Rodney Bruno

Task 1: Following a log file

Tue. Sep 9 2015 Start: 6:21pm

Saw all events for one day (9/17/2014) and greped for browser

-Now greping for one person RachaelBrandenburg

\* Should I have to pipe each time I add a new thing ex: “Name , File name pipe browser?

- saved info. under rachaelexample.txt

|  |  |  |  |
| --- | --- | --- | --- |
| Video | Event Type | Old /New/CT\* | Time |
| N/A | Page\_close | `````` | 01:39:19.730385+00:00 |
| Biochemistry 2 | Showed transcript |  | 01:39:22.793895 |
| Loaded video |  | 01:39:24.604557 |
| Seq\_goto | :1/:3 | 01:39:25.465169 |
| :3/:5 | 01:39:36.347012 |
| :5/:7 | 01:39:44.335374 |
| :7/:9 | 01:40:01.209566 |
| :9/:10 | 01:40:13.572081 |
| Seq\_next | :10/:11 | 01:41:08.727863 |
| Seq\_goto | :11/:1 | 01:41:31.592769 |
| Showed Transcript |  | 01:41:31.596401 |
| Loaded video |  | 01:41:33.745439 |
| Play\_video |  | 01:41:40.370781 |
| Pause\_video | :81.7132114\* | 01:43:02.501937 |
| Stop\_video | 01:43:02.916594 |
| seq\_next | :1/:2 | 01:43:04.491502 |
| Showed Transcript | 0\* | 01:43:04.616174 |

Important notes:

-Discover of Current time could be key in the project

-Having a code to note all of these events would be helpful

-We should watch the video to know what is happening during that section

-Logging everything can be very tedious

Questions:

What is the difference between Seq\_next and seq go/to?

Ask brian to get IDLE and ask to add adobe

 seq\_goto is emitted when a user jumps between units in a sequence.

 seq\_next is emitted when a user navigates to the next unit in a sequence.

Could we add a computer programing major to this project?

End 9/29/15 7:00pm

Code try 2

RP

Rachel Peterson <*lollipops\_peterson*@comcast.net>

|

To:

Rodney Bruno;

Wed 9/30/2015 3:34 PM

print "Goal: Finding Watchers of Videos"  
  
  
  
import csv #This imports ability to read it  
  
from sets import Set #?  
  
import liblytics  
  
  
  
print "Translating video names..."  
  
#read in video list with nice names  
  
videoNames = {}  
  
with open("Video Names F14 2.csv", "rU") as f:  
  
    reader = csv.reader(f)  
  
    for row in reader:  
  
        videoNames[row[0]] = row[1] #cannot remember how this line works exactly  
  
  
  
print "Determing valid usernames..."  
  
#make usernames into a set  
  
usernames = Set()  
  
with open("SPOC Grades Edited.csv", "rb") as f:  
  
    reader = csv.reader(f)  
  
    for row in reader:  
  
        usernames.add(row[1])  
  
  
  
print "Putting videos with usernames..."  
  
#add usernames and video ids associated with "play\_video" to data  
  
data = {}  
  
for line in liblytics.read\_log\_file("umass\_boston-edge-events-ALL.gz"):  
  
    if line["event\_type"] == "play\_video":  
  
        username = line["username"]  
  
        if (username != ""):  
  
            if username not in data:  
  
                data[username] = Set()  
  
            event = line["event"]  
  
            if not isinstance(event, dict):  
  
                try:  
  
                    event = eval(event)  
  
                except (NameError):  
  
                    event = event.replace("null", "0")  
  
                    event = eval(event)  
  
            data[username].add(event["id"])  
  
  
  
print "Looking for unmatched videos..." #Videos that are not included in a file  
  
unmatchedVideos = Set()  
  
for username in data:  
  
        for video in data[username]:  
  
                if video not in videoNames.values():  
  
                        unmatchedVideos.add(video)  
  
          
  
print "   Found these that were not in Video Names file:"  
  
for name in unmatchedVideos:  
  
        print name  
  
  
  
print "Determining who watched what..."  
  
