INTRODUCTION TO WEB SCIENCE- FALL 2014

Assignment Nine

Sybil Melton

December 3, 2014

CONTENTS

1 Blog Term Matrix 3
1.1 Solution 3

2 Dendrograms 7
2.1 Solution 7

3 K-clustering 10
3.1 Solution 10

4 MDS 10
4.1 Solution 10

5 TFIDF 12
5.1 Solution 12

6 Python Code 15

LISTINGS

1 Retrieve Blogs 3
2 Retrieve Wordcounts 4
3 Create Blog Matrix 5
4 Pages vs Number of Blogs R 7
5 Create Dendrograms 7
6 Create K-clusters 10
7 Create MDS 10
8 TFIDF Calculation 12
9 Complete Feed Parser 15

1 Blog Term Matrix

Create a blog-term matrix. Start by grabbing 100 blogs; include:

- http://f-measure.blogspot.com/
- http://ws-dl.blogspot.com/

and grab 98 more as per the method shown in class.

Use the blog title as the identifier for each blog (and row of the matrix). Use the terms from every item/title (RSS) or entry/title (Atom) 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. Limit the number of terms to the most "popular" (i.e., frequent) 500 terms, this is *after* the criteria on p. 32 (slide 7) has been satisfied.

Create a histogram of how many pages each blog has (e.g., 30 blogs with just one page, 27 with two pages, 29 with 3 pages and so on).

1.1 Solution

My first step was to grab a list of blogs, using Blogger.com's "next blog" functionality. The blogs were all written to file by net location. After writing the two required blogs, a loop of 200 iterations ran to get additional blogs, using pyCurl to get the net location. [8] Once each blog was retrieved, it was checked to make sure it didn't already exist in the list with the regular expression module. This is the same function used in Assignment One. [2] Once I had the list, I went through it to check for inappropriate content and to remove any picture, video, or foreign language blogs. My final list of blogs was saved as blogs.txt and is included with this report. Listing 1 is the code used to accomplish the first part of this task.

```python
def check(url):
  with open("blogs.txt") as f:
    found = False
    for line in f:  #iterate over the file one line at a time(memory efficient)
      if re.search(url, line):    #if string found is in current line then keep it
        found = True
  return found

blogger = 'https://www.blogger.com/next-blog?navBar=true&blogID=953024975153422094'
bfile = open('blogs.txt', 'w',0)
bfile.write('f-measure.blogspot.com\n')
bfile.write('ws-dl.blogspot.com\n')
bfile.close()
for i in range(1, 200):
  buffer = BytesIO()
  c = pycurl.Curl()
  c.setopt(c.URL, blogger)
  c.setopt(c.WRITEDATA, buffer)
  c.setopt(c.FOLLOWLOCATION, True)
  c.setopt(c.HTTPHEADER, ['Accept-Language: en'])
  bfile = open('blogs.txt', 'a',0)
  try:
    c.perform()
    if c.getinfo(c.RESPONSE_CODE) == 200:
      o = urlparse(c.getinfo(c.EFFECTIVE_URL))
      ch = check(o.netloc)
      if not ch:
        bfile.write(o.netloc + '\n')
        bfile.close()
      else:
        continue
    c.close()
  except pycurl.error, error:
    errno, errstr = error
    print ’An error occurred: ’, errstr
```

Listing 1: Retrieve Blogs

My next step was to retrieve the blogs. Each blog in “blogs.txt” was added the feed path */feeds/posts/default* and encoded with query arguments for Atom, in order to get the raw feed. A feed list was kept, for the stop word hack calculation later on in the program. Each feed was retrieved using the given code to create a getFeed function, using getwordcounts and getwords from the PCI textbook to create the wordcounts dictionary of word counts per blog and apcount dictionary of counts per word. [10] For testing, the Web Science/Digital Libraries blog was downloaded to make sure the feed was retrieved and all BeautifulSoup was used to look for the link to the next page, if available. [3] A counter kept track of the number of pages in the blog, which was used for debugging and stored in a dictionary. The pages per blog and number of blogs was written to blog_pages.txt in order to create the histogram and is included with this report. [7] The apcount dictionary was unsorted, so in order to get the top 500 words, I converted it to a sorted list by word count. A counter was added to the code and stopped the loop if 500 was reached. Listing 2 is the code used to for these tasks.

```
def getwordcounts(url):
    #Returns title and dictionary of word counts for an RSS feed
    # 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 != ’’]

def getFeed(feedurl):
    global wordcounts
    global apcount
    try:
        (title, wc) = getwordcounts(feedurl)
        if title in wordcounts:
            for (w, c) in wc.iteritems():
                if w in wordcounts[title].iteritems():
                    wordcounts[title][w] += c
                else:
                    wordcounts[title][w] = c
        else:
            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

pages = {}
feedlist = []
```

```
apcount = {}
wordcounts = {}
wordlist = []
scheme = 'http://'
path = '/feeds/posts/default'
query_arg = {'alt' : 'atom'}
udata = urllib.urlencode(query_arg)
f = open('blogs.txt', 'r', 0)
for line in f:
  line = line.strip()
  data = urllib.urlencode(query_arg)
  full_url = scheme+line+path+"?"+udata
  feedlist.append(full_url)
  r = requests.get(full_url)
  getFeed(full_url)
  ddata = r.text
  soup = BeautifulSoup(ddata)
  next = soup.find('link', rel='next')
  count = 1
  while next:
    n= next.get('href')
    getFeed(n)
    r = requests.get(n)
    data = r.text
    soup = BeautifulSoup(data)
    count = count + 1
    if count % 10 == 0:
      print ' -- parsing ' + line + ' ' + str(count) + ' pages so far'
    next = soup.find('link', rel='next')
  print 'Finished parsing ' + line + ' ' + str(count) + ' pages'
  if str(count) in pages:
    pages[str(count)].append(line)
  else:
    pages[str(count)] = []
    pages[str(count)].append(line)
f.close()

p = open('blog_pages.txt', 'w', 0)
p.write('Pages\tNumber of Blogs\n')
for (pg, num) in pages.iteritems():
  p.write(pg + '\t' + str(len(num)) + '\n')

a = sorted(apcount.items(), key=lambda x: x[1], reverse=True)
count = 0
for (w, bc) in a:
    frac = float(bc) / len(feedlist)
    if frac > 0.1 and frac < 0.5:
        wordlist.append(w)
        count = count + 1
        if count >= 500:
            break
```

