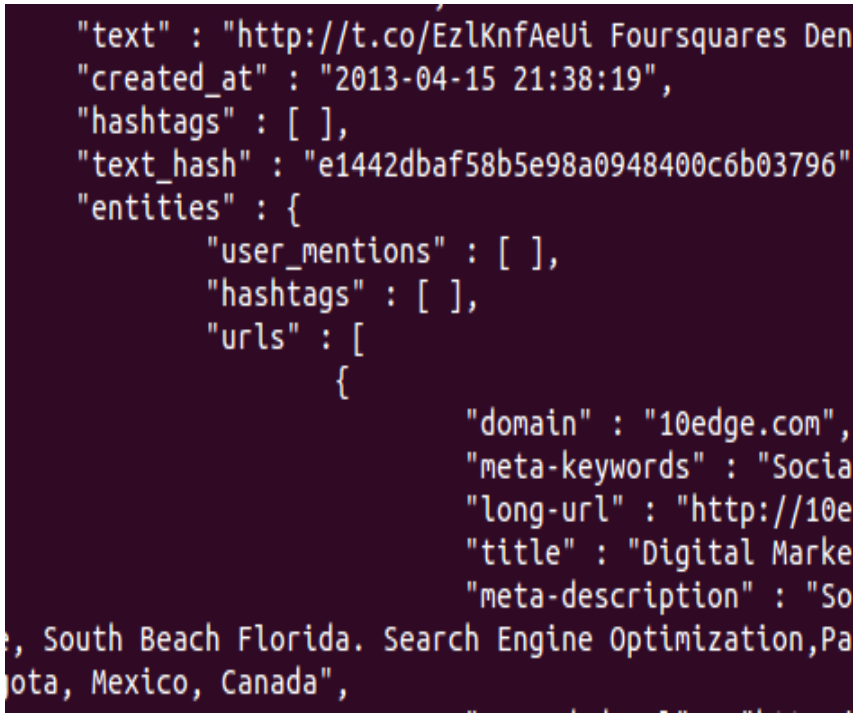
- relational
- “table-like” structure
- SELECT <columns> FROM <table> WHERE...
- vertically scalable
- powerful for processing multiple documents

```
mysql> select id,author,tweet_source,time from tweets limit 10;
+---------+----------------+-------------------+---------------------+
| id      | author         | tweet_source      | time                |
+---------+----------------+-------------------+---------------------+
| 4211615 | Charlotteee_xox | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211616 | kfhradio       | TweetDeck         | 2013-04-15 21:25:13 |
| 4211617 | Bambs01        | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211618 | Atilajuventino | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211619 | kelseysparkss  | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211620 | Patty_Jack     | web               | 2013-04-15 21:25:13 |
| 4211621 | Kanelbullenn   | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211622 | TVconormac     | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211623 | AndreaPucillo  | Twitter for iPhone | 2013-04-15 21:25:13 |
| 4211624 | TLStokes21     | Twitter for iPhone | 2013-04-15 21:25:13 |
+---------+----------------+-------------------+---------------------+
10 rows in set (0.00 sec)
```

Mongo is...

- non-relational
- json based structure
- machine code:

  db.find({ <field> :
  <query> })
- horizontally scalable
- powerful for accessing individual documents



Some Similarities

- dbs = databases
- collections = tables
- documents = rows
- logical operators
  - $and = and
  - $or = or
  - $regex = like

Some MongoDB Advantages

- web APIs return json - complex data models with arrays
- documents can contain different fields
- no migration and slow joins to update data model
- aggregation, mapreduce
- query syntax CAN be easier to understand in driver code
- maps to objects (like sql alchemy)

examples

Create a new "document" in the "students" collection:

```
db.students.insert( {
     'name': "sue",
     'class': "freshman"
     'gpa': 3.5
  } )
```

```
INSERT INTO students
name, class, gpa
VALUES
"sue", "freshman", 3.5;
```

examples

find all documents with the class "freshman" and gpa greater than 3:

```
db.students.find( {
        'class' : 'freshman',
        'gpa' : { $gt : 3 }
} )
```

```
SELECT *
FROM students
WHERE class like 'freshman'
AND gpa >= 3;
```

reference

- download: http://www.mongodb.org/
- intro: http://docs.mongodb.org/manual/
- twitter and mongo: the book!
- sql to mongo: http://docs.mongodb.org/manual/reference/sql-comparison/
- specific questions: Google!
- python connector: pymongo

Connecting to MySQL

Using the HCDE user data module

Using MySQL in Python

- Need MySQL installed
- Need a DB (Schema/table structure)
- Need a Python to MySQL connector
- Nice to have a Object-Relation Manager (ORM)

Using MySQL in Python

- Need MySQL installed
  - You just need to install this
- Need a DB (Schema/table structure)
  - Sample provided in hcde.data.db.schemas
    - e.g. “tweet_hcde530-db-schemas.sql”
- Need a Python to MySQL connector
  - pymysql
- Nice to have a Object-Relation Manager (ORM)
  - SQL Alchemy

Test the install

```
[Tiki:~Development/python/] dwmc% python
Python 2.7.10 (default, Oct 23 2015, 18:05:06)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pymysql
>>> import sqlalchemy
>>>
```

Basic Twitter SQL Schema

- Tweets Table
- User Table
- Friends Table
- Followers Table
- User Meta Table

Simple Twitter Schema Demo

- Browse - hcde.data.db.schemas
  - tweet_hcde530-db-schemas.sql

ORM (Object-Relation Manager)

- An ORM Manages
  - The relation between the data in the memory of the machine (while the program is running) and the more permanent storage of that data in a database
  - Changes to an 'object' in memory is (eventually) mirrored to the database.

Working With sqlalchemy (ORM)

- Need to develop an Python Object abstraction for each database table
  - See hcde.data.db.base

    `tweetObj.py`
    `userObj.py`
    `userMetaObj.py`
    `friendObj.py`
    `followerObj.py`

- Also, convenient to have an abstraction for the DB itself

  `TweetsDB.py`

- These are all usable base classes, we probably want specialized versions for a real data collection

Specializing ORM Classes

- Specialized "example" classes
  - See hcde.data.db.example

```
ExampleTweetObj.py
ExampleUserObj.py
ExampleUserMetaObj.py
ExampleFriendObj.py
ExampleFollowerObj.py
ExampleTweetsDB.py
```

- These could be used for your project if you wanted to use MySQL for your storage

Demo Connecting

- HCDE user module tries to hide some of the complexity of using ORM
  - hcde.data.db.base
    - dbConfig() object
    - baseDB() object
  - hcde.data.db.fitness
    - FitTweetsDB()
    - FitTweetObj()
    - settings_db.py

Demo Using Fitness Data

- HCDE user module has four datasets
  - hcde.data.election_2012
  - hcde.data.election_2016
  - hcde.data.fitness
  - hcde.data.oscar_2016

- Let's try working with the fitness data