RACHEL MCCREARY
CS 432 ASSIGNMENT 8 REPORT

PART 1

In an effort to get a bunch of blogs with hopefully no repeats or offensive material, I first searched Twitter for "my blog" and made a set of usernames from the results, then took each user's website URI, assuming most of them would have their blog on their website (I got this idea, but not the code, here):

generatefeedvector.py × | get_blogs.py × | long_links.py ×

```python
import tweepy
from tweepy import OAuthHandler

access_token =
access_token_se
consumer_key =
consumer_secret

if __name__ == '__main__':

    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)
    search_results = api.search(q="my%20blog", count=250)
    users = []
    blogs = []
    i = 0

    for tweet in search_results:
            if tweet.user.screen_name in users:
                next
            else:
                users.append(tweet.user.screen_name)

    for user in users:
        data = api.get_user(user)
        if (data.url != None):
            if data.url in blogs:
                next
            else:
                blogs.append(data.url)
                print data.url
```

```
https://t.co/yigFI9LbLy
http://t.co/1AFect31hS
https://t.co/j8ZCsI8IoN
https://t.co/zK3OEo8aTY
http://t.co/MtFE1GiYy5
http://t.co/MmgEwKYHs6
https://t.co/53mjyaHknd
https://t.co/okSpMUhcM4
http://t.co/6gRy54fCh4
http://t.co/mFdwguHhEx
http://t.co/NRYFpHRaPT
http://t.co/gpBf5fQBVJ
http://t.co/RpdYzq9tPG
```

I used this for guidance in following the links and getting the long URI, skipping broken links:

generatefeedvector.py | get_blogs.py | long_links.py

```python
from __future__ import unicode_literals, print_function
import ttp
import requests
import urllib2

def follow_shortlinks(shortlinks):
    links_followed = {}
    for shortlink in shortlinks:
        url = shortlink
        request_result = requests.get(url)
        redirect_history = request_result.history
        all_urls = []
        for redirect in redirect_history:
            all_urls.append(redirect.url)
        all_urls.append(request_result.url)
        links_followed[shortlink] = all_urls
    return links_followed

def on_error(self, status):
    pass

if __name__ == '__main__':

    shortlinks = []

    with open('short_links.txt')as f:
        for line in f:
            shortlinks.append(line)

    followed_shortlinks = (follow_shortlinks(shortlinks))

    for link in followed_shortlinks:
        try:
            print (urllib2.urlopen(link).url)
        except urllib2.HTTPError as e:
            print ('\t', e)
```

```
http://morganna-designs.squarespace.com/
http://ec.europa.eu/france/index_fr.htm
http://salessolveeverything.com/
https://www.instagram.com/justjess369/
http://www.audriestorme.com/
http://www.amazon.com/registry/wishlist/17TM87YY2ZF67/ref=cm_sw_r
https://jacquelinechambliss.wordpress.com/
https://annebellesbooks.wordpress.com/
http://www.songofthestitch.com/
http://fineartamerica.com/profiles/spyder-webb.html
http://jamesoneseventeen.blogspot.com/
http://www.garywalker.net/
https://configuroweb.blogspot.com/
http://www.adamchudy.com/
```

After that, I picked through and rid the list of unwanted links. Then, to find the RSS feeds of these websites, I used [this](#) for guidance (StackOverflow question):

generatefeedvector.py × get_rss.py ×

```python
import requests
from bs4 import BeautifulSoup

def get_rss_feed(website_url):
    if website_url is None:
        print ("URL should not be null")
    else:
        source_code = requests.get(website_url)
        plain_text = source_code.text
        soup = BeautifulSoup(plain_text, "lxml")
        for link in soup.find_all("link", {"type" : "application/rss+xml"}):
            href = link.get('href')
            print str(href)

with open('blogs.txt') as f:
    for line in f:
        try:
            print get_rss_feed(line)
        except:
            pass
```

```
http://f-measure.blogspot.com/feeds/posts/default?alt=rss
None
http://ws-dl.blogspot.com/feeds/posts/default?alt=rss
None
https://craftytaramarie.wordpress.com/feed/
https://craftytaramarie.wordpress.com/comments/feed/
None
http://radicalreadsbook.blogspot.com/feeds/posts/default?alt=rss
None
http://eroseweb.co/blog/feed/
http://eroseweb.co/blog/comments/feed/
None
None
http://hiiijaaackie.blogspot.com/feeds/posts/default?alt=rss
```

Note: I'm uploading the full results shown so far to GitHub, but will only put the RSS links in this report.

List of RSS links:

http://f-measure.blogspot.com/feeds/posts/default?alt=rss
http://ws-dl.blogspot.com/feeds/posts/default?alt=rss
https://craftytaramarie.wordpress.com/feed/
http://radicalreadsbook.blogspot.com/feeds/posts/default?alt=rss
http://eroseweb.co/blog/feed/
http://hiiijaaackie.blogspot.com/feeds/posts/default?alt=rss
http://www.notesfromcaroline.com/feeds/posts/default?alt=rss
http://thebusinessgirlsnetwork.com/feed/
http://deenasdays.com/feed/
https://siefkenpublications.wordpress.com/feed/

