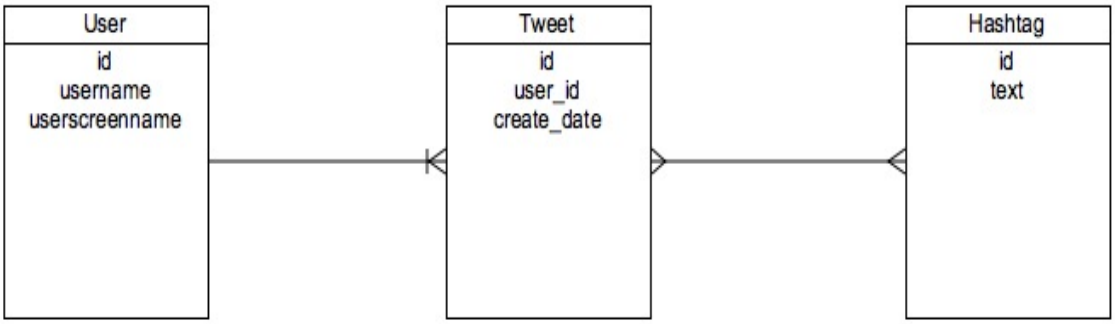
Malini Mittal
W205
Assignment 3

Task 1:



Task 2:

Data Cleaning:

For all the three formats, each tweet that gets read in will be cleaned to store only the required fields:

```
for tweet in tweets:
    d = {}
    d['created_at'] = parser.parse(tweet['created_at']).isoformat()
    d['userhashtags'] = [h[u'text'] for h in tweet['entities']['hashtags']]
    d['userscreenname'] = tweet['user']['screen_name'].encode('utf-8')
    d['username'] = tweet['user']['name'].encode('utf-8')
    d['tweet_id'] = tweet['id']  # needed for S3 and sqlite
    # … any other operation … #
```

Key/Value (AWS S3):

Data Organization:

The data will be organized to store each tweet as a single value. To be able to answer the questions about the user with maximum tweets and the number of tweets produced in a particular hour, the following organization can be used:

s3://ConferenceTweetData/dates/<datetime>/<user_screen_name>/<tweet_id>.json

For the question regarding the top hashtags used, the following organization will be used:

s3://ConferenceTweetData/hashtags/<hashtag>/<tweet_id>

Storage:

To store information in this model, my python script will first create the main bucket. Then it will load each file, iterate through the tweets, create the two keys as follows:

1. key1 = ''.join('dates/', d['created_at'], '/', d['userscreenname'], '/', str(d['tweet_id']), '.json')

   The tweet will be uploaded onto S3 as a json file on this key.

2. The number of keys generated for hashtags will depend on the number of hashtags in the tweet:
   ```
   for ht in d['userhashtags':
       hashkey = ''.join('hashtags/', ht, str(d['tweet_id']))
       …
   ```
   For each of these keys, set the contents from string by converting the tweet_id into a string.

Retrieval:

Create an S3 connection using boto, and then get the bucket:

```
conn = S3Connection()
bucket = conn.get_bucket('ConferenceTweetData')
keylist = bucket.list('hashtags')
```

1) Who tweeted the most during the conference?

```
def getUserDictionary():
    datesiter = bucket.list("dates/", "/")
    user_dict = {}
    foreach date in dateiter:
        dt = date.name.strip('/').split('/')[-1]
        useriter = bucket.list(date.name, '/')
        for user in useriter:
            username = user. name.strip('/').split('/')[-1]
            if username not in user_dict:
                user_dict[username] = 0
            tweetiter = bucket.list(user.name)
```

```
                numtweets = 0
                for tweet in tweetiter:
                        numtweets += 1
                user_dict[username] += numtweets

    return user_dict

udict = getUserDictionary()
sortedlist = sorted(udict, key=udict.__getitem__, reverse=True)[0:5]
print sortedlist
```

2) What were the top ten hashtags used?

```
def getHashTagDictionary():
    hashtagsiter = bucket.list("hashtags/", "/")
    hashtag_dict = {}
    for hashtag in hashtagiter:
            name = hashtag.name
            tweetlist = bucket.list(name)
            count = 0
            for tweet in tweetlist:
                    count += 1
            hashtag_dict[name.strip('/').split('/')[-1]] = count
    return hashtag_dict

htdict = getHashTagDictionary()
sortedlist = sorted(htdict, key=htdict.__getitem__, reverse=True)[0:10]
print sortedlist
```

3) How many tweets were produced each hour?

```
def getTweetCount(timestart, timeend):
    datesiter = bucket.list("dates/", "/")
    numtweets = 0
    foreach date in dateiter:
            dt = date.name.strip('/').split('/')[-1]
            if (dt >= starttime and dt < endtime):
                    useriter = bucket.list(date.name, '/')
                    for user in useriter:
                            tweetiter = bucket.list(user.name)
                            for tweet in tweetiter:
                                    numtweets += 1
    return numtweets
```

```
date1 = datetime.datetime.strptime("2015-02-14T08:00:00", "%Y-%m-
%dT%H:%M:%S")
date2 = date1 + datetime.timedelta(days=1)
for day in (date1, date2):
    for hour in range(0,7):
        starttime = day + datetime.timedelta(hours=hour)
        endtime = day + datetime.timedelta(hours=hour+1)
        numtweets = getNumTweets(tweets, starttime.isoformat(),
        endtime.isoformat())
        print ("Number of tweets on {0} between {1} and {2} =
        {3}").format(starttime.date(), starttime.time(), endtime.time(), numtweets)
```

NoSQL (MongoDB):

Data Organization:

Create a database called 'test_database'. Then create a collection in this database called 'test_tweet_collection'.