#writing to a new csv file  
  
#determining if a video has been watched by a user  
  
f = open("Testing F14 7.csv", "w")  
  
f.write("username,")  
  
for cleanName in sorted(videoNames):  
  
    f.write (cleanName)  
  
    f.write (",")  
  
f.write (" \n")  
  
for username in sorted(data):  
  
    if username in usernames:  
  
        f.write (username)  
  
        f.write (",")  
  
        ~~for cleanName in sorted(videoNames):  
  
            if videoNames[cleanName] in data[username]:  
  
                f.write ("1,")  
  
            else:  
  
                f.write ("0,")~~  
  
        f.write (" \n")  
  
f.close()

October 7, 2015 ~7:30pm

Important points

-Prints username, time, and event type for “browser events”

-Sample: {'': Set(['page\_close', '2013-12-20T14:56:40.555572+00:00']), 'ousham13': Set(['2013-12-03T21:45:34.178256+00:00', 'page\_close']),…

import liblytics

from sets import Set

data={}

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"):

if line["event\_source"] == "browser": #This looks for the browser event type

username=line["username"]

time=line["time"]

data[username] = Set()

if username != " ":

event = line ["event\_type"]

data[username].add(event)

data[username].add(time)

print data

Question:

How to remove the word “sets” from it?

How to add time “time” and format?

Goals:

Make “events” in a separate csv

Go over definition function

End 10/7/15 8:20pm

Does not work

import liblytics

from sets import Set

data={}

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"):

if line["event\_source"] == "browser" or line["event\_source"] == "mobile": #This looks for the browser event type

username=line["username"]

T=line["time"]

data[username] = Set()

if username != " ":

ET = line ["event\_type"]

data[username].add("Time: " + T)

data[username].add("Event Type: " + ET)

if ET != ("page\_close", "problem\_show", "problem\_graded"):

E= line ["event"]

data[username].add("Event: " + E)

print data

10/21/15

Program Realizations:

Play video, page close, and pause video all include the current time. Current time could be used be use to see how much of the video they watched.

Program Goals:

Have a set under the username to hold “id” (video name) and “CurrentTime” of the event, and the event types.

Have it count the first play and calculate distance from next play video or page close.

10/29/15

Code created to make multiple layers of dicts (GolgenEgg.py).

Example: ctosidase': {'playCount': 0, 'timeLastWatched': 0}, 'i4x-MITx-7\_00x\_UMass-video-Enzymes\_and\_biochemical\_reactions': {'playCount': 0, 'timeLastWatched': 0}}, 'Jonaysia': {'i4x-MITx-7\_00x\_UMass-video-DNA\_Polymorphisms\_within\_Human\_Populations': {'playCount': 0, 'timeLastWat…

Code:

import liblytics

from sets import Set

data={}

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"):

if (line["event\_type"] == "play\_video") or (line["event\_type"] == "pause\_video"):

username=line["username"]

# time=line["time"]

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersVideoDict = data[username]

if videoName not in usersVideoDict:

x = {}

x["timeLastWatched"] = 0

x["playCount"] = 0

usersVideoDict[videoName] = x

userVideoDataDict = usersVideoDict[videoName]

print data

Changes (test.py):

Imported time and calander to

Goal:

Get time in second format

Count all play videos

Create loop to calculate different in second for 1st to second play video then 2-3 ect.

Questions:

Why dies it show only one time when it loops

11/3/15

PREVIOUS PROGRAM

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

#t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

#times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersVideoDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersVideoDict: #second loop creates third dict

videoInfoDict = {}

videoInfoDict["playCount"] = 0

pCount= videoInfoDict["playCount"]

videoInfoDict["Times"] = []

time\_list= videoInfoDict["Times"]

usersVideoDict[videoName] = videoInfoDict

userVideoDataDict = usersVideoDict[videoName]

for times in line['time']:

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

time\_list.append(times)

pCount = pCount + 1

print data

**Try 1**

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

#t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

#times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

#if videoName not in usersDict: #second loop creates third dict

# videoDict = {}

# videoDict["playCount"] = 0

# pCount = videoDict["playCount"]

# videoDict["Times"] = []

# time\_list = videoDict["Times"]

# usersDict[videoName] = videoDict

#userVideoDataDict = usersDict[videoName]