http://www.thekmprojects.gr/feeds/posts/default?alt=rss
http://www.coldknowledge.com/feeds/posts/default?alt=rss
http://www.gingeybites.com/feeds/posts/default?alt=rss
https://jasminecarlisleblog.wordpress.com/feed/
http://chinenyeugonna.com/feed/
http://theguyliner.com/feed/
http://edufisbil.blogspot.com/feeds/posts/default?alt=rss
http://ericabachelor.com/feed/
http://labellablog.co/feed/
https://lillyteardrop.wordpress.com/feed/
http://kreshimir.com/feed/
http://momsncharge.com/feed/
http://sami-spoon.blogspot.com/feeds/posts/default?alt=rss
http://pedrothedagger.tumblr.com/rss
https://indiewatchsite.wordpress.com/feed/
http://justbellefashionblog.blogspot.com/feeds/posts/default?alt=rss
http://www.jessicafoley.ca/feed/
https://whenyousaidtulips.wordpress.com/feed/
http://whimsicaljoy.com/?feed=rss2
http://doginasweatershowreviews.blogspot.com/feeds/posts/default?alt=rss
http://justthatgirlonline.blogspot.com/feeds/posts/default?alt=rss
http://www.audriestorme.com/feed.xml
https://jacquelinechambliss.wordpress.com/feed/
https://annebellesbooks.wordpress.com/feed/
http://jamesoneseventeen.blogspot.com/feeds/posts/default?alt=rss
https://configuroweb.blogspot.com/feeds/posts/default?alt=rss
http://www.adamchudy.com/feed/
http://www.papersouldesign.co.uk/feed.xml
http://apilgrimontheroad.blogspot.com/feeds/posts/default?alt=rss
http://theonionfield.blogspot.com/feeds/posts/default?alt=rss
http://fitlifefranticfree.blogspot.com/feeds/posts/default?alt=rss
http://drgperformancesolutions.com/feed
http://theidealcopy.blogspot.com/feeds/posts/default?alt=rss
http://www.matthewmegyese.com/feed.xml
http://floorshimezipperboots.blogspot.com/feeds/posts/default?alt=rss
http://stewartkirby.blogspot.com/feeds/posts/default?alt=rss
http://thepioneerwoman.com/feed/
http://thepioneerwoman.com/life-and-style/feed/
http://emmersonbloede.blogspot.com/feeds/posts/default?alt=rss
http://mcomv2.blogspot.com/feeds/posts/default?alt=rss
http://cookeatrun.com/feed/
http://toxicbreedsfunhouse.blogspot.com/feeds/posts/default?alt=rss
http://skinnyshoes.blogspot.com/feeds/posts/default?alt=rss
http://fivescrambledeggs.blogspot.com/feeds/posts/default?alt=rss
http://aninconsistentscrapper.blogspot.com/feeds/posts/default?alt=rss
http://mondaywakeup.blogspot.com/feeds/posts/default?alt=rss
http://didnotchart.blogspot.com/feeds/posts/default?alt=rss
http://robynpeterman.com/feed/
http://robynpeterman.com/blog/feed/
https://fromauniqueperspective.wordpress.com/feed/
http://www.adventuresofalondonkiwi.com/feeds/posts/default?alt=rss
http://itslostitsfound.blogspot.com/feeds/posts/default?alt=rss
http://www.kevinkruse.com/feed/
http://www.kevinkruse.com/blog/feed/
https://thousandscarsblog.wordpress.com/feed/
http://henrycleaves.blogspot.com/feeds/posts/default?alt=rss
http://www.danjjroberts.co.uk/feed/

https://siefkenpublications.wordpress.com/feed/
https://justjeanine.wordpress.com/feed/
http://www.careercheatcodes.com/feed/
http://extremeways.blog.jp/index.rdf
https://chasrad.wordpress.com/feed/
http://rilphly.com/feed/
https://sardecoo.wordpress.com/feed/
http://iguacen.com/rss
http://iguacen.com/feed/
https://www.etsy.com/shop/CrowsNestJewels/rss
http://littleheartsbiglove.co.uk/feed/
http://www.oxv234.com/feeds/posts/default?alt=rss
https://latinashadows.wordpress.com/feed/
http://powai.tumblr.com/rss
http://theesongbird.tumblr.com/rss
https://tylershepard1991.wordpress.com/feed/
http://madh-mama.blogspot.com/feeds/posts/default?alt=rss
https://earth2levi.wordpress.com/feed/
https://thegreenmockingbird.wordpress.com/feed/
https://ajbookreviewclub.wordpress.com/feed/
https://www.mmstartups.com/feed/
https://www.mmstartups.com/author/abe-salisbury/feed/
http://www.danjjroberts.co.uk/feed/
http://kmillshernandez.com/feed/
http://iguacen.com/rss
http://iguacen.com/feed/
https://justjeanine.wordpress.com/feed/
http://rkt2.blog.fc2.com/?xml
https://latinashadows.wordpress.com/feed/
http://carrie-i.blogspot.com/feeds/posts/default?alt=rss
https://thebookreviewcafe.wordpress.com/feed/
http://katiebugstory.blogspot.com/feeds/posts/default?alt=rss
http://configuroweb.blogspot.com/feeds/posts/default?alt=rss