Listing 2: Retrieve Wordcounts

At this point I realized I did not have a the titles or subtitles stored. So I wrote a function, getTitles, in order to accomplish this and were stored as a dictionary. The feed had to be parsed again for both title and subtitle. [5] I found a few still had html markup which caused extra lines and illegal unicode characters which had to be removed. [4], [6] The blog matrix was created and saved as blogdata.txt and is included with this report. The code from the textbook was modified to include the subtitle, or part of the subtitle, as long as the length was less than 100 characters. [11] Listing 3 is the code used to create the blog matrix.

```
_illegal_xml_chars_RE = re.compile(u'[\x00-\x08\x0b\xa9\x0c\x0e-\x1F\uD800-\uDFFF\uFFFE\uFFFF]')

def remove_tags(text):
    cleanr =re.compile('<.*?>')
    cleantext = re.sub(cleanr,'', text)
    cleantext = re.sub("\n", ' ', cleantext)
    cleantext = cleantext.strip()
    return _illegal_xml_chars_RE.sub(' ', cleantext)

def getTitles():
  titles = {}
  global scheme, path, udata
  f = open('blogs.txt', 'r', 0)
  for line in f:
    line = line.strip()
    full_url = scheme+line+path+"?"+udata
    d = feedparser.parse(full_url)
    titles[d.feed.title] = remove_tags(d.feed.subtitle)
  return titles

titles = {}
titles = getTitles()
# Blog Term Matrix
out = file('blogdata.txt', 'w', 0)
out.write('Blog')
for word in wordlist:
    out.write('\t%s' % word)
out.write('\n')

for (blog, wc) in wordcounts.items():
    if titles[blog] != '':
        blog = blog + ' - ' + titles[blog]
    blog = blog.replace(u'\u0144', 'n')
    blog = _illegal_xml_chars_RE.sub(' ', blog)
    blog = blog.strip()
    if len(blog) > 100:
        blog = blog[:99]
    print blog
    out.write(blog)
    for word in wordlist:
        if word in wc:
            out.write('\t%d' % wc[word])
        else:
            out.write('\t0')
    out.write('\n')
out.close()
```

Listing 3: Create Blog Matrix

Finally, the histogram was created in R, to show the number of blogs and pages. Figure 1.1 show there were more blogs with fewer pages, as there were only 12 blogs with greater than 18 pages. Listing 4 is the R code used to create it.

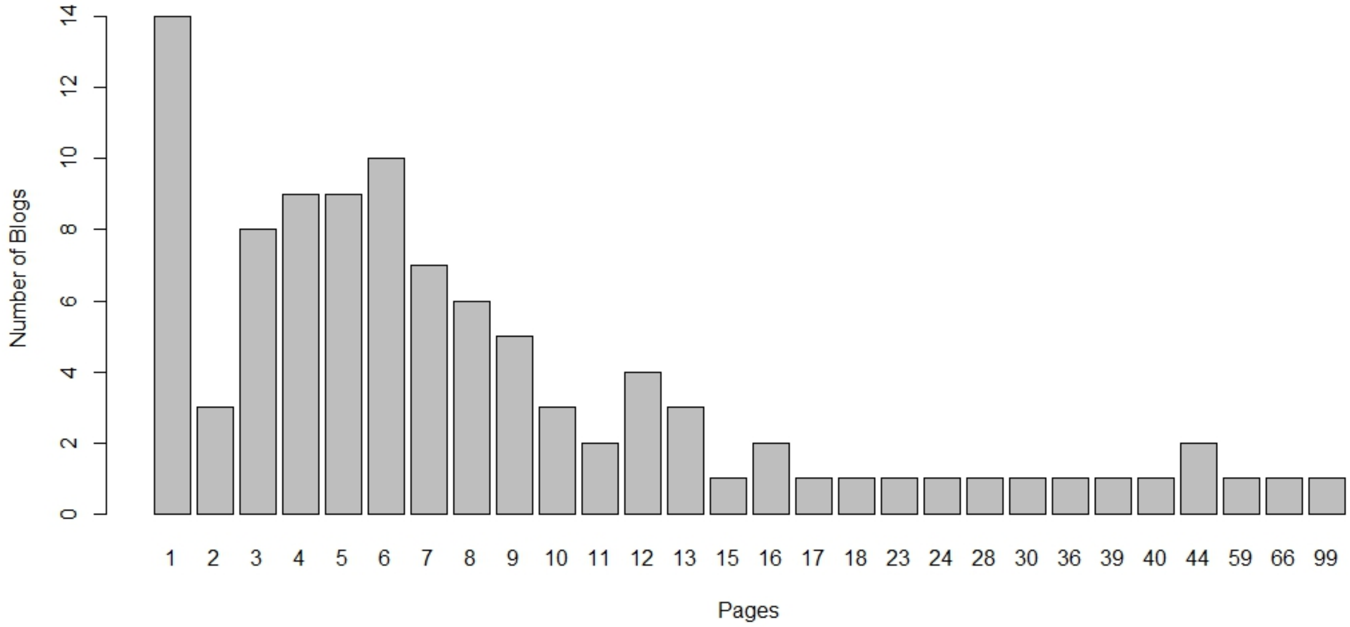

Figure 1.1: Pages vs Number of Blogs

```
bdata = read.table("C:/Users/sybil/Documents/2014 Fall/Web Science/A9/output/blog_pages.txt", header=T, sep
    ="\t")

newdata <- bdata[order(pages),]
pages <- c(newdata$Pages)
blogs <- c(newdata$Number.of.Blogs)

barplot(blogs, names.arg=pages, xlab="Pages", ylab="Number of Blogs")
```

Listing 4: Pages vs Number of Blogs R

2 Dendrograms

Create an ASCII and JPEG dendrogram that clusters (i.e., HAC) the most similar blogs (see slides 12 & 13). Include the JPEG in your report and upload the ascii file to github (it will be too unwieldy for inclusion in the report).

