Ryan Condotta

Final Report Assignment 9

Question 1:

In order to compute the nearest neighbor for both links as shown in Figure 1, I created the python script of A10Q1.py. In this script, we compute the cosine distance metric from the 500 dimensions of our blogdata from Assignment 8.

http://f-measure.blogspot.com/
http://ws-dl.blogspot.com/
Figure 1

The first step in accomplishing the task was to be able to get the data from the text file and store them into the relative lists that are needed. This was accomplished through the function of readfile which would take in a file and would store the row names, column names, and data from that file. I then had to create the cosine distance metