CS432 Web Science: Assignment 7

Finished on April 24, 2020

Dr. Michele C. Weigle

Hannah Holloway
hhollowa@odu.edu

Contents

List of Figures

Listings

Problem 1

Question

Generate a list of 100 popular accounts on Twitter. The accounts must be verified, have > 10,000 followers, and have > 5000 tweets. See GET users/lookup and User object for details on obtaining this information for a set of accounts. You may also generate this information manually by visiting individual account pages. For example:

https://twitter.com/weiglemc - not verified, 414 followers, 2189 tweets - don't include https://twitter.com/WNBA - verified (blue checkmark), 615,000+ followers, 69,000+ tweets - could include

Save the list of accounts (screen_names) in a text file (one per line) and upload to your GitHub repo.

Download 200 tweets from the 100 accounts. See GET statuses/user_timeline for details. Note that you may receive fewer than 200 tweets in a single API call due to deleted or protected tweets. It's OK as long as you get somewhere close to 200 tweets for each account. (I don't want to you have to make more than one API call per account.)

Save the responses received from the 100 accounts to your GitHub repo. It can all be in a single file or a separate file for each account. Since this is an intermediate file, the format is up to you.

Answer

In order to get a list of 100 popular accounts on Twitter containing all the requirements specified, I used GET friends/list and appended it wrote the data to a text file. The full script is shown below in Listing 1.

```
import tweepy
from tweepy import *
import time
import os

CONSUMER_KEY = ""
CONSUMER_SECRET = ""

OAUTH_TOKEN = ""
OAUTH_TOKEN_SECRET = ""

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN,OAUTH_TOKEN_SECRET)
api=tweepy.API(auth)

fdata=open('thepopulars.txt','w')
user=api.get_user('SenWarren')
fdata.write('SenWarren'+'\n')

for user in tweepy.Cursor(api.friends, screen_name="SenWarren",count=100).items():
    #users.append(user)
    fdata.write(user.screen_name+',\n')
fdata.close()
```

Listing 1: Friend Scraper Script

text
RT @gabbysalinas: Like @ewarren says, “Let’s dream big, fight hard, and win”!!! https://t.co/MwNAzgXBES
Stay home! But when you can’t stay home—stay safe. Sending strength from my family to yours. https://t.co/k97jzECib6
RT @nbcsnl: **Flipped the switch**. https://t.co/t8qHcGRUhY
I know that we will one day elect a woman to the White House. https://t.co/logAh3YEFC
I believe in the America that we can build together. Thank you, Detroit! https://t.co/vT9rnnnZNP
Trickle down economics has been a failure. We need to build an economy from the grassroots up. https://t.co/6k34XJKmfl
I believe in the America that we can build together. https://t.co/i7BnOZG5aR
Hope over fear. Courage over cynicism. Let’s do this together. Vote: https://t.co/5tQ5A1gVhA https://t.co/rRBLAfugdo
RT @Nick_Offerman: I wanted to be a winner so I geared up and went to Pawnee to vote #VoteWarren https://t.co/ii83TJtkcm
RT @AyannaPressley: Outwork. Out-organize. Outlast. https://t.co/mQt4hS5SbQ
This is our moment. Make your voice heard. Vote today. https://t.co/5tQ5A0Zkq2 https://t.co/lBDENmCjpB

Figure 1: Sample of tweets extracted

Sample of some of the users I retrieved:

```
SenWarren
RepGwenMoore
RepRonKind
repmarkpocan
RepDennyHeck
RepAdamSmith
RepKimSchrier
RepJayapal
RepDerekKilmer
RepRickLarsen
```