2.1 Solution

First I tried to change the provided function to output to a text file, but it was not successful. So I redirected the standard output temporarily to blog_ascii.txt, which is included with this report. [1] The JPEG dendogram was successful, as shown in Figure 2.1. The python code used is shown in Listing 5.

```
old = sys.stdout
sys.stdout = open(’blog_ascii.txt’, ’w’, 0)
blognames,words,data=clusters.readfile(’blogdata.txt’)
clust = clusters.hcluster(data)
```

```
clusters.printclust(clust,labels=blognames)
sys.stdout = old
# jpeg dendogram
clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')
```

Listing 5: Create Dendrograms

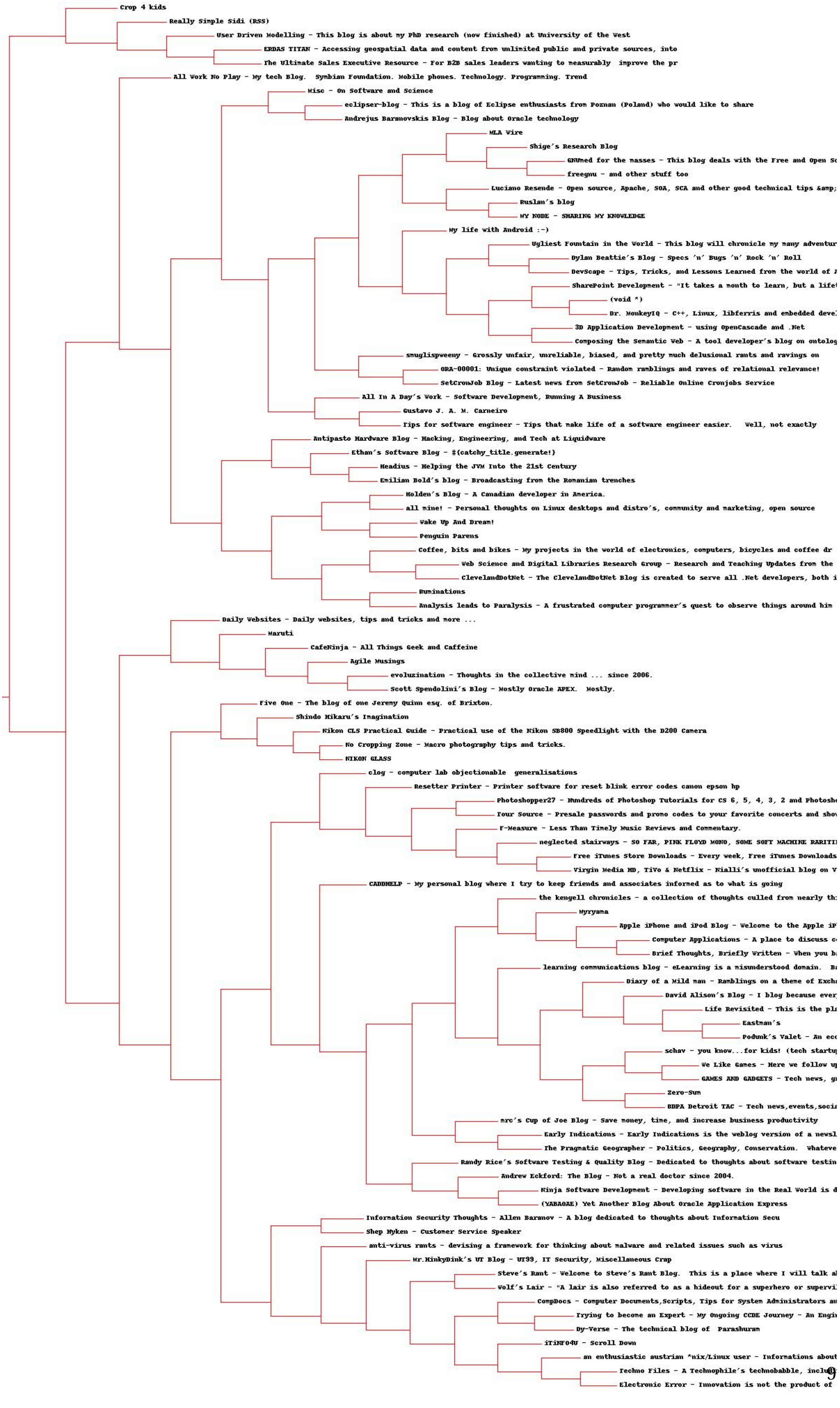
Crop 4 kids
Really Simple Sidi (RSS)
User Driven Modelling - This blog is about my PhD research (now finished) at University of the West
ERDAS TITAN - Accessing geospatial data and content from unlimited public and private sources, into
The Ultimate Sales Executive Resource - For B2B sales leaders wanting to measurably improve the pr
All Work No Play - My tech Blog. Symbian Foundation. Mobile phones. Technology. Programming. Trend
Misc - On Software and Science
eclipser-blog - This is a blog of Eclipse enthusiasts from Poznan (Poland) who would like to share
Andrejus Baranovskis Blog - Blog about Oracle technology
MLA Wire
Shige's Research Blog
GNUmed for the masses - This blog deals with the Free and Open Sc
freegnu - and other stuff too
Luciano Resende - Open source, Apache, SOA, SCA and other good technical tips &amp;
Ruslan's blog
MY NODE - SHARING MY KNOWLEDGE
My life with Android :-)
Ugliest Fountain in the World - This blog will chronicle my many adventur
Dylan Beattie's Blog - Specs 'n' Bugs 'n' Rock 'n' Roll
DevScape - Tips, Tricks, and Lessons Learned from the world of A
SharePoint Development - "It takes a month to learn, but a lifet
(void *)
Dr. MonkeyIQ - C++, Linux, libferris and embedded devel
3D Application Development - using OpenCascade and .Net
Composing the Semantic Web - A tool developer's blog on ontolog
smuglispweeny - Grossly unfair, unreliable, biased, and pretty much delusional rants and ravings on
ORA-00001: Unique constraint violated - Random ramblings and raves of relational relevance!
SetCronJob Blog - Latest news from SetCronJob - Reliable Online Cronjobs Service
All In A Day's Work - Software Development, Running A Business
Gustavo J. A. M. Carneiro
Tips for software engineer - Tips that make life of a software engineer easier. Well, not exactly
Antipasto Hardware Blog - Hacking, Engineering, and Tech at Liquidware
Ethan's Software Blog - #{catchy_title.generate!}
Headius - Helping the JVM Into the 21st Century
Emilian Bold's blog - Broadcasting from the Romanian trenches
Holden's Blog - A Canadian developer in America.
all mine! - Personal thoughts on Linux desktops and distro's, community and marketing, open source
Wake Up And Dream!
Penguin Parens
Coffee, bits and bikes - My projects in the world of electronics, computers, bicycles and coffee dr
Web Science and Digital Libraries Research Group - Research and Teaching Updates from the
ClevelandDotNet - The ClevelandDotNet Blog is created to serve all .Net developers, both i