Storage:

All the cleaned up tweets will be stored in this collection as separate documents.

Retrieval:

```
# get the database and the collection

client = MongoClient('mongodb://localhost:27017/')
db = client.test_database
tweets = db.test_tweet_collection
```

1. Who tweeted the most during the conference?

```
def userTweetCount(tweets, outdb):
    #count the number of tweets per user, and sort
    pipe = [
        { '$group': {'_id': '$userscreenname', 'total' : {'$sum' : 1}}},
        { '$sort': {'total': -1}},
        { '$out': outdb}
    ]
    tweets.aggregate(pipeline=pipe)

userTweetCount(tweets, "users")
users = db.users
print "users:"
for user in users.find().limit(5):
    print user
```

2. What were the top ten hashtags used?

```
def hashtagCount(tweets, outdb):
    #count the number of times each hashtag occurs, and sort
    pipe = [
            { '$project': {'_id': 0, 'hashtags': '$userhashtags.text'}},
            { '$unwind' : '$hashtags'},
            { '$group': {'_id': '$hashtags', 'total': {'$sum': 1}}},
            { '$sort': {'total': -1}},
            { '$out': outdb}
    ]
    tweets.aggregate(pipeline=pipe)

hashtagCount(tweets, "hashtags")
hashtags = db.hashtags
for hashtag in hashtags.find().limit(10):
    print hashtag
```

3. How many tweets were produced each hour?

```
def getNumTweets(tweets, timestart, timeend):
    return tweets.find({"created_at" : { "$gte" : timestart, "$lt" :
                                timeend}}).count()

date1 = datetime.datetime.strptime("2015-02-14T08:00:00", "%Y-%m-
%dT%H:%M:%S")
date2 = date1 + datetime.timedelta(days=1)
for day in (date1, date2):
    for hour in range(0,7):
        starttime = day + datetime.timedelta(hours=hour)
        endtime = day + datetime.timedelta(hours=hour+1)
        numtweets = getNumTweets(tweets, starttime.isoformat(), endtime.isoformat())
        print ("Number of tweets on {0} between {1} and {2} =
                {3}").format(starttime.date(), starttime.time(), endtime.time(),
                numtweets)
```

Relational Database (sqlite):

Data Organization:

The following four tables will be created, with schemas as shown:

1. Hashtags:

```
CREATE TABLE Hashtags(
        id INT PRIMARY KEY,
```

```
        name TEXT NOT NULL
);
```

2. Tweets:
```
CREATE TABLE Tweets(
        id INT PRIMARY KEY,
        date TEXT,
        user_id INT,
        FOREIGN KEY(user_id) REFERENCES Users(id)
);
```

3. Tweets_Hashtags:
```
CREATE TABLE Tweets_Hashtags(
        tweet_id INT,
        hashtag_id INT,
        FOREIGN KEY(tweet_id) REFERENCES Tweets(id),
        FOREIGN KEY(hashtag_id) REFERENCES Hashtags(id)
);
```

4. Users:
```
CREATE TABLE Users(
        id INT PRIMARY KEY,
        name TEXT,
        screen_name TEXT NOT NULL
);
```

Storage:

Iterating through the tweets after loading the json files, the relevant data is added to the appropriate tables.

Retrieval:

1. Who tweeted the most during the conference?

```
def printUserWithMostTweets(cursor, num):
    query = 'Select user_id, Count(*) from Tweets Group by user_id ORDER by Count(*)
    DESC'
    cursor.execute(query)

    cursor2 = conn.cursor()
    for i in range(0, num):
        row = cursor.fetchone()
        query = ''.join(['Select * from Users where id=', str(row[0])])
        cursor2.execute(query)
        print ("total = {0}, screen_name = {1}").format(row[1], cursor2.fetchone()[2])
```

2. What were the top ten hashtags used?

```
def printMostUsedHashtags(cursor, num):
    query = 'Select hashtag_id, Count(*) from Tweets_Hashtags Group by hashtag_id
    ORDER by Count(*) DESC'
    cursor.execute(query)

    cursor2 = conn.cursor()
    for i in range(0, num):
        row = cursor.fetchone()
        query = ''.join(['Select * from Hashtags where id=', str(row[0])])
        cursor2.execute(query)
        print ("total = {0}, hashtag = {1}").format(row[1], cursor2.fetchone()[1])
```

3. How many tweets were produced each hour?

```
def getNumTweets(cursor, starttime, endtime):
    query = ''.join(['Select Count(*) from Tweets where date >= "', starttime, '" and date
    < "', endtime, '"'])
    cursor.execute(query)
    data = cursor.fetchone()
    if data == None:
            return 0
    return data[0]
```