Next, I created a matrix of word frequencies for the blogs, limiting the word list to the 500 most popular terms. I used the code found in generatefeedvector.py from Programming Collective Intelligence here to create the matrix and put a red rectangle around the code I added when I was trying to get the 500 most frequent words with a sorted dictionary:

```python
import feedparser
import re

# Returns title and dictionary of word counts for an RSS
def getwordcounts(url):
    # Parse the feed
    d=feedparser.parse(url)
    wc={}

    # Loop over all the entries
    for e in d.entries:
        if 'summary' in e: summary=e.summary
        else: summary=e.description

        # Extract a list of words
        words=getwords(e.title+' '+summary)
        for word in words:
            wc.setdefault(word,0)
            wc[word]+=1
    return d.feed.title,wc

def getwords(html):
    # Remove all the HTML tags
    txt=re.compile(r'<[^>]+>').sub('',html)

    # Split words by all non-alpha characters
    words=re.compile(r'[^A-Z^a-z]+').split(txt)

    # Convert to lowercase
    return [word.lower() for word in words if word!='']

if __name__ == '__main__':
```

blogdata.txt

Blog	don	has	how	they	post	will	love	been	blog	who	make	really
back	our	get	know	had	their	well	here	year	think	ve	things	her
no	which	see	these	people	very	than	were	them	would	because	most	two
into	way	great	work	now	take	why	over	got	part	say	good	he
other	go	only	always	re	also	even	going	where	last	could	another	find
she	much	while	d	through	want	after	years	reading	many	life	around	world
long	made	am	every	few	didn	since	his	week	off	us	little	did
may	still	never	right	found	lot	look	being	let	next	something		then
days	need	should	old	down	before	ll	feel	show	ever	live	end	put
any	sure	same	thought	read	does	start	once	today	went	doing	nbsp	better
those	own	everyone		use	each	him	thing	too	review	set	away	come
change	give	such	free	times	top	hope	said	both	used	night	again	full
started	making	name	list	place	came	best	probably		big	bit	doesn	friends
school	family	music	actually		pretty	different		left	book	home	getting	looking
nothing	enough	amp	continue		everything		together		having	finally	girl	took
seen	able	com	mind	three	happy	until	video	tell	help	perfect	side	try
story	bad	myself	fun	check	trying	follow	must	share	high	keep	watch	coming
its	style	house	wanted	least	point	others	whole	yet	easy	real	anything	
months	ago	e	makes	talk	looks	might	someone	quite	already	haven	b	though
stuff	almost	without	yes	often	run	sometimes		far	past	second	write	course
amazing	mean	hard	idea	done	kind	future	especially		later	face	black	decided
however	comes	light	less	couple	become	twitter	oh	single	th	enjoy	writing	along
beautiful		man	remember		maybe	facebook		group	isn	friend	loved	saw
during	won	between	taking	n	super	working	head	called	young	under	small	
business		wrong	break	couldn	yourself		wasn	white	believe	feeling	online	play
definitely		stop	case	person	rest	spring	water	nice	felt	blue	room	heart
favorite		huge	interesting		half	close	please	job	c	moment	early	photo
lost	series	forward	call	inside	picture	age	focus	month	sound	order	hours	wait
sunday	instead	front	special	red	anyone	behind	words	morning	art	seems	recently	
sweet	ready	etc	simple	march	dark	although		using	four	posts	told	kids
experience		usually	heard	means	including		thank	l	books	god	reason	turn
sounds	open	seem	ways	based	money	hit	across	asked	pop	running	understand	
deep	leave	hand	thanks	questions		either	dead	knew	road	seeing	support	true
minutes	trip	worth	outside	cool	above	ask	taken	known	learn	rather	soon	
personal		late	weeks	space	size	girls	needed	looked	link	pictures		april
create	due	else	country	spent	add	possible		five	works	v	buy	below
general	important		fact	lives	starting		level	p	strong	changed	visit	phone
given	word	following		january	date	short	main	similar	expect	tried	friday	city
guess	eyes	mostly	exactly	mine	talking	self	note	clear	release	national		lots
appeared		walk	simply	women	worked	http	thoughts		hands	excited	power	rock
area	form	line	places	whether	fall	completely		win	played	type	car	crazy
playing	cold	choose	food	february								
	2	1	0	0	0	0	0	0	2	0	2	0
0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	1