Ruminations
Analysis leads to Paralysis - A frustrated computer programmer's quest to observe things around him
Daily Websites - Daily websites, tips and tricks and more ...
Maruti
CafeNinja - All Things Geek and Caffeine
Agile Musings
evoluzination - Thoughts in the collective mind ... since 2006.
Scott Spendolini's Blog - Mostly Oracle APEX. Mostly.
Five One - The blog of one Jeremy Quinn esq. of Brixton.
Shindo Hikaru's Imagination
Nikon CLS Practical Guide - Practical use of the Nikon SB800 Speedlight with the D200 Camera
No Cropping Zone - Macro photography tips and tricks.
NIKON GLASS
clog - computer lab objectionable generalisations
Resetter Printer - Printer software for reset blink error codes canon epson hp
Photoshopper27 - Hundreds of Photoshop Tutorials for CS 6, 5, 4, 3, 2 and Photosho
Your Source - Presale passwords and promo codes to your favorite concerts and show
F-Measure - Less Than Timely Music Reviews and Commentary.
neglected stairways - SO FAR, PINK FLOYD MONO, SOME SOFT MACHINE RARITII
Free iTunes Store Downloads - Every week, Free iTunes Downloads
Virgin Media HD, TiVo & Netflix - Nialli's unofficial blog on V
CADDHELP - My personal blog where I try to keep friends and associates informed as to what is going
the kengell chronicles - a collection of thoughts culled from nearly thi
Myryama
Apple iPhone and iPod Blog - Welcome to the Apple iP
Computer Applications - A place to discuss co
Brief Thoughts, Briefly Written - When you ba
learning communications blog - eLearning is a misunderstood domain. Ba
Diary of a Mild man - Ramblings on a theme of Excha
David Alison's Blog - I blog because every
Life Revisited - This is the pla
Eastman's
Podunk's Valet - An eco
schav - you know...for kids! (tech startup
We Like Games - Here we follow up
GAMES AND GADGETS - Tech news, ga
Zero-Sum
BDPA Detroit TAC - Tech news,events,socia
mrc's Cup of Joe Blog - Save money, time, and increase business productivity
Early Indications - Early Indications is the weblog version of a newsl
The Pragmatic Geographer - Politics, Geography, Conservation. Whateve
Randy Rice's Software Testing & Quality Blog - Dedicated to thoughts about software testin
Andrew Eckford: The Blog - Not a real doctor since 2004.
Ninja Software Development - Developing software in the Real World is d
(YABAOAE) Yet Another Blog About Oracle Application Express
Information Security Thoughts - Allen Baranov - A blog dedicated to thoughts about Information Secu
Shep Hyken - Customer Service Speaker
anti-virus rants - devising a framework for thinking about malware and related issues such as virus
Mr.HinkyDink's UT Blog - UT99, IT Security, Miscellaneous Crap
Steve's Rant - Welcome to Steve's Rant Blog. This is a place where I will talk al
Wolf's Lair - "A lair is also referred to as a hideout for a superhero or supervil
CompDocs - Computer Documents,Scripts, Tips for System Administrators an
Trying to become an Expert - My Ongoing CCDE Journey - An Engin
Dy-Verse - The technical blog of Parashuram
iTiNFO4U - Scroll Down
an enthusiastic austrian *nix/Linux user - Informations about
Techno Files - A Technophile's technobabble, inclu
Electronic Error - Innovation is not the product of