Next, I needed to download 200 tweets from each of the 100 accounts.The script I used for this can be viewed below in Listing 2. I ended up appending all the tweets to the same .csv file. Sample list of tweets from the .csv can be viewed below in Figure 1.

```
import tweepy
import json
import time
from tweepy import OAuthHandler
import tweepy #https://github.com/tweepy/tweepy
import csv
import sys


CONSUMER_KEY = "YwekEH9UrlXUFKUH2XmImE681"
CONSUMER_SECRET = "NWaB0jN5HaQ3f9VqCrMhP2nP8154KGXoPeDxZk6TIOVptAxErb"

OAUTH_TOKEN = "4860544225-qbjIQvrGlrj493eIHAjNu2OH0rdrHlM94XMHe1x"
OAUTH_TOKEN_SECRET = "vfAc4sJwbWExBjrMZiMqRMuknTzbl2le55DKX5gGYOOXR"

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN,OAUTH_TOKEN_SECRET)
api=tweepy.API(auth)

def get_all_tweets(screen_name):



        #initialize a list to hold all the tweepy Tweets
        alltweets = []

        #make initial request for most recent tweets (200 is the maximum allowed count)
        new_tweets = api.user_timeline(screen_name = screen_name,count=1)

        #save most recent tweets
        alltweets.extend(new_tweets)

        #save the id of the oldest tweet less one
        oldest = alltweets[-1].id - 1

        #keep grabbing tweets until there are no tweets left to grab
        while len(new_tweets) > 0:
                print "getting tweets before %s" % (oldest)

                #all subsequent requests use the max_id param to prevent duplicates
                new_tweets = api.user_timeline(screen_name = screen_name,count=200,max_id=
                    oldest)

                #save most recent tweets
                alltweets.extend(new_tweets)

                #update the id of the oldest tweet less one
                oldest = alltweets[-1].id - 1

                print "...%s tweets downloaded so far" % (len(alltweets))

        #go through all found tweets and remove the ones with no images
        outtweets = [] #initialize master list to hold our ready tweets
        for tweet in alltweets:
                #not all tweets will have media url, so lets skip them
                try:
                        print tweet.entities['media'][0]['media_url']
                except (NameError, KeyError):
                        #we dont want to have any entries without the media_url so lets do
                            nothing
                        pass
                else:
                        #got media_url - means add it to the output
                        outtweets.append([tweet.text.encode("utf-8")])

        #write the csv
        with open('%s_tweets.csv' % screen_name, 'wb') as f:
```

```
                writer = csv.writer(f)
67              writer.writerow(["text"])
                writer.writerows(outtweets)

        pass


72
if __name__ == '__main__':
        #pass in the username of the account you want to download
        get_all_tweets("ewarren")
```

Listing 2: Tweet Scraper Script

Problem 2

Question

Generate an account-term matrix from the accounts' tweets.

Using the responses from Q1, extract the text from each tweet to generate terms. Remove any URIs in the tweets, but keep regular text and hashtags as terms. Limit the number of terms to the most "popular" (i.e., frequent) 1000 terms, this is after the criteria on p. 32 (chapter 3 PCI book) (slide 11 - Week 10) has been satisfied.

Save the terms for each account in a file and upload to your GitHub repo. It can all be in a single file or a separate file for each account. Since this is an intermediate file, the format is up to you.

In the account-term matrix, the account screen_name is the account identifier and should be start each row of the matrix. Use the (max 1000) terms for the columns of the matrix. The values are the frequency of occurrence. Essentially you are replicating the format of the "blogdata.txt" file included with the PCI book code.

Save the matrix in a text file (either tab-separated like blogdata.txt or comma-separated) and upload to your GitHub repo.

Answer

Next, I found the account term matrix using my script "generateaccountvector.py" similar to the Programming Collective Intelligence text. I used the feedparse library to parse through the twitter status URI. I limited the word count to 1000. Each row in the account term matrix represents a screen_name and and each column represents a specific word. Every cell represents the number of times a word is present in the the screen_name's tweets. I also used stopwords to prevent irrelevent data from joining the account term matrix. The account term matrix consists of 100 rows (screen_names) and 1,000 columns (terms). The script used to create the account term matrix is displayed in Listing 3.