Plain Text · Tab Width: 8 · Ln 5, Col 1 · INS

generatefeedvector.py

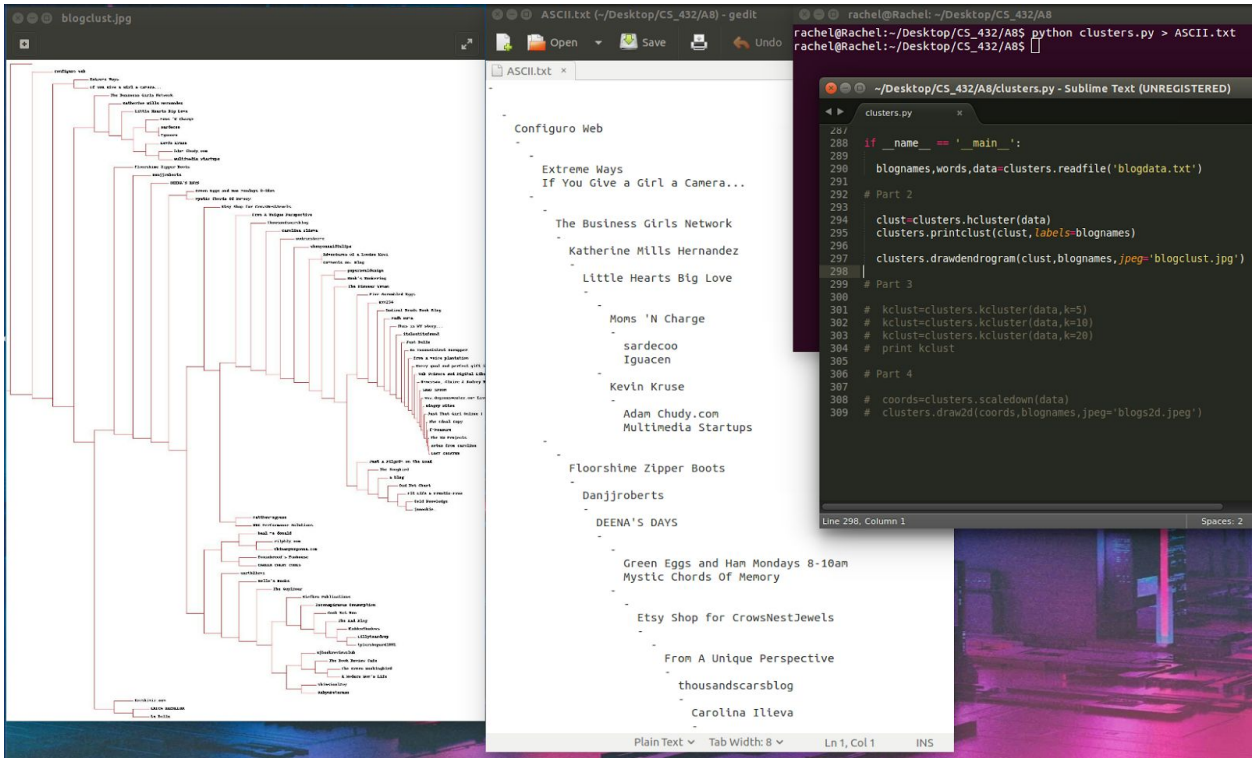
```python
if __name__ == '__main__':

    apcount={}
    wordcounts={}
    feedlist=[line for line in file('rss_links.txt')]

    for feedurl in feedlist:
        try:
            title,wc=getwordcounts(feedurl)
            wordcounts[title]=wc
            for word,count in wc.items():
                apcount.setdefault(word,0)
                if count>1:
                    apcount[word]+=1
        except:
            print 'Failed to parse feed %s' % feedurl
    wordlist=[]
    for w,bc in apcount.items():
        frac=float(bc)/len(feedlist)
        if frac>0.1 and frac<0.5:
            wordlist.append(w)

    temp = {}
    most_frequent_words = []

    for word in wordlist:
        temp[word] = apcount[word]
    i = 1
    for word, count in sorted(temp.items(), key=lambda x: x[1], reverse=True):
        if i <=500:
            most_frequent_words.append(word)
            i = i + 1
        else: break
    out=file('blogdata.txt','w')
    out.write('Blog')
    for word in most_frequent_words: out.write('\t%s' % word)
    out.write('\n')

    for blog,wc in wordcounts.items():
        try:
            out.write('\n')
            out.write(blog)
            for word in most_frequent_words:
                if word in wc: out.write('\t%d' % wc[word])
                else: out.write('\t0')
        except:
            pass
    out.write('\n\n')
```

```
Failed to parse feed /services/feeds/photos_public.gne?id=73341591@N00&lang=en-us&format=rss_200
[Finished in 80.2s]
```

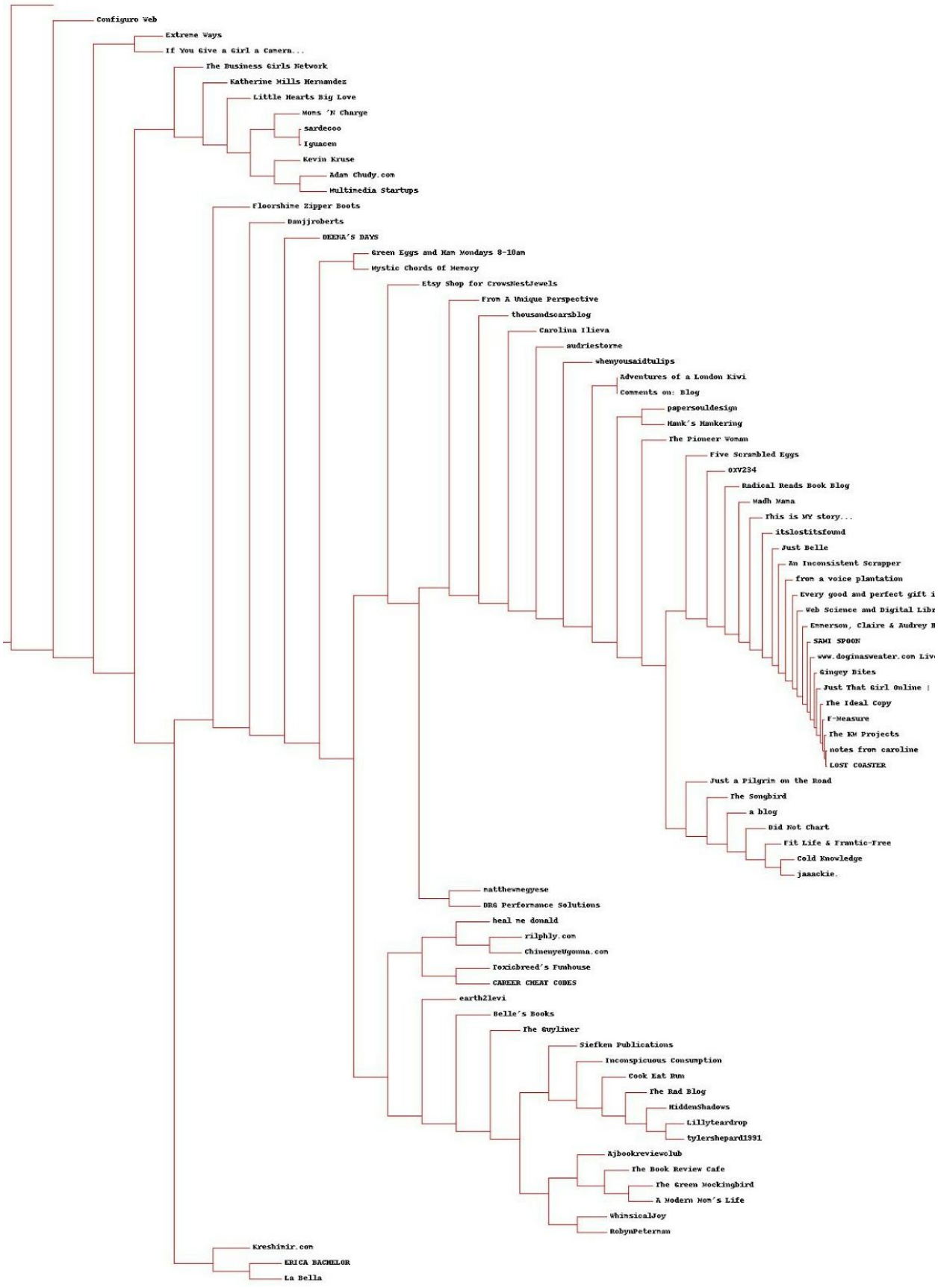
Note: I had to run it again to fix the formatting, so the blogdata.txt shown here isn't going to look the same as the one I submit.

PART 2

I used the code found in clusters.py from Programming Collective Intelligence here to print out a cluster of the blogs in both ASCII and jpeg formats:



Closer look at blogclust.jpeg on the next page:


Configuro Web
Extreme Ways
If You Give a Girl a Camera...
The Business Girls Network
Katherine Mills Hernandez
Little Hearts Big Love
Moms 'N Charge
sardecoo
Iguacen
Kevin Kruse
Adam Chudy.com
Multimedia Startups
Floorshine Zipper Boots
Danjjroberts
DEENA'S DAYS
Green Eggs and Ham Mondays 8-10am
Mystic Chords Of Memory
Etsy Shop for CrowsNestJewels
From A Unique Perspective
thousandscarsblog
Carolina Ilieva
audriestorme
whenyousaidtulips
Adventures of a London Kiwi
Comments on: Blog
papersouldesign
Hank's Hankering
The Pioneer Woman
Five Scrambled Eggs
OXV234
Radical Reads Book Blog
Madh Mama
This is MY story...
itslostitsfound
Just Belle
An Inconsistent Scrapper
from a voice plantation
Every good and perfect gift i
Web Science and Digital Libr
Emmerson, Claire & Audrey H
SAMI SPOON
www.doginasweater.com Liv
Gingey Bites
Just That Girl Online |
The Ideal Copy
F-Measure
The KM Projects
notes from caroline
LOST COASTER
Just a Pilgrim on the Road
The Songbird
a blog
Did Not Chart
Fit Life & Frantic-Free
Cold Knowledge
jaaackie.
matthewmegyese
DRG Performance Solutions
heal me donald
rilphly.com
ChinenyeUgonna.com
Toxicbreed's Funhouse
CAREER CHEAT CODES
earth2levi
Belle's Books
The Guyliner
Siefken Publications
Inconspicuous Consumption
Cook Eat Run
The Rad Blog
HiddenShadows
Lillyteardrop
tylershepard1991
Ajbookreviewclub
The Book Review Cafe
The Green Mockingbird
A Modern Mom's Life
WhimsicalJoy
RobynPeterman
Kreshimir.com
ERICA BACHELOR
La Bella

PART 3

Continuing with the same code found in clusters.py from Programming Collective Intelligence here to cluster the blogs using K-means.

K = 5 (4 iterations):

clusters.py

```python

if __name__ == '__main__':

  blognames,words,data=clusters.readfile('blogdata.txt')

# Part 2

#  clust=clusters.hcluster(data)
#  clusters.printclust(clust,labels=blognames)

#  clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')

# Part 3

  kclust=clusters.kcluster(data,k=5)
#  kclust=clusters.kcluster(data,k=10)
#  kclust=clusters.kcluster(data,k=20)
  print kclust

# Part 4

#  coords=clusters.scaledown(data)
#  clusters.draw2d(coords,blognames,jpeg='blogs2d.jpeg')
```

```
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
[[0, 1, 2, 3, 5, 6, 8, 11, 12, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 49, 51, 52,
53, 55, 57, 58, 59, 60, 62, 63, 65, 66, 69, 70, 72, 73, 74, 76, 77, 78, 79, 80, 81],
[7, 9, 10, 15, 33, 54, 56, 61, 68], [67], [4, 39, 71], [17, 48, 50, 64, 75]]
[Finished in 0.9s]
```

K = 10 (3 iterations):

```python
if __name__ == '__main__':

  blognames,words,data=clusters.readfile('blogdata.txt')

# Part 2

#  clust=clusters.hcluster(data)
#  clusters.printclust(clust,labels=blognames)

#  clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')

# Part 3

#  kclust=clusters.kcluster(data,k=5)
  kclust=clusters.kcluster(data,k=10)
#  kclust=clusters.kcluster(data,k=20)
  print kclust

# Part 4

#  coords=clusters.scaledown(data)
#  clusters.draw2d(coords,blognames,jpeg='blogs2d.jpeg')
```

```
Iteration 0
Iteration 1
Iteration 2
Iteration 3
[[40, 48, 64], [0, 2, 3, 5, 8, 11, 12, 13, 14, 16, 18, 19, 20, 21, 23, 26, 27, 28,
29, 30, 31, 32, 36, 37, 42, 43, 45, 46, 47, 49, 51, 52, 53, 55, 57, 60, 62, 63, 66,
70, 72, 73, 74, 76, 78, 79, 80, 81], [], [], [1, 6, 17, 22, 24, 25, 35, 38, 41, 44,
58, 59, 65, 69, 77], [7, 9, 34, 61], [4, 39, 71], [], [50, 54, 68, 75], [10, 15, 33,
56, 67]]
[Finished in 1.4s]
```