Figure 2.1: jPEG Dendogram

3 K-CLUSTERING

Cluster the blogs using K-Means, using k=5,10,20. (see slide 18). How many interations were required for each value of k?

3.1 SOLUTION

To get the k-clustering, I used the provided functions in clusters.py from the PCI textbook, however, I also printed the resulting cluster to a file, blog_k.txt. For K = 5, the code took three iterations, but when K = 10 and 20, it took four iterations. With five clusters, there was a group of ‘tips and tricks‘ blogs, three groups of technology blogs, for gadgets and software development, and a group of hobbies - photoshop, photography, concerts, etc. When K was larger, the technology blogs were split up into smaller clusters, while the other clusters seemed to have stuck together. The python code used to accomplish this is in Listing 6 The screen output from the code was saved as k_run.txt

```
def printClusters(kclust, out):
  for i in range(0,len(kclust)):
    klist = [blognames[r] for r in kclust[i]]
    for x in range(0, len(klist)):
      out.write(klist[x] + '\n')
    out.write('\n')

k = open('blog_k.txt', 'w', 0)
k.write('K = 5\n')
kclust=clusters.kcluster(data,k=5)
printClusters(kclust, k)
k.write('K = 10\n')
kclust=clusters.kcluster(data,k=10)
printClusters(kclust, k)
k.write('K = 20\n')
kclust=clusters.kcluster(data,k=20)
printClusters(kclust, k)
k.close()
```

Listing 6: Create K-clusters

4 MDS

Use MDS to create a JPEG of the blogs similar to slide 29. How many iterations were required?

4.1 SOLUTION

In order to create the JPEG, the provided code from the PCI textbook was used. The code used is shown in Listing 7 This section took 265 iterations. Although it wasn’t required, the iteration numbers were saved as mds.txt, which also made it easier to count the iterations.

```
old = sys.stdout
sys.stdout = open('blog_ascii.txt', 'w', 0)
blognames,words,data=clusters.readfile('blogdata.txt')
clust = clusters.hcluster(data)
clusters.printclust(clust,labels=blognames)
sys.stdout = old
# jpeg dendogram
clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')
```