I chose Senator Warren, Representative Sean Maloney and Representative John Lewis as my 3 twitter account choices. The 5 nearest neighbors for "ewarren" can be found in Figure 2. The 5 nearest neighbors for "RepSeanMaloney" can be found in Figure 3. The 5 nearest neighbors for "repjohnlewis" can be found in Figure 4.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	user	voting	found	young	light	real	ensure	kind	heart	hard	lot	friends	high	left	track	set
2	ewarren	12	7	22	3	7	6	3	3	4	10	6	13	5	13	4
3	RepGwenMoore	9	3	6	2	1	8	18	2	14	9	1	0	3	6	2
4	RepRonKind	1	0	4	0	4	11	0	1	4	1	0	1	0	11	1
5	repmarkpocan	0	0	1	1	0	1	1	7	0	0	0	0	1	0	0
6	RepDennyHeck	23	10	2	7	26	29	27	2	12	36	11	22	6	27	27
7	RepAdamSmith	2	0	11	1	2	0	2	1	3	1	2	2	1	2	2
8	RepKimSchrier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	RepJayapal	0	0	1	0	0	0	0	0	0	2	1	2	1	0	0
10	RepDerekKilmer	0	1	6	1	6	10	4	1	4	10	1	3	3	6	1
11	RepRickLarsen	4	26	1	10	4	3	2	0	9	2	5	10	9	8	16
12	RepDelBene	4	7	6	0	7	13	3	1	4	6	2	5	6	11	1
13	GerryConnolly	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	RepWexton	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	RepWexton	1	2	0	0	0	2	0	0	1	1	3	2	2	1	6
16	RepSpanberger	0	2	12	4	2	1	0	4	0	0	0	1	4	3	0
17	RepMcEachin	2	1	3	4	6	0	7	6	4	2	2	2	3	6	1
18	BobbyScott	1	2	6	5	2	0	1	5	0	1	7	2	1	1	8
19	RepElaineLuria	0	1	2	6	1	1	1	6	1	1	0	3	0	0	0
20	StaceyPlaskett	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
21	PeterWelch	19	2	11	9	8	32	44	4	12	27	6	8	12	5	13
22	RepBenMcAdams	0	1	1	2	1	0	0	0	1	0	1	1	0	8	0
23	RepLloydDoggett	4	2	8	0	1	4	2	3	3	13	4	4	9	2	3