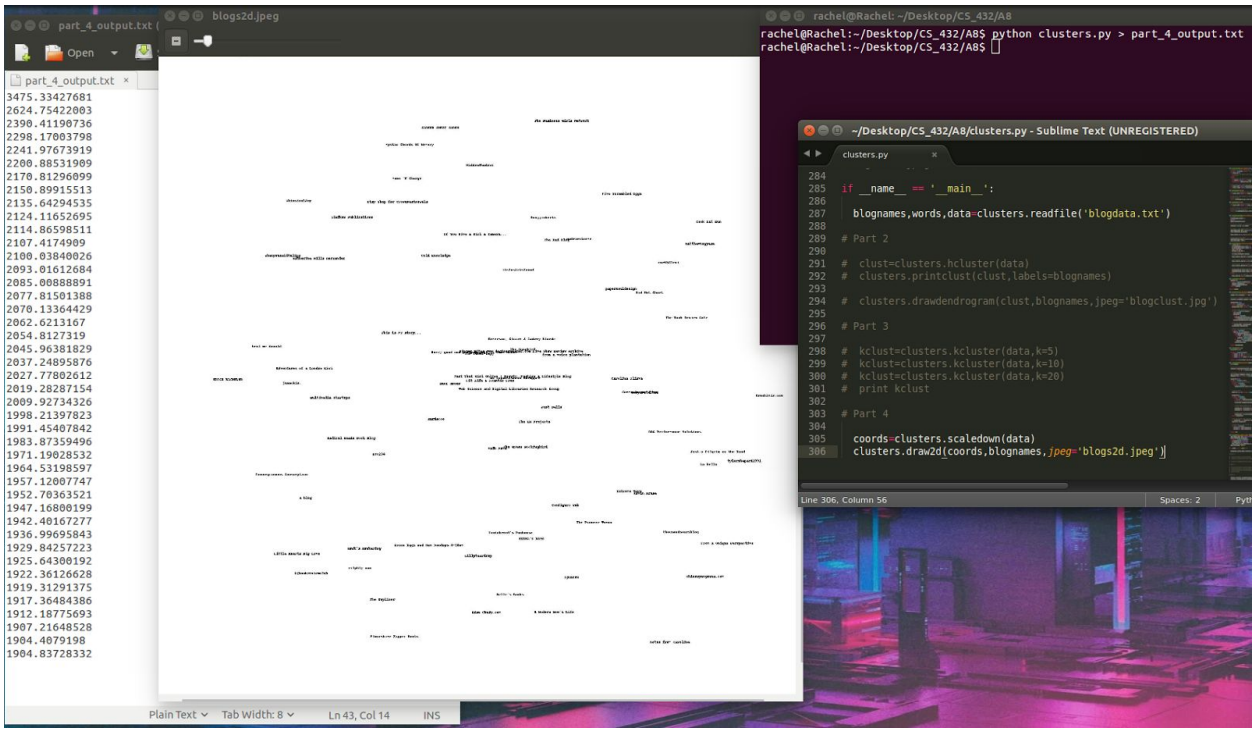
K = 20 (6 iterations):

```python
if __name__ == '__main__':

  blognames,words,data=clusters.readfile('blogdata.txt')

# Part 2

#  clust=clusters.hcluster(data)
#  clusters.printclust(clust,labels=blognames)

#  clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')

# Part 3

#  kclust=clusters.kcluster(data,k=5)
#  kclust=clusters.kcluster(data,k=10)
  kclust=clusters.kcluster(data,k=20)
  print kclust

# Part 4

#  coords=clusters.scaledown(data)
#  clusters.draw2d(coords,blognames,jpeg='blogs2d.jpeg')
```

```
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
[[7, 40], [], [1, 6, 38, 41, 59, 69], [33, 56, 68], [39], [5, 9, 20, 21, 22, 25, 29,
34, 43, 52, 60, 63, 70], [], [], [0, 2, 3, 8, 13, 14, 16, 18, 19, 23, 24, 26, 27, 30,
31, 32, 35, 36, 37, 42, 44, 45, 46, 47, 49, 51, 55, 58, 62, 65, 66, 72, 73, 74, 76,
78, 79, 80], [4], [11, 15, 28, 53, 77, 81], [71], [10, 17, 48, 50, 54], [], [75],
[12, 57, 64], [], [67], [61], []]
[Finished in 4.9s]
```

PART 4

Still continuing with the same code found in clusters.py from Programming Collective Intelligence [here]() to cluster the blogs using multidimensional scaling:



Closer look at blogs2d.jpeg on the next page:

DEENA'S DAYS

Toxicbreed's Funhouse

notes from carolinerson, Claire & Audrey Bloede

The Ideal Copy

F-Measure

Etsy Shop for CrowsNestJewels

Just That Girl Online | Beauty, Fashion & Lifestyle Blog

Web Science and Digital Libraries Research Group

Gingey Bites

www.ginasweater.com Live Show Review Archive

itslostitsfound

Danjjroberts

The Business Girls Network

Every good and perfect gift is from above.

SAVI SPOON

from a voice plantation

Multimedia Startups

An Inconsistent Scrapper

Just Belle

Radical Reads Book Blog

thousandscarsblog

earth2levi

From A Unique Perspective

ChinenyeUgonna.com

Belle's Books

This is MY story...