Listing 7: Create MDS

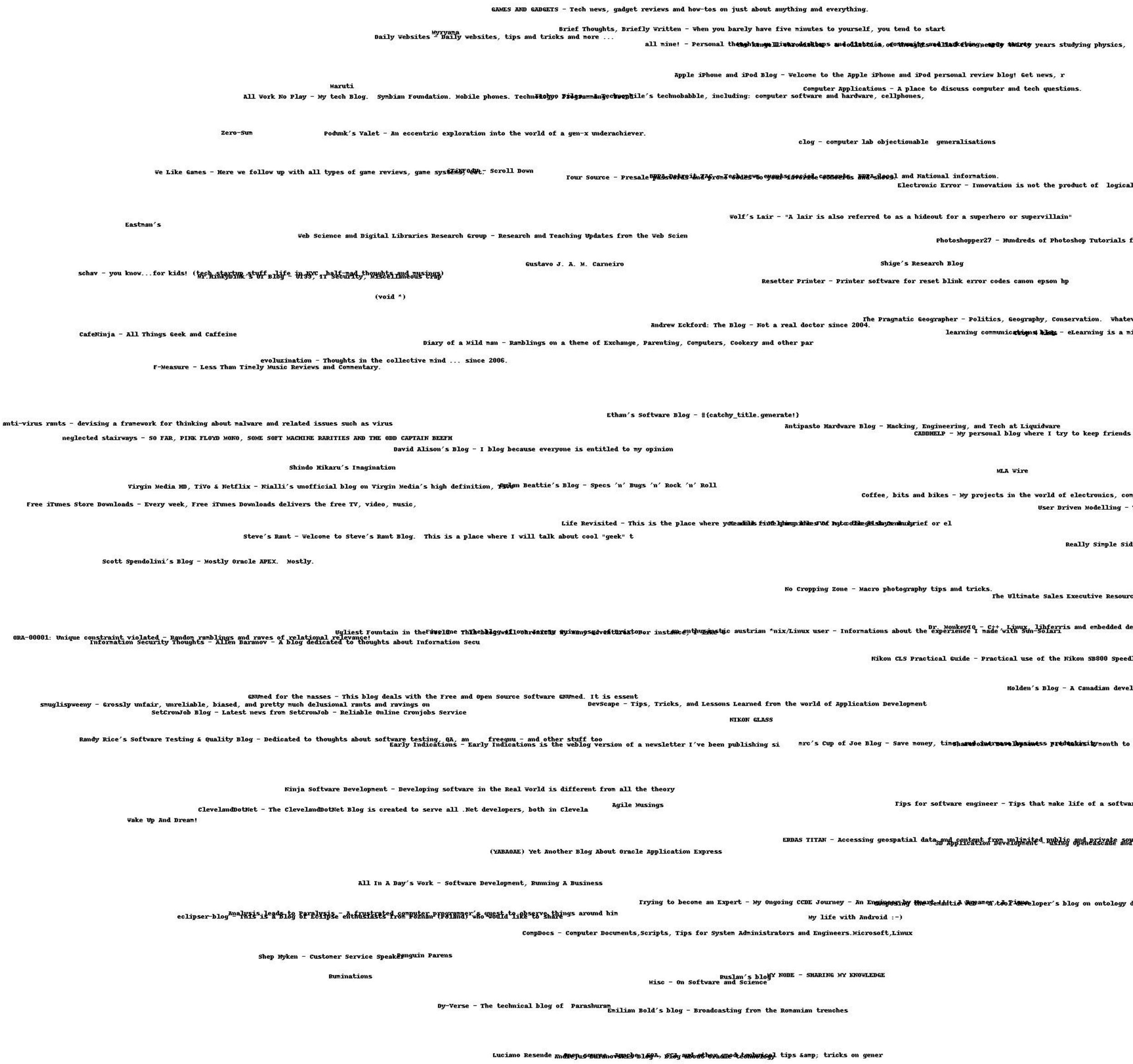

Figure 4.1: MDS

5 TFIDF

Re-run question 2, but this time with proper TFIDF calculations instead of the hack discussed on slide 7 (p. 32). Use the same 500 words, but this time replace their frequency count with TFIDF scores as computed in assignment #3. Document the code, techniques, methods, etc. used to generate these TFIDF values. Upload the new data file to github. Compare and contrast the resulting dendrogram with the dendrogram from question #2.

Note ideally you would not reuse the same 500 terms and instead come up with TFIDF scores for all the terms and then choose the top 500 from that list, but I'm trying to limit the amount of work necessary.

5.1 Solution

TF is calculated as the word frequency divided by the total words in the document. The IDF for each term was calculated as:

$$IDF(\text{term}) = log_2(\text{total docs in corpus}/\text{docs with term})$$

The total documents in corpus was used as the same as in Assignment Three, which was 42 billion. In order to get the total documents with the term, the python module requests was used to perform the Google search. Beautiful soup found the resultStats, which is where the number of results is stored. My first run was caught by Google's bot detection, so I added a random number sleep element. This took the program a little longer to run, but I was able to get all of the data. The IDF for each term was saved in a dictionary keyed on the term, in order to calculate the TFIDF.
Another blog matrix was created and saved as blog_tfidf.txt. For each word, the TF was calculated using the count from the wordcount dictionary for the term, which was by blog, multiplied by the length of the wordcount entry for the blog, which was the total number of words in the blog. Then TFIDF could be calculated as TF multiplied by IDF, and was rounded to three decimal points. [9]
The last step was to create the new dendrograms. It was similar to the previous dendrogram, if the blogs were not exactly next to each other after the TFIDF calculation, they were in close proximity. This tells me that the "hack" provided by the book was a pretty good estimation for the TFIDF for each blog. Figure 5.1 is the new JPEG dendrogram. Listing 8 is the python code used to accomplish this task.

```
google = 'http://www.google.com/search'
corp = 42000000000
idf = {}
for word in wordlist:
  print 'Searching for ' + word
  query_arg = {'q' : word}
  sdata = urllib.urlencode(query_arg)
  full_url = google+'?'+sdata
  r = requests.get(full_url)
  ddata = r.text
  soup = BeautifulSoup(ddata)
  results = soup.find('div', id='resultStats')
  res = (results.text).rsplit()
  term = float(res[1].replace(',',''))
  idf[word] = log((corp/term),2)
  sl = random.randint(1, 100)
  time.sleep(sl)

ti = file('blog_tfidf.txt', 'w', 0)
ti.write('Blog')
for word in wordlist:
  ti.write('\t%s' % word)

ti.write('\n')

for (blog, wc) in wordcounts.items():
  if titles[blog] != ’’:
    blog = blog + ’ - ’ + titles[blog]
  blog = blog.replace(u’\u0144’, ’n’)
  blog = _illegal_xml_chars_RE.sub(’ ’, blog)
  blog = blog.strip()
  if len(blog) > 100:
    blog = blog[:99]
  ti.write(blog)
  if len(blog) > 100:
    blog = blog[0:99]
  for word in wordlist:
    if word in wc:
      tf = float(wc[word]) / len(wc)
      tfidf = round(tf * idf[word], 3)
      ti.write(’\t’ + str(tfidf))
    else:
      ti.write(’\t0’)
  ti.write(’\n’)

old = sys.stdout
sys.stdout = open(’blog_tfidf_ascii.txt’, ’w’, 0)
blognames,words,data=clusters.readfile(’blog_tfidf.txt’)
clust = clusters.hcluster(data)
clusters.printclust(clust,labels=blognames)
sys.stdout = old
clusters.drawdendrogram(clust,blognames,jpeg=’tfidfclust.jpg’)
```