Figure 2: Sample of Account Term Matrix

```
from __future__ import division
import glob
from nltk.corpus import words
from nltk.corpus import stopwords
from nltk import *
import re
import nltk
import os
nltk.download('words')
import csv


words = words.words()
# Bring in the default English NLTK stop words
stoplist = stopwords.words('english')

# Define additional stopwords in a string
additional_stopwords = "com https http img osx a's accordingly again allows also amongst
    anybody anyways appropriate aside available because before below between by can't
    certain com consider corresponding definitely different don't each else et everybody
    exactly fifth follows four gets goes greetings has he her herein him how i'm immediate
    indicate instead it itself know later lest likely ltd me more must nd needs next none
    nothing of okay ones others ourselves own placed probably rather regarding right saying
    seeing seen serious she so something soon still t's th that theirs there therein they'd
    third though thus toward try under unto used value vs way we've weren't whence whereas
    whether who's why within wouldn't you'll yourself according afterwards allow already
    among any anyway appreciate as at became been believe better but can causes co
    consequently contains currently didn't doing during either especially every ex few
    following forth get go gotten hardly having hence hereby hi hopefully i'll ignored
    indeed insofar isn't its kept lately less liked looks maybe might much namely need new
    non not obviously ok one other ours overall perhaps presumably qv reasonably
    respectively say see seems sent shall six someone somewhere specifying sure tends thanx
    their thence therefore they think those thru took truly un until use usually viz wasn't
    we're were when whereafter wherever who whose with would you'd yours able across against
```

```
    almost although an anyhow anywhere are ask away become beforehand beside beyond c'mon
    cannot certainly come considering could described do done edu elsewhere etc everyone
    example first for from getting going had hasn't he's here hereupon himself howbeit i've
    in indicated into it'd just known latter let little mainly mean moreover my near neither
     nine noone novel off old only otherwise out particular please provides rd regardless
    said says seem self seriously should some sometime sorry sub take than that's them there
    's theres they'll this three to towards trying unfortunately up useful various want we
    welcome what whenever whereby which whoever will without yes you're yourselves about
    actually ain't alone always and anyone apart aren't asking awfully becomes behind
    besides both c's cant changes comes contain couldn't despite does down eg enough even
    everything except five former further given gone hadn't have hello here's hers his
    however ie inasmuch indicates inward it'll keep knows latterly let's look many meanwhile
     most myself nearly never no nor now often on onto ought outside particularly plus que
    re regards same second seemed selves seven shouldn't somebody sometimes specified such
    taken thank thats themselves thereafter thereupon they're thorough through together
    tried twice unless upon uses very wants we'd well what's where wherein while whole
    willing won't yet you've zero above after all along am another anything appear around
    associated be becoming being best brief came cause clearly concerning containing course
    did doesn't downwards eight entirely ever everywhere far followed formerly furthermore
    gives got happens haven't help hereafter herself hither i'd if inc inner is it's keeps
    last least like looking may merely mostly name necessary nevertheless nobody normally
    nowhere oh once or our over per possible quite really relatively saw secondly seeming
    sensible several since somehow somewhat specify sup tell thanks the then thereby these
    they've thoroughly throughout too tries two unlikely us using via was we'll went
    whatever where's whereupon whither whom wish wonder you your a b c d e f g h i j k l m n
     o p q r s t u v w x y z lo en png jps jpg eh ext tq lj pbs rt"

stoplist += additional_stopwords.split()

total=[]
def individual_count(filename,tokens,counter):
    file_location="./texts/"+filename
    counter[0]=filename
    word=""
    with open(file_location,"r") as obj:
        for i in obj.read():
                content = obj.read()
                only_words = re.sub("[^a-zA-Z]", " ", content) # Remove anything that isn't
                    a 'word'
                no_single = re.sub(r'(?:^| )\w(?:$| )', ' ', only_words).strip()
                no_double = re.sub(r'(\b\w{1,3}\b)', ' ', only_words).strip()
                text = re.sub(r'(?:(?:http|https):\/\/)?([-a-zA-Z0-9.]{2,256}\.[a-z]{2,4})\b
                    (?:\/[-a-zA-Z0-9@:%_\+.~#?&//=]*)', ' ', only_words).strip()
                total.append(text)
                total.append(no_single)
                total.append(no_double)
                if i is ",":
                        i=tokens.index(word)
                        counter[i]+=1
                        total[i]+=1
                        word=""
                else:
                        word+=i
                writer(counter)

def get_overall_tokens():
    tokens=["filename"]
    for i in files:
        file_location="./texts/"+i
        column=[]
        word=""
        with open(file_location,"r") as obj:
            for i in obj.read():
                if i is ",":
                    if word not in tokens:
                        tokens.append(word)
                    word=""
                else:
```

```
                    word+=i
    writer(tokens)
    return tokens

def writer(list):
    with open('./matrices/userdata.csv', mode='a') as obj:
        obj_w = csv.writer(obj, quotechar='"', quoting=csv.QUOTE_MINIMAL)
        obj_w.writerow(list)



if __name__=="__main__":
    files=os.listdir("./texts/")
    tokens=get_overall_tokens()
    counter=[]
    for i in tokens:
        counter.append(0)
    counter[0]="total"
    total.extend(counter)
    for i in range(len(files)):
        individual_count(files[i],tokens,counter[:])
    print(total)
    writer(total)
```