#for times in line['time']:

# t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

# times = time.mktime(t)

# time\_list.append(times)

# pCount = pCount + 1

print data

#expected result: {username, {} }

Result: {'ousham13': {}, 'jackiegirl33': {}, 'RonyP': {},

**Try 1 was a success**

**Try 2**

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

#t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

#times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates second dict

videoDict = {}

# videoDict["playCount"] = 0

# pCount = videoDict["playCount"]

# videoDict["Times"] = []

# time\_list = videoDict["Times"]

# usersDict[videoName] = videoDict

#userVideoDataDict = usersDict[videoName]

#for times in line['time']:

# t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

# times = time.mktime(t)

# time\_list.append(times)

# pCount = pCount + 1

print data

#expected result: {username, {video id {}, video id 2 {}, etc.} }

#current goal: get {username, {video id {play count: etc, times: etc}, video id 2 {play count: ... }, username 2 ...}

Result: {'ousham13': {}, 'jackiegirl33': {}, 'RonyP': {},

**Try 2 failed.**

**Try 3**

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

#t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

#times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates second dict

usersDict[videoName] = {}

videoDict = usersDict[videoName]

# videoDict = {}

# videoDict["playCount"] = 0

# pCount = videoDict["playCount"]

# videoDict["Times"] = []

# time\_list = videoDict["Times"]

# usersDict[videoName] = videoDict

#userVideoDataDict = usersDict[videoName]

#for times in line['time']:

# t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

# times = time.mktime(t)

# time\_list.append(times)

# pCount = pCount + 1

print data

#expected result: {username, {video id {}, video id 2 {}, etc.} }

#current goal: get {username, {video id : {play count: etc, times: etc}, video id 2 {play count: ... }, username 2 ...}

Result: {'ousham13': {'i4x-MITx-7\_00x\_UMass-video-Week\_s\_Introduction': {}}, 'jackiegirl33': {'i4x-MITx-7\_00x\_UMass-video-Fractionating\_life': {}},

**Try 3 was somewhat successful**

11/4/15

program: allplaysf13.py

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

time\_list= videoDict["Times"]

time\_list.append(times)

videoDict["playCount"] = 1

usersDict[videoName] = videoDict

else:

time\_list.append(times)

videoDict["playCount"] = len(time\_list)

userVideoDataDict = usersDict[videoName]

print data

This code currently counts all the play videos for each person example.

Example: {'ousham13': {'i4x-MITx-7\_00x\_UMass-video-Amazing\_proteins\_tertiary\_and\_quaternary\_structure': {'playCount': 16, 'Times': [1387365502.0, 1387365504.0, 1387365507.0, 1387365510.0, 1387365513.0, 1387365531.0, 1387365533.0, 1387365621.0, 1387365643.0, 1387365654.0, 1387365696.0, 1387365698.0, 1387365700.0, 1387365708.0, 1387365710.0, 1387365713.0]}, 'i4x-MITx-7\_00x\_UMass-video-Biological\_applications\_in\_forensics\_history\_and\_agriculture': {'playCount': 2, 'Times': [1387355679.0, 1387355681.0]}, 'i4x-MITx-7\_00x\_UMass-video-Medical\_revolutions\_in\_Biology': {'playCount': 7, 'Times': [1387355421.0, 1387355423.0, 1387355423.0, 1387355451.0,

Notes:

Time\_list.append(times) is needed in both the *if* and the *else* loop to add the first time the program sees the video and user and the subsequent times

In the *if* loop the play count can equal 1 because that loop will only be processed once for each user-video pair. Then, if that user-video pair appears again, it will be replaced each time by the number of items in the list.

What to do next?