Listing 8: TFIDF Calculation

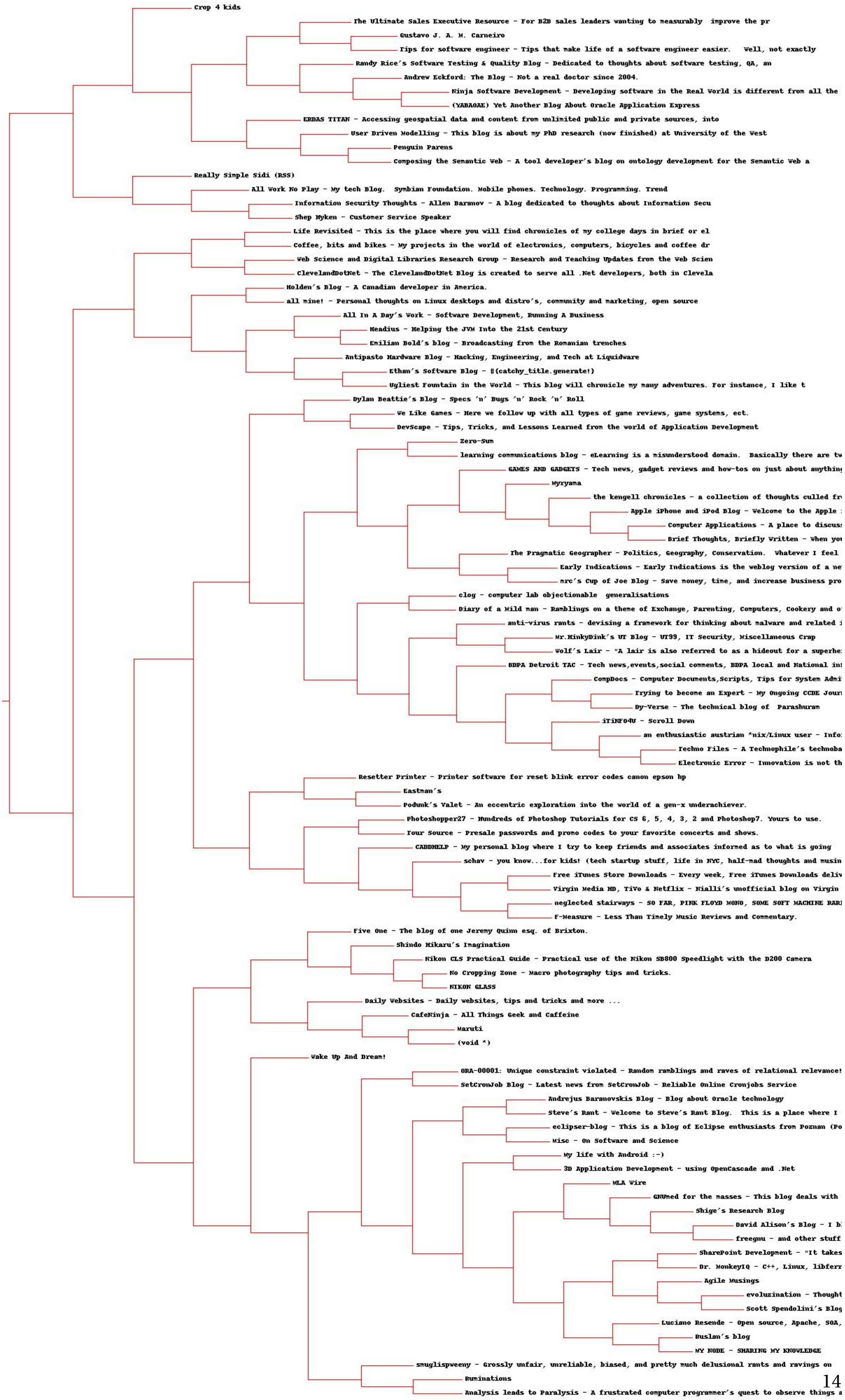
Crop 4 kids
The Ultimate Sales Executive Resource - For B2B sales leaders wanting to measurably improve the pr
Gustavo J. A. M. Carneiro
Tips for software engineer - Tips that make life of a software engineer easier.   Well, not exactly
Randy Rice's Software Testing & Quality Blog - Dedicated to thoughts about software testing, QA, an
Andrew Eckford: The Blog - Not a real doctor since 2004.
Ninja Software Development - Developing software in the Real World is different from all the
(YABAOAE) Yet Another Blog About Oracle Application Express
ERDAS TITAN - Accessing geospatial data and content from unlimited public and private sources, into
User Driven Modelling - This blog is about my PhD research (now finished) at University of the West
Penguin Parens
Composing the Semantic Web - A tool developer's blog on ontology development for the Semantic Web a
Really Simple Sidi (RSS)
All Work No Play - My tech Blog. Symbian Foundation. Mobile phones. Technology. Programming. Trend
Information Security Thoughts - Allen Baranov - A blog dedicated to thoughts about Information Secu
Shep Hyken - Customer Service Speaker
Life Revisited - This is the place where you will find chronicles of my college days in brief or el
Coffee, bits and bikes - My projects in the world of electronics, computers, bicycles and coffee dr
Web Science and Digital Libraries Research Group - Research and Teaching Updates from the Web Scien
ClevelandDotNet - The ClevelandDotNet Blog is created to serve all .Net developers, both in Clevela
Holden's Blog - A Canadian developer in America.
all mine! - Personal thoughts on Linux desktops and distro's, community and marketing, open source
All In A Day's Work - Software Development, Running A Business
Headius - Helping the JVM Into the 21st Century
Emilian Bold's blog - Broadcasting from the Romanian trenches
Antipasto Hardware Blog - Hacking, Engineering, and Tech at Liquidware
Ethan's Software Blog - #{catchy_title.generate!}
Ugliest Fountain in the World - This blog will chronicle my many adventures. For instance, I like t
Dylan Beattie's Blog - Specs 'n' Bugs 'n' Rock 'n' Roll
We Like Games - Here we follow up with all types of game reviews, game systems, ect.
DevScape - Tips, Tricks, and Lessons Learned from the world of Application Development
Zero-Sum
learning communications blog - eLearning is a misunderstood domain. Basically there are tw
GAMES AND GADGETS - Tech news, gadget reviews and how-tos on just about anything
Myryama
the kengell chronicles - a collection of thoughts culled fr
Apple iPhone and iPod Blog - Welcome to the Apple
Computer Applications - A place to discuss
Brief Thoughts, Briefly Written - When you
The Pragmatic Geographer - Politics, Geography, Conservation. Whatever I feel
Early Indications - Early Indications is the weblog version of a ne
mrc's Cup of Joe Blog - Save money, time, and increase business pro
clog - computer lab objectionable generalisations
Diary of a Mild man - Ramblings on a theme of Exchange, Parenting, Computers, Cookery and o
anti-virus rants - devising a framework for thinking about malware and related i
Mr.HinkyDink's UT Blog - UT99, IT Security, Miscellaneous Crap
Wolf's Lair - "A lair is also referred to as a hideout for a superhe
BDPA Detroit TAC - Tech news,events,social comments, BDPA local and National inf
CompDocs - Computer Documents,Scripts, Tips for System Admi
Trying to become an Expert - My Ongoing CCDE Jour
Dy-Verse - The technical blog of Parashuram
iTiNFO4U - Scroll Down
an enthusiastic austrian *nix/Linux user - Info
Techno Files - A Technophile's technoba
Electronic Error - Innovation is not th
Resetter Printer - Printer software for reset blink error codes canon epson hp
Eastman's
Podunk's Valet - An eccentric exploration into the world of a gen-x underachiever.
Photoshopper27 - Hundreds of Photoshop Tutorials for CS 6, 5, 4, 3, 2 and Photoshop7. Yours to use.
Tour Source - Presale passwords and promo codes to your favorite concerts and shows.
CADDHELP - My personal blog where I try to keep friends and associates informed as to what is going
schav - you know...for kids! (tech startup stuff, life in NYC, half-mad thoughts and musin
Free iTunes Store Downloads - Every week, Free iTunes Downloads deliv
Virgin Media HD, TiVo & Netflix - Nialli's unofficial blog on Virgin
neglected stairways - SO FAR, PINK FLOYD MONO, SOME SOFT MACHINE RARI
F-Measure - Less Than Timely Music Reviews and Commentary.
Five One - The blog of one Jeremy Quinn esq. of Brixton.
Shindo Hikaru's Imagination
Nikon CLS Practical Guide - Practical use of the Nikon SB800 Speedlight with the D200 Camera
No Cropping Zone - Macro photography tips and tricks.
NIKON GLASS
Daily Websites - Daily websites, tips and tricks and more ...
CafeNinja - All Things Geek and Caffeine
Maruti
(void *)
Wake Up And Dream!
ORA-00001: Unique constraint violated - Random ramblings and raves of relational relevance!
SetCronJob Blog - Latest news from SetCronJob - Reliable Online Cronjobs Service
Andrejus Baranovskis Blog - Blog about Oracle technology
Steve's Rant - Welcome to Steve's Rant Blog. This is a place where I
eclipser-blog - This is a blog of Eclipse enthusiasts from Poznan (Po
Misc - On Software and Science
My life with Android :-)
3D Application Development - using OpenCascade and .Net
MLA Wire
GNUmed for the masses - This blog deals with
Shige's Research Blog
David Alison's Blog - I b
freegnu - and other stuff
SharePoint Development - "It takes
Dr. MonkeyIQ - C++, Linux, libferr
Agile Musings
evoluzination - Thought
Scott Spendolini's Blog
Luciano Resende - Open source, Apache, SOA,
Ruslan's blog
MY NODE - SHARING MY KNOWLEDGE
smuglispweeny - Grossly unfair, unreliable, biased, and pretty much delusional rants and ravings on
Ruminations
Analysis leads to Paralysis - A frustrated computer programmer's quest to observe things a