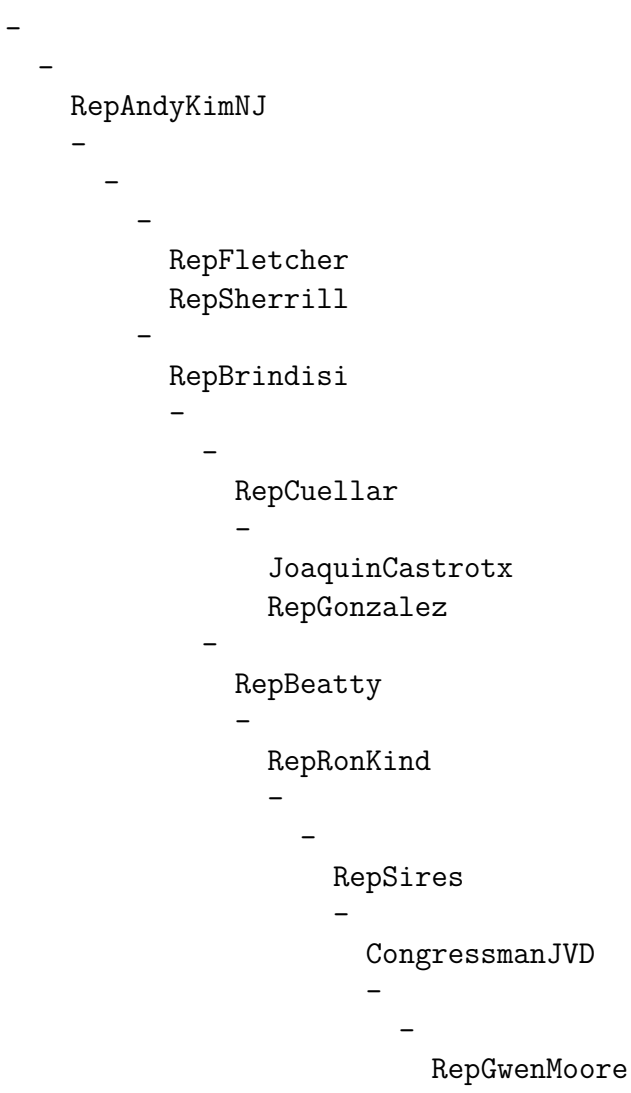
Listing 3: Tweet Scraper Script

Problem 3

Question

Create an ASCII dendrogram and a JPEG dendrogram that uses hierarchical clustering to cluster the most similar accounts (see slides 21 23 - Week 10). Include the JPEG in your report and upload the ASCII file to GitHub (it will be too unwieldy for inclusion in the report).