Another loop needed to find difference between times is within windows and subtract

11/5/15

Idea to calculate difference:

* The ones with only one time do not have a difference in times
* Even number play count
  + Grabs last time and next probably in a while statements
  + Timecount = play count (or len of time)
  + if number%2==0:

While timecount >0

* + - Get dufferece and add to (diffence section of dict)
    - Timecount = Timecount -1
  + Else:
  + While timecount >1
    - Get diffence and add to (diffence section of dict)
    - Timecount = Timecount -1

Nest week goals:

1. Make excel file of current data example row username / head column video name and

|  |  |  |
| --- | --- | --- |
| Names | Video 1 | Video 2 |
| Jane | Playcount | 2 |

2. Create histogram of time differences

1/9/15

Goal: Add difference in times to a list

program 2

Name: diffinplays.py

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

time\_list= videoDict["Times"]

time\_list.append(times)

videoDict["Playcount"] = 1

usersDict[videoName] = videoDict

else:

time\_list.append(times)

videoDict["Playcount"] = len(time\_list)

timecount = len(time\_list)

videoDict["TDiffer"] = []

if timecount > 1:

T1 = timecount - 1

T2 = timecount - 2

secdiff = time\_list[T1] - time\_list[T2]

diff\_list = videoDict["TDiffer"]

diff\_list.append(secdiff)

timecount=timecount-1

userVideoDataDict = usersDict[videoName]

print data

Things it does: It subtracts only the last two numbers (works for itmes with two times)

Problems: Does not add additional time differences to list

Example: {'ousham13': {'i4x-MITx-7\_00x\_UMass-video-Amazing\_proteins\_tertiary\_and\_quaternary\_structure': {'Playcount': 16, 'TDiffer': [3.0], 'Times': [1387365502.0, 1387365504.0, 1387365507.0, 1387365510.0, 1387365513.0, 1387365531.0, 1387365533.0, 1387365621.0, 1387365643.0, 1387365654.0, 1387365696.0, 1387365698.0, 1387365700.0, 1387365708.0, 1387365710.0, 1387365713.0]}, 'i4x-MITx-7\_00x\_UMass-video-Biological\_applications\_in\_forensics\_history\_and\_agriculture': {'Playcount': 2, 'TDiffer': [2.0], 'Times': [1387355679.0, 1387355681.0]

Notes: I wanted to add a “while” loop but it never broke example:

else:

*time\_list.append(times)*

*videoDict["Playcount"] = len(time\_list)*

*timecount = len(time\_list)*

*videoDict["TDiffer"] = []*

*while timecount > 1:*

*T1 = timecount - 1*

*T2 = timecount - 2*

*secdiff = time\_list[T1] - time\_list[T2]*

*diff\_list = videoDict["TDiffer"]*

*diff\_list.append(secdiff)*

*timecount=timecount-1*

*else:*

*break*

*userVideoDataDict = usersDict[videoName]*

Update:

Program: diffinplaysf13.py

Works grabs all the time differences in a list

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

time\_list= videoDict["Times"]

time\_list.append(times)

videoDict["Playcount"] = 1

usersDict[videoName] = videoDict

else:

time\_list.append(times)

videoDict["Playcount"] = len(time\_list)

timecount = len(time\_list)

videoDict["TDiffers"] = []

while timecount > 1:

T1 = timecount - 1

T2 = timecount - 2

secdiff = time\_list[T1] - time\_list[T2]

diff\_list = videoDict["TDiffers"]

diff\_list.append(secdiff)

timecount=timecount-1

userVideoDataDict = usersDict[videoName]

print data

example: {'ousham13': {'i4x-MITx-7\_00x\_UMass-video-Amazing\_proteins\_tertiary\_and\_quaternary\_structure': {'Playcount': 16, 'TDiffers': [3.0, 2.0, 8.0, 2.0, 2.0, 42.0, 11.0, 22.0, 88.0, 2.0, 18.0, 3.0, 3.0, 3.0, 2.0], 'Times': [1387365502.0, 1387365504.0, 1387365507.0, 1387365510.0, 1387365513.0, 1387365531.0, 1387365533.0, 1387365621.0, 1387365643.0, 1387365654.0, 1387365696.0, 1387365698.0, 1387365700.0, 1387365708.0, 1387365710.0, 1387365713.0]}, 'i4x-MITx-7\_00x\_UMass-video-Biological\_applications\_in\_forensics\_history\_and\_agriculture': {'Playcount': 2, 'TDiffers': [2.0], 'Times': [1387355679.0, 1387355681.0]}

Goals:

- Make Histogram/ Graph and determine time window

- Consider idea of time window being different for every video. (Maybe if they watched a percentage of the video that would count as a view. 5 min gap in a 7 min video should not be the same as a 5min gap in a 30 min video.)