Floorshine Zipper Boots

Wadh Wana

Katherine Hills Hernandez

Five Scrambled Eggs

Just a Pilgrim on the Road

Hank's Hankering

Green Eggs and Man Mondays 8-10am

The Pioneer Woman

Did Not Chart

The Book Review Cafe

OXV234

Lillyteardrop

matthewneggese

a blog

Adventures of a London Kiwi

Ignacen

Cold Knowledge

sardecoo

The Green Mockingbird

Fit Life & Frantic-Free

Carolina Ilieva

The Songbird

ERICA BACHELOR

Audriestorne

papersouldesign

tylershepard1991

jaaackie.

rilphly.com

Kevin Kruse

Mons 'N Charge

Kreshinir.com

Comments on: Blog

HiddenShadows

Siefken Publications

whenyousaidtulips

heal me donald

Inconspicuous Consumption

La Bella

Mystic Chords Of Memory

The Rad Blog

RobynPeterman

Little Hearts Big Love

Adam Chudy.com

WhimsicalJoy

If You Give a Girl a Camera...

CAREER CHEAT CODES

BRG Performance Solutions

The Guyliner

Ajbookreviewclub

A Modern Mom's Life

Cook Eat Run

Extreme Ways

Configure Web

The rest of the code from clusters.py used for parts 2, 3, and 4 (I put these at the end in case you wanted to get to the other stuff first):