Answer



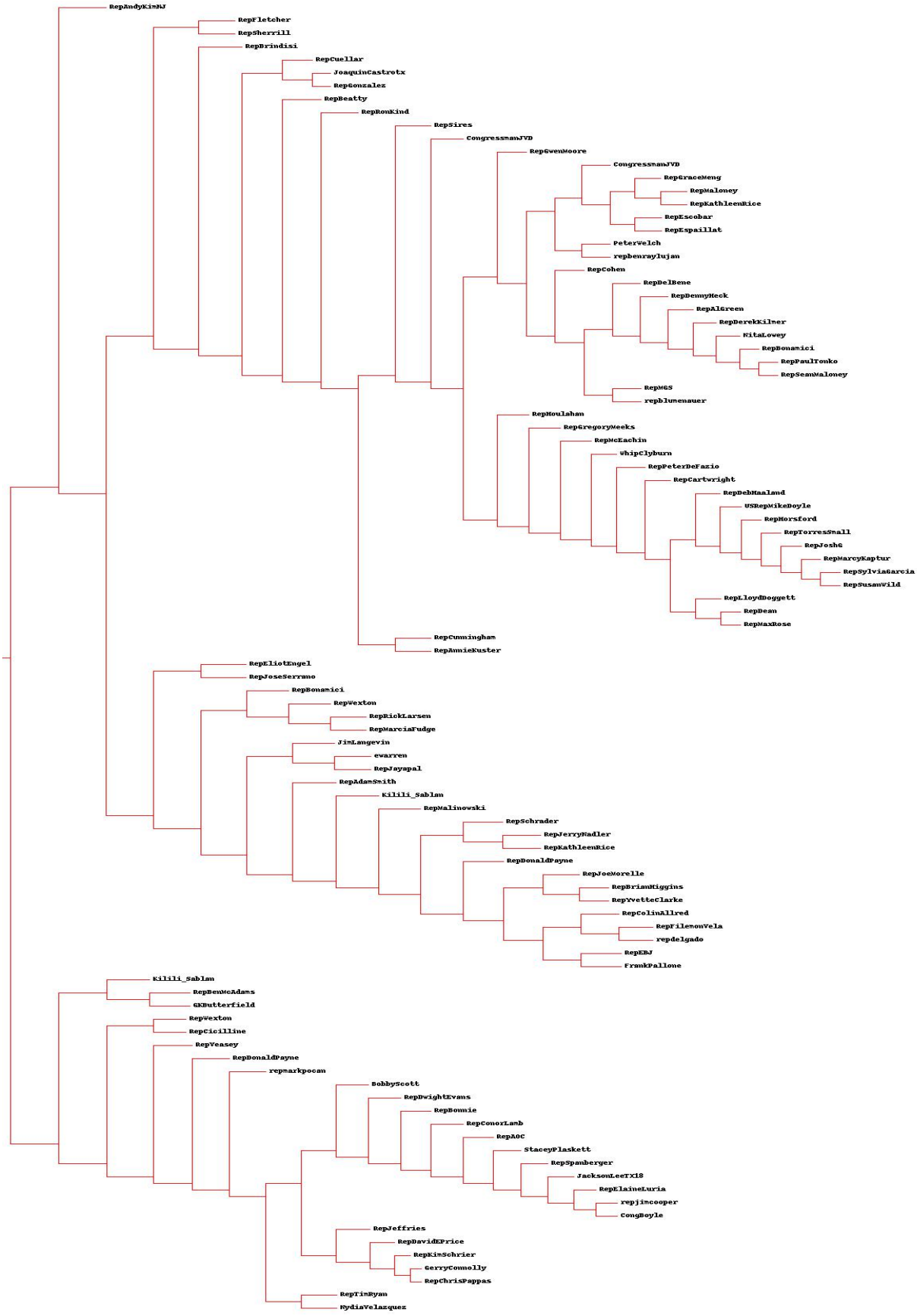


Figure 3: Dendogram clustering most similar accounts

Figure 4: Close up of Dendogram clustering most similar accounts

Problem 4

Question

Cluster the accounts using k-Means, using k=5,10,20 (see slide 37 - Week 10). Print the accounts in each cluster, for each value of k. How many iterations were required for each value of k?

Answer

I used my script clusters.py which is based off of the code from Programming Collective Intelligence. The number of iteration for k=5 is 6. The number of iterations for k=10 is 7 and the number of iterations for k=20 is 6.

```
import clusters
blog,words,data=clusters.readfile('userdata.txt')

print "For k=5"
kclust=clusters.kcluster(data, k=5)
for i in range(0,5):
    print "accounts in cluster"+' '+str(i+1)+'-------'
    for r in kclust[i]:
        print blog[r]
    print '\n'
print "For k=10"
kclust=clusters.kcluster(data, k=10)
for i in range(0,10):
    print "accounts in cluster"+' '+str(i+1)+'-------'
    for r in kclust[i]:
        print blog[r]
    print '\n'

print "For k=20"
kclust=clusters.kcluster(data, k=20)
for i in range(0,20):
    print "accounts in cluster"+' '+str(i+1)+'-------'
    for r in kclust[i]:
        print blog[r]
    print '\n'
```