11/9/15

program: allplaysf13toCSV.py

import liblytics

import csv

import time

from sets import Set

videoNames = {}

with open("Video Names 3.csv", "rU") as f:

reader = csv.reader(f)

for row in reader:

videoNames[row[0]] = row[1] #cannot remember how this line works exactly

usernames = Set()

with open("Names for SPOC.csv", "rb") as f:

reader = csv.reader(f)

for row in reader:

usernames.add(row[1])

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

time\_list= videoDict["Times"]

time\_list.append(times)

videoDict["Playcount"] = 1

playCount = videoDict["Playcount"]

usersDict[videoName] = videoDict

else:

time\_list.append(times)

videoDict["Playcount"] = len(time\_list)

playCount = videoDict["Playcount"]

userVideoDataDict = usersDict[videoName]

f = open("Testing Plays.csv", "w")

f.write("username,")

for cleanName in sorted(videoNames):

f.write(cleanName)

f.write(",")

f.write(" \n")

for username in sorted(usernames):

f.write(username)

f.write(",")

for cleanName in sorted(videoNames):

if videoNames[cleanName] in usersDict:

for Playcount in videoDict:

f.write(str(playCount))

print playCount

else:

f.write("0,")

f.write(" \n")

f.close()

Failed to work properly, gives endless “2” as the playCount, which are interspersed with “0” in the csv file. Every single user has the same playCount for each video.

Goals: Have the playCount accurately displayed for each user. First step is to either vary the playCount for videos and have it be the same for each user or to vary the playCount for the videos vary between users. –Rachel

11/18/15

program: winplays.py

Timewindow was made at the bottom

import liblytics

import time

from sets import Set

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

time\_list= videoDict["Times"]

time\_list.append(times)

videoDict["Playcount"] = 1

usersDict[videoName] = videoDict

else:

time\_list.append(times)

videoDict["Playcount"] = len(time\_list)

timecount = len(time\_list)

videoDict["TDiffers"] = []

while timecount > 1:

T1 = timecount - 1

T2 = timecount - 2

secdiff = time\_list[T1] - time\_list[T2]

diff\_list = videoDict["TDiffers"]

diff\_list.append(secdiff)

timecount=timecount-1

if secdiff<30: # Number is the time window

videoDict["Playcount"] = videoDict["Playcount"] -1

userVideoDataDict = usersDict[videoName]

print data

Goal:

-Play with different time intervals: 1min 2min 3min 4min 5min 6min 7min 8min 9min 10min

- README file

-histograms

12/2/15

Looking at Rachel’s program for printing out the data, we saw that the numbers in the excel file did not match the data made when winplaysf13.py was ran. We eventually saw that the print program was using Prof. Whites version of printing the data (that was on top of the printing to CSV code). Comparing the raw data for greping for Lydia (1), we found that there were only 9 plays (which correlated with Prof. Whites program).

(1) Grepped for: grep Lydia tracking\_700x\_UMass\_\_Fall\_2013.log |grep play\_video |grep An\_overview\_of\_evolution\_and\_a\_comparison\_of\_prokaryotic\_and\_eukaryotic\_cells > lydiaraw.txt

Why?: Lydia was the random person chosen to check the data.

File name: lydiaraw.txt

Plan of action:

1. Modifiy time difference and window programs with Prof. White’s play program code
2. Use that data to make excel file using Rachel’s program

Program name: diffinplaysf13final.py

We tried adding Brian’s program to the difference program, but there are some problems with the playcount. Check the text file to see what it gives:

Files: testdiffinal.txt

Plan of action:

1. Fix time difference program

12 wgaf

import liblytics

import csv

import time

from sets import Set

videoNames = {}

with open("Video Names 3.csv", "rU") as f:

reader = csv.reader(f)

for row in reader:

videoNames[row[0]] = row[1] #cannot remember how this line works exactly

usernames = Set()

with open("Names for SPOC.csv", "rb") as f:

reader = csv.reader(f)

for row in reader:

usernames.add(row[3])