Figure 5.1: TFIDF JPEG Dendogram

6 Python Code

```python
import requests
import pycurl
import re
from urlparse import urlparse
import urllib
import feedparser
from bs4 import BeautifulSoup
import clusters
import time
from math import log
import HTMLParser
try:
    from io import BytesIO
except ImportError:
    from StringIO import StringIO as BytesIO

def check(url):
  with open("blogs.txt") as f:
    found = False
    for line in f:  #iterate over the file one line at a time(memory efficient)
      if re.search(url, line):    #if string found is in current line then keep it
        found = True
  return found

blogger = 'https://www.blogger.com/next-blog?navBar=true&blogID=953024975153422094'
bfile = open('blogs.txt', 'w',0)
bfile.write('f-measure.blogspot.com\n')
bfile.write('ws-dl.blogspot.com\n')
bfile.close()
for i in range(1, 200):
  buffer = BytesIO()
  c = pycurl.Curl()
  c.setopt(c.URL, blogger)
  c.setopt(c.WRITEDATA, buffer)
  c.setopt(c.FOLLOWLOCATION, True)
  c.setopt(c.HTTPHEADER, ['Accept-Language: en'])
  bfile = open('blogs.txt', 'a',0)
  try:
    c.perform()
    if c.getinfo(c.RESPONSE_CODE) == 200:
      o = urlparse(c.getinfo(c.EFFECTIVE_URL))
      ch = check(o.netloc)
      if not ch:
        bfile.write(o.netloc + '\n')
        bfile.close()
      else:
        continue
    c.close()
  except pycurl.error, error:
    errno, errstr = error
    print 'An error occurred: ', errstr



def getwordcounts(url):
    #Returns title and dictionary of word counts for an RSS feed
    # 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 != '']

def getFeed(feedurl):
    global wordcounts
    global apcount
    try:
        (title, wc) = getwordcounts(feedurl)
        if title in wordcounts:
            for (w, c) in wc.iteritems():
                if w in wordcounts[title].iteritems():
                    wordcounts[title][w] += c
                else:
                    wordcounts[title][w] = c
        else:
            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

pages = {}
feedlist = []
apcount = {}
wordcounts = {}
wordlist = []
scheme = 'http://'
path = '/feeds/posts/default'
query_arg = {'alt' : 'atom'}
udata = urllib.urlencode(query_arg)
f = open('blogs.txt', 'r', 0)
for line in f:
  line = line.strip()
  data = urllib.urlencode(query_arg)
  full_url = scheme+line+path+"?"+udata
  feedlist.append(full_url)
  r = requests.get(full_url)
  getFeed(full_url)
  ddata = r.text
  soup = BeautifulSoup(ddata)
  next = soup.find('link', rel='next')
  count = 1
  while next:
    n= next.get('href')
    getFeed(n)
    r = requests.get(n)
    data = r.text
    soup = BeautifulSoup(data)
    count = count + 1
    if count % 10 == 0:
      print ' -- parsing ' + line + ' ' + str(count) + ' pages so far'
    next = soup.find('link', rel='next')
  print 'Finished parsing ' + line + ' ' + str(count) + ' pages'
  if str(count) in pages:
    pages[str(count)].append(line)
  else:
    pages[str(count)] = []
    pages[str(count)].append(line)
f.close()

p = open('blog_pages.txt', 'w', 0)
p.write('Pages\tNumber of Blogs\n')
for (pg, num) in pages.iteritems():
  p.write(pg + '\t' + str(len(num)) + '\n')
```

```

a = sorted(apcount.items(), key=lambda x: x[1], reverse=True)
count = 0
for (w, bc) in a:
    frac = float(bc) / len(feedlist)
    if frac > 0.1 and frac < 0.5:
        wordlist.append(w)
        count = count + 1
        if count >= 500:
            break

_illegal_xml_chars_RE = re.compile(u'[\x00-\x08\x0b\xa9\x0c\x0e-\x1F\uD800-\uDFFF\uFFFE\uFFFF]')

def remove_tags(text):
    cleanr =re.compile('<.*?>')
    cleantext = re.sub(cleanr,'', text)
    cleantext = re.sub("\n", ' ', cleantext)
    cleantext = cleantext.strip()
    return _illegal_xml_chars_RE.sub(' ', cleantext)

def getTitles():
  titles = {}
  global scheme, path, udata
  f = open('blogs.txt', 'r', 0)
  for line in f:
    line = line.strip()
    full_url = scheme+line+path+"?"+udata
    d = feedparser.parse(full_url)
    titles[d.feed.title] = remove_tags(d.feed.subtitle)
  return titles

titles = {}
titles = getTitles()
# Blog Term Matrix
out = file('blogdata.txt', 'w', 0)
out.write('Blog')
for word in wordlist:
    out.write('\t%s' % word)
out.write('\n')

for (blog, wc) in wordcounts.items():
    if titles[blog] != '':
        blog = blog + ' - ' + titles[blog]
    blog = blog.replace(u'\u0144', 'n')
    blog = _illegal_xml_chars_RE.sub(' ', blog)
    blog = blog.strip()
    if len(blog) > 100:
        blog = blog[:99]
    print blog
    out.write(blog)
    for word in wordlist:
        if word in wc:
            out.write('\t%d' % wc[word])
        else:
            out.write('\t0')
    out.write('\n')
out.close()
# Ascii
old = sys.stdout
sys.stdout = open('blog_ascii.txt', 'w', 0)
blognames,words,data=clusters.readfile('blogdata.txt')
clust = clusters.hcluster(data)
clusters.printclust(clust,labels=blognames)
sys.stdout = old
# jpeg dendogram
clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')
# K-clustering
def printClusters(kclust, out):
  for i in range(0,len(kclust)):
    klist = [blognames[r] for r in kclust[i]]
    for x in range(0, len(klist)):
      out.write(klist[x] + '\n')
    out.write('\n')
```

```

k = open('blog_k.txt', 'w', 0)
k.write('K = 5\n')
kclust=clusters.kcluster(data,k=5)
printClusters(kclust, k)
k.write('K = 10\n')
kclust=clusters.kcluster(data,k=10)
printClusters(kclust, k)
k.write('K = 20\n')
kclust=clusters.kcluster(data,k=20)
printClusters(kclust, k)
k.close()
# MDS
coords = clusters.scaledown(data)
clusters.draw2d(coords,blognames,jpeg='blogs2d.jpg')
#
#Extra credit - TFIDF
#
google = 'http://www.google.com/search'
corp = 42000000000
idf = {}
for word in wordlist:
  print 'Searching for ' + word
  query_arg = {'q' : word}
  sdata = urllib.urlencode(query_arg)
  full_url = google+'?'+sdata
  r = requests.get(full_url)
  ddata = r.text
  soup = BeautifulSoup(ddata)
  results = soup.find('div', id='resultStats')
  res = (results.text).rsplit()
  term = float(res[1].replace(',',''))
  idf[word] = log((corp/term),2)
  sl = random.randint(1, 100)
  time.sleep(sl)

ti = file('blog_tfidf.txt', 'w', 0)
ti.write('Blog')
for word in wordlist:
  ti.write('\t%s' % word)

ti.write('\n')

for (blog, wc) in wordcounts.items():
  if titles[blog] != '':
    blog = blog + ' - ' + titles[blog]
  blog = blog.replace(u'\u0144', 'n')
  blog = _illegal_xml_chars_RE.sub(' ', blog)
  blog = blog.strip()
  if len(blog) > 100:
    blog = blog[:99]
  ti.write(blog)
  if len(blog) > 100:
    blog = blog[0:99]
  for word in wordlist:
    if word in wc:
      tf = float(wc[word]) / len(wc)
      tfidf = round(tf * idf[word], 3)
      ti.write('\t' + str(tfidf))
    else:
      ti.write('\t0')
  ti.write('\n')

old = sys.stdout
sys.stdout = open('blog_tfidf_ascii.txt', 'w', 0)
blognames,words,data=clusters.readfile('blog_tfidf.txt')
clust = clusters.hcluster(data)
clusters.printclust(clust,labels=blognames)
sys.stdout = old
clusters.drawdendrogram(clust,blognames,jpeg='tfidfclust.jpg')
```

Listing 9: Complete Feed Parser

References

[1] Alan Gauld. how to redirect sys.stdout back to screen in IDLE. https://mail.python.org/pipermail/tutor/2002-November/018792.html. Accessed: 2014-11-20.

[2] Ashwini Chaudhary. Python: Checking if string is in a text file. http://stackoverflow.com/questions/16432203/python-checking-if-string-is-in-a-text-file. Accessed: 2014-9-20.

[3] Beautiful soup documentation. http://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: 2014-10-03.

[4] c24b. Python code to remove HTML tags from a string. http://stackoverflow.com/questions/9662346/python-code-to-remove-html-tags-from-a-string. Accessed: 2014-11-20.

[5] Kurt McKee. Common Atom Elements. https://pythonhosted.org/feedparser/common-atom-elements.html. Accessed: 2014-11-20.

[6] Leo Simons. stripping illegal characters out of xml in python. http://lsimons.wordpress.com/2011/03/17/stripping-illegal-characters-out-of-xml-in-python/. Accessed: 2014-11-20.

[7] Philip Southam. 'too many values to unpack', iterating over a dict. key=>string, value=>list. http://stackoverflow.com/questions/5466618/too-many-values-to-unpack-iterating-over-a-dict-key-string-value-list. Accessed: 2014-11-20.

[8] HTTP with Python âĂŞ PycURL by Example. http://www.angryobjects.com/2011/10/15/http-with-python-pycurl-by-example/. Accessed: 2014-11-20.

[9] Python Software Foundation. Built-in Functions. https://docs.python.org/2/library/functions.html. Accessed: 2014-11-20.

[10] Toby Segaran. *Programming Collective Intelligence*. O'Reilly Media, 2007.

[11] tutorialspoint. Python Strings. http://www.tutorialspoint.com/python/python_strings.htm. Accessed: 2014-11-20.