clusters.py

```python
from PIL import Image,ImageDraw
import clusters

def readfile(filename):
  lines=[line for line in file(filename)]

  # First line is the column titles
  colnames=lines[0].strip().split('\t')[1:]
  rownames=[]
  data=[]
  for line in lines[1:]:
    p=line.strip().split('\t')
    # First column in each row is the rowname
    rownames.append(p[0])
    # The data for this row is the remainder of the row
    data.append([float(x) for x in p[1:]])
  return rownames,colnames,data


from math import sqrt

def pearson(v1,v2):
  # Something weird's going on with my data, this is just a workaround
  if len(v1) == 0 or len(v2) == 0:
    return 1.0
  # Simple sums
  sum1=sum(v1)
  sum2=sum(v2)

  # Sums of the squares
  sum1Sq=sum([pow(v,2) for v in v1])
  sum2Sq=sum([pow(v,2) for v in v2])

  # Sum of the products
#  print ('V1 len = ', len(v1))
#  print ('V2 len = ', len(v2))
  pSum=sum([v1[i]*v2[i] for i in range(len(v1))])

  # Calculate r (Pearson score)
  num=pSum-(sum1*sum2/len(v1))
  den=sqrt((sum1Sq-pow(sum1,2)/len(v1))*(sum2Sq-pow(sum2,2)/len(v1)))
  if den==0: return 0

  return 1.0-num/den

class bicluster:
  def __init__(self,vec,left=None,right=None,distance=0.0,id=None):
    self.left=left
    self.right=right
    self.vec=vec
    self.id=id
    self.distance=distance

def hcluster(rows,distance=pearson):
  distances={}
  currentclustid=-1

  # Clusters are initially just the rows
  clust=[bicluster(rows[i],id=i) for i in range(len(rows))]
```

```python
  # Clusters are initially just the rows
  clust=[bicluster(rows[i],id=i) for i in range(len(rows))]

  while len(clust)>1:
    lowestpair=(0,1)
    closest=distance(clust[0].vec,clust[1].vec)

    # loop through every pair looking for the smallest distance
    for i in range(len(clust)):
      for j in range(i+1,len(clust)):
        # distances is the cache of distance calculations
        if (clust[i].id,clust[j].id) not in distances:
          distances[(clust[i].id,clust[j].id)]=distance(clust[i].vec,clust[j].vec)

        d=distances[(clust[i].id,clust[j].id)]

        if d<closest:
          closest=d
          lowestpair=(i,j)

    # calculate the average of the two clusters
    mergevec=[
    (clust[lowestpair[0]].vec[i]+clust[lowestpair[1]].vec[i])/2.0
    for i in range(len(clust[0].vec))]

    # create the new cluster
    newcluster=bicluster(mergevec,left=clust[lowestpair[0]],
                         right=clust[lowestpair[1]],
                         distance=closest,id=currentclustid)

    # cluster ids that weren't in the original set are negative
    currentclustid-=1
    del clust[lowestpair[1]]
    del clust[lowestpair[0]]
    clust.append(newcluster)

  return clust[0]

def printclust(clust,labels=None,n=0):
  # indent to make a hierarchy layout
  for i in range(n): print ' ',
  if clust.id<0:
    # negative id means that this is branch
    print '-'
  else:
    # positive id means that this is an endpoint
    if labels==None: print clust.id
    else: print labels[clust.id]

  # now print the right and left branches
  if clust.left!=None: printclust(clust.left,labels=labels,n=n+1)
  if clust.right!=None: printclust(clust.right,labels=labels,n=n+1)

def getheight(clust):
  # Is this an endpoint? Then the height is just 1
  if clust.left==None and clust.right==None: return 1

  # Otherwise the height is the same of the heights of
```

```
  # Otherwise the height is the same of the heights of
  # each branch
  return getheight(clust.left)+getheight(clust.right)

def getdepth(clust):
  # The distance of an endpoint is 0.0
  if clust.left==None and clust.right==None: return 0

  # The distance of a branch is the greater of its two sides
  # plus its own distance
  return max(getdepth(clust.left),getdepth(clust.right))+clust.distance


def drawdendrogram(clust,labels,jpeg='clusters.jpg'):
  # height and width
  h=getheight(clust)*20
  w=1200
  depth=getdepth(clust)

  # width is fixed, so scale distances accordingly
  scaling=float(w-150)/depth

  # Create a new image with a white background
  img=Image.new('RGB',(w,h),(255,255,255))
  draw=ImageDraw.Draw(img)

  draw.line((0,h/2,10,h/2),fill=(255,0,0))

  # Draw the first node
  drawnode(draw,clust,10,(h/2),scaling,labels)
  img.save(jpeg,'JPEG')

def drawnode(draw,clust,x,y,scaling,labels):
  if clust.id<0:
    h1=getheight(clust.left)*20
    h2=getheight(clust.right)*20
    top=y-(h1+h2)/2
    bottom=y+(h1+h2)/2
    # Line length
    ll=clust.distance*scaling
    # Vertical line from this cluster to children
    draw.line((x,top+h1/2,x,bottom-h2/2),fill=(255,0,0))

    # Horizontal line to left item
    draw.line((x,top+h1/2,x+ll,top+h1/2),fill=(255,0,0))

    # Horizontal line to right item
    draw.line((x,bottom-h2/2,x+ll,bottom-h2/2),fill=(255,0,0))

    # Call the function to draw the left and right nodes
    drawnode(draw,clust.left,x+ll,top+h1/2,scaling,labels)
    drawnode(draw,clust.right,x+ll,bottom-h2/2,scaling,labels)
  else:
    # If this is an endpoint, draw the item label
    draw.text((x+5,y-7),labels[clust.id],(0,0,0))

def rotatematrix(data):
  newdata=[]
```

```python
def rotatematrix(data):
  newdata=[]
  for i in range(len(data[0])):
    newrow=[data[j][i] for j in range(len(data))]
    newdata.append(newrow)
  return newdata

import random

def kcluster(rows,distance=pearson,k=4):
  # Determine the minimum and maximum values for each point
  ranges=[(min([row[i] for row in rows]),max([row[i] for row in rows]))
  for i in range(len(rows[0]))]

  # Create k randomly placed centroids
  clusters=[[random.random()*(ranges[i][1]-ranges[i][0])+ranges[i][0]
  for i in range(len(rows[0]))] for j in range(k)]

  lastmatches=None
  for t in range(100):
    print 'Iteration %d' % t
    bestmatches=[[] for i in range(k)]

    # Find which centroid is the closest for each row
    for j in range(len(rows)):
      row=rows[j]
      bestmatch=0
      for i in range(k):
        d=distance(clusters[i],row)
        if d<distance(clusters[bestmatch],row): bestmatch=i
      bestmatches[bestmatch].append(j)

    # If the results are the same as last time, this is complete
    if bestmatches==lastmatches: break
    lastmatches=bestmatches

    # Move the centroids to the average of their members
    for i in range(k):
      avgs=[0.0]*len(rows[0])
      if len(bestmatches[i])>0:
        for rowid in bestmatches[i]:
          for m in range(len(rows[rowid])):
            avgs[m]+=rows[rowid][m]
        for j in range(len(avgs)):
          avgs[j]/=len(bestmatches[i])
        clusters[i]=avgs

  return bestmatches

def tanamoto(v1,v2):
  c1,c2,shr=0,0,0

  for i in range(len(v1)):
    if v1[i]!=0: c1+=1 # in v1
    if v2[i]!=0: c2+=1 # in v2
    if v1[i]!=0 and v2[i]!=0: shr+=1 # in both

  return 1.0-(float(shr)/(c1+c2-shr))
```

```python
def scaledown(data,distance=pearson,rate=0.01):
  n=len(data)

  # The real distances between every pair of items
  realdist=[[distance(data[i],data[j]) for j in range(n)]
            for i in range(0,n)]

  # Randomly initialize the starting points of the locations in 2D
  loc=[[random.random(),random.random()] for i in range(n)]
  fakedist=[[0.0 for j in range(n)] for i in range(n)]

  lasterror=None
  for m in range(0,1000):
    # Find projected distances
    for i in range(n):
      for j in range(n):
        fakedist[i][j]=sqrt(sum([pow(loc[i][x]-loc[j][x],2)
                                 for x in range(len(loc[i]))]))

    # Move points
    grad=[[0.0,0.0] for i in range(n)]

    totalerror=0
    for k in range(n):
      for j in range(n):
        if j==k: continue
        # The error is percent difference between the distances
        errorterm=(fakedist[j][k]-realdist[j][k])/realdist[j][k]

        # Each point needs to be moved away from or towards the other
        # point in proportion to how much error it has
        grad[k][0]+=((loc[k][0]-loc[j][0])/fakedist[j][k])*errorterm
        grad[k][1]+=((loc[k][1]-loc[j][1])/fakedist[j][k])*errorterm

        # Keep track of the total error
        totalerror+=abs(errorterm)
    print totalerror

    # If the answer got worse by moving the points, we are done
    if lasterror and lasterror<totalerror: break
    lasterror=totalerror

    # Move each of the points by the learning rate times the gradient
    for k in range(n):
      loc[k][0]-=rate*grad[k][0]
      loc[k][1]-=rate*grad[k][1]

  return loc

def draw2d(data,labels,jpeg='mds2d.jpg'):
  img=Image.new('RGB',(2000,2000),(255,255,255))
  draw=ImageDraw.Draw(img)
  for i in range(len(data)):
    x=(data[i][0]+0.5)*1000
    y=(data[i][1]+0.5)*1000
    draw.text((x,y),labels[i],(0,0,0))
  img.save(jpeg,'JPEG')
```