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

videoDict["Playcount"] = 1

usersDict[videoName] = videoDict

videoDict = usersDict[videoName]

videoDict["Times"].append(times)

videoDict["Playcount"] = len(videoDict["Times"])

usersDict[videoName] = videoDict

data[username] = usersDict

#calculate list of differences

filtered\_playtime\_list = []

for username in sorted(usernames):

if username in data:

usersDict = data[username]

for cleanName in sorted(videoNames):

if videoNames[cleanName] in data[username]:

videoDict = usersDict[videoNames[cleanName]]

playtime\_list = videoDict["Times"]

T1 = 0 # start at 0 so you automatically include the first play\_video event

for playtime in sorted(playtime\_list):

T2 = playtime

diff = T2 - T1

if diff > 0:

filtered\_playtime\_list.append(diff)

T1=T2

videoDict["filtered\_play\_list"]=filtered\_playtime\_list

12/10/15

Working Time window to CSV!

Important notes= setting time window to zero eliminates all the repeats ☺

Program name: draftwin.py

Accompanying csv: Real filtered f13.csv

import liblytics

import csv

import time

from sets import Set

videoNames = {}

with open("Video Names 3.csv", "rU") as f:

reader = csv.reader(f)

for row in reader:

videoNames[row[0]] = row[1] #cannot remember how this line works exactly

usernames = Set()

with open("Names for SPOC.csv", "rb") as f:

reader = csv.reader(f)

for row in reader:

usernames.add(row[3])

data={} #first dict with username (key) : second dict (value)

def parseEventText(eventLine):

line = eventLine.replace("{","")

line = line.replace("}","")

units = line.split(",")

result = {}

for unit in units:

pieces = unit.split(":")

result[pieces[0].replace('"',"")] = pieces[1].replace('"',"")

return result

for line in liblytics.read\_log\_file("tracking\_700x\_UMass\_\_Fall\_2013.log.gz"): #Reads line in log file

if (line["event\_type"] == "play\_video"): # Grabs only play\_videos

username=line["username"]

t = time.strptime(line['time'].split('+')[0], "%Y-%m-%dT%H:%M:%S.%f")

times = time.mktime(t)

#first loop creates first dict

if username != "":

videoName = parseEventText(line["event"])["id"]

if username not in data:

data[username] = {}

usersDict = data[username]

#creates third dict with playCount and Times (keys) and their values

if videoName not in usersDict: #second loop creates third dict

videoDict = {}

videoDict["Times"] = []

videoDict["Playcount"] = 1

usersDict[videoName] = videoDict

videoDict = usersDict[videoName]

videoDict["Times"].append(times)

videoDict["Playcount"] = len(videoDict["Times"])

usersDict[videoName] = videoDict

data[username] = usersDict

#calculate list of differences

filtered\_playtime\_list = []

for username in sorted(usernames):

if username in data:

usersDict = data[username]

for cleanName in sorted(videoNames):

if videoNames[cleanName] in data[username]:

videoDict = usersDict[videoNames[cleanName]]

filtered\_playtime\_list = []

playtime\_list = videoDict["Times"]

LPT = 0 # start at 0 so you automatically include the first play\_video event

for playtime in sorted(playtime\_list):

T2 = playtime

diff = T2 - LPT

if diff > 0:

filtered\_playtime\_list.append(T2)

LPT=T2

videoDict["Filtered\_play\_list"]=filtered\_playtime\_list

videoDict["Filtered Plays"]= len(videoDict["Filtered\_play\_list"])

# Prints to CSV

f = open("Real Filtered f13.csv", "w")

f.write("username,")

for cleanName in sorted(videoNames):

f.write(cleanName)

f.write(",")

f.write(" \n")

##for username in sorted(data):

##if username in usernames:

for username in sorted(usernames):

if username in data:

f.write(username)

f.write(",")

for cleanName in sorted(videoNames):

if videoNames[cleanName] in data[username]:

f.write(str(data[username][videoNames[cleanName]]["Filtered Plays"]))

f.write(",")

else:

f.write("0,")

f.write(" \n")

f.close()