Listing 4: k-Means cluster Script

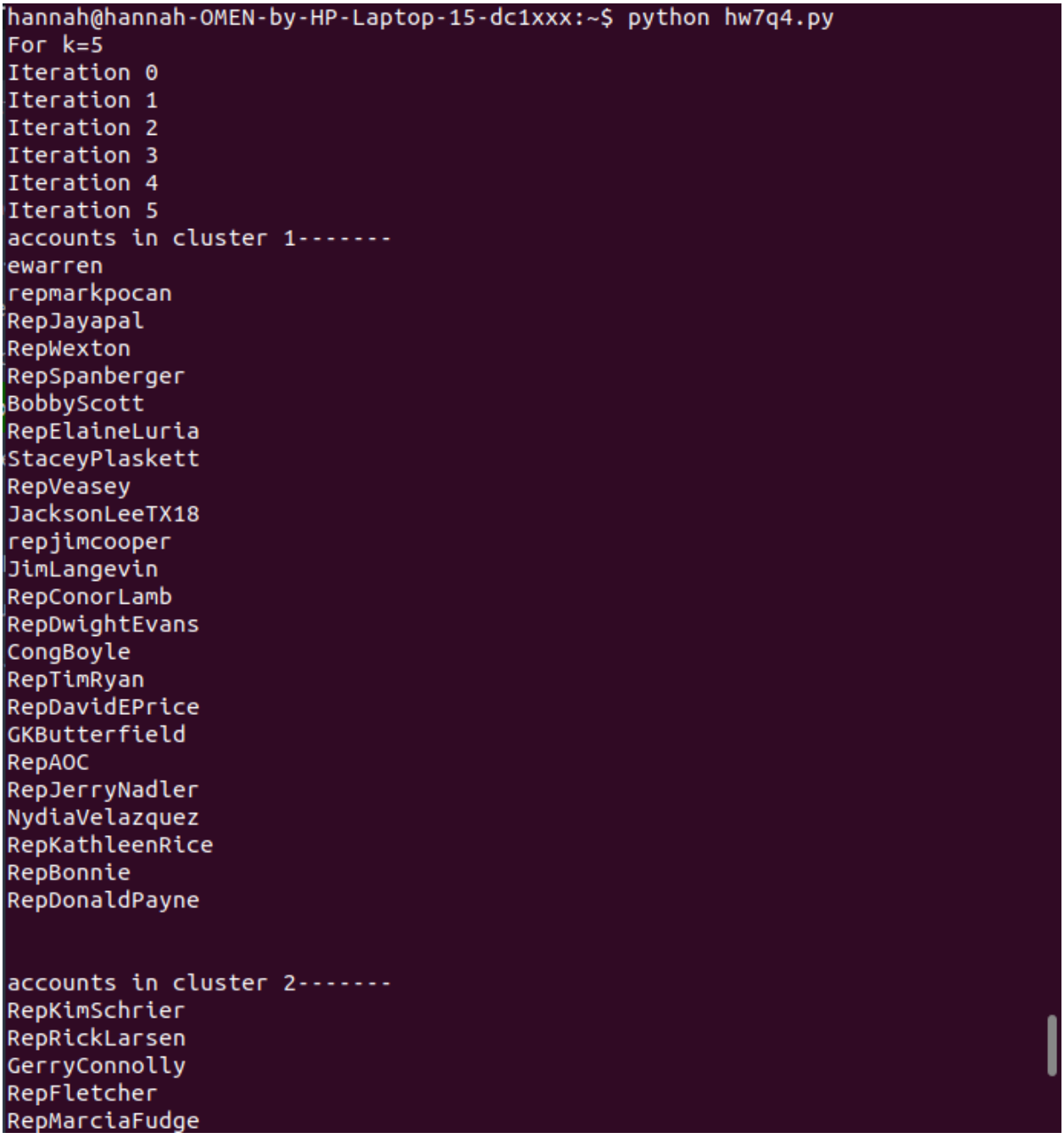


Figure 5: Sample values in each cluster and the number of iterations

Problem 5

Question

Use MDS to create a JPEG of the accounts (see slide 50 - Week 10). Include the JPEG in your report. How many iterations were required?