1/29/16

* Draftwin.py name changed to windowf13.py and README created
* Github repository and currently tracking Research log created “RodneyBruno”
* Goal: Use list of video lengths, set a time window (20-40min using windowf13.py), Compare # of plays (Both f13 and f14)

2/2/16

* Avg. Length of Videos for F13 were 678.81sec (11.31 min)
* Checking 20min (1200), 25min (1500), 30min (1800),35min (2100) and 40min(2400)
* Comparing plays
* Need access to Fall 13 on edx
* (Are there different neames to access F14 data )
* Fall 2013 Comparison of Time widows made called
  + CSV name: F13 Rime Window Comparision
  + 20 mins (1200 sec) is an appropriate time window
* Note: Brian white F13 are not logged as they were

2/5/16

* Attempted to modify windowf14.py to work for Fall 2013 (currently exact copy of windowf13.py)
* Grabbed F14 files from main computer
* ~~Address issues with zip file reading~~

2/16/16

* Changed lining on Abstract for conference
* Modified all diff plays for fall 14
* Using log file from one day, it was identified that there is a problem with the big log file for Fall 2014
* ~~Overview: new log file needed for windowf14.py and ListAllDiff14.py~~
  + Example of result of one day in “Diff list for 9.17.14.csv”

2/17/16

* Problems with both programs were fixed by “re-gziping” the log file for Fall 2014
* Want to add mins 5, 10, 15min to analysis to reinforce choice of having 20min time window
* “Monster List Diff F14.csv” made to make histogram of data for distribution of Fall 2014
* Next: Re-analyze F13 window with added 5, 10, 15 min and analyze F14 (windowf1\_.py programs) 22

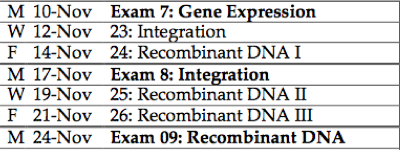
2/22/16

* Fix time window output for F13/F14 (and remember that time window is in seconds !!!!)
  + Redo comparisons for windows 5-15 for F13 and 0-40 for F14

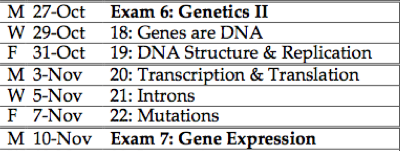
3/8/16

* Worked last two weeks on seek video program with praj and reviewing Casey’s program on duedates
* Created file called “practiceduewindowf13.py” (which originally was an exact copy of the after due date counter called “duewindowf13.py”)
  + Goal to double check output of code times > dueDates[videoName]
    - Found that the times match due dates
    - Data in file called “practiceresultofduedate.txt”
* Next Goal: Clean up due date programs as they contain some repeated codes for example it has two import lyblitics ect. (should be simple)

3/22/16

* Reviewed Casey’s “Due.Date.Compare.xlsx”
  + The program looked at the total views before and after the due date
  + These Sessions were seen more after the due date:
    - Lesson 24 (Recombinant DNA 1) Due Nov. 14, 2014
    - Lesson 25 (Recombinant DNA 2) Due Nov. 19, 2014
    - Syllabus:

|  |
| --- |
| 24.2-Overview  24.3-CuttingAndPastingDNA  24.4-Vectors  24.5-TransformationAndSelection  24.6-ShaggyDog  25.1-GelElectrophoresis  25.2-DNASequencing  25.3-DNASequencingImplementation |

* + And these but it seems like more of an isolated situation:
    - Lesson 19 (DNA Structure and Replication) Due Oct. 31 15:00
    - Syllabus:

|  |
| --- |
| 19.2-DetailsOfDNAReplication  19.3-DNAReplicationEnzymes  19.4-DNAReplicationFidelity |

Next Steps: Could look at performance on exam 9: Recombinant DNA

What does this mean about that topic?

Could look at the specific date that most are watching it again: is it right before doing the part 2 or 3

The fact that these are sequential could play an important factor

Question: Is this Fall 2014?!