Answer

I used "clusters.py" as well for this task. I added a line of code to print the number of iterations. The script I used to create the jpeg of accounts is below as well as the jpeg image itself.

```
#!/usr/local/bin/python

import clusters

account,words,data=clusters.readfile('userdata.txt')

coordinates = clusters.scaledown(data)

clusters.draw2d(coordinates, account, jpeg='accounts.jpg')
```

Listing 5: MDS JPEG Creator Script

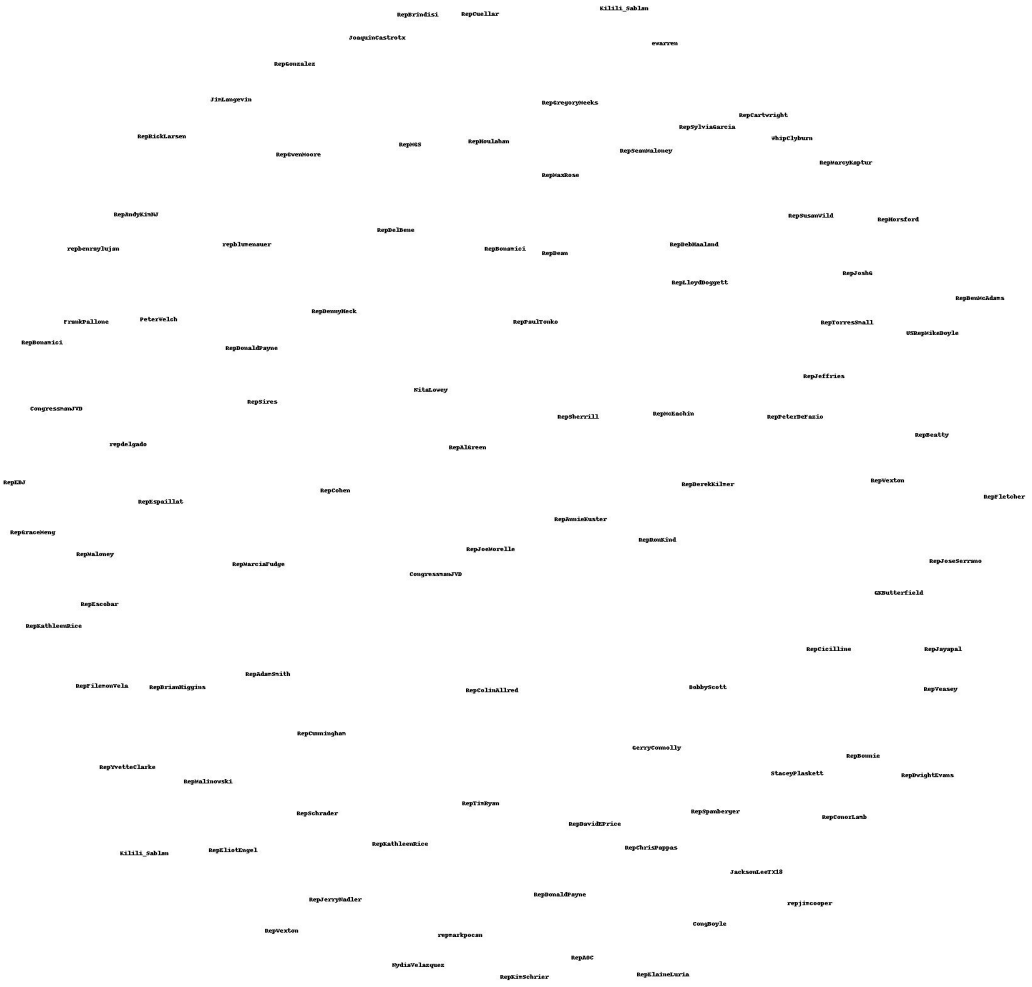


Figure 6: JPEG of accounts

References

[1] https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/ .

[2] http://jurgens.people.si.umich.edu/tutorials .

[3] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-5-50b4e87d9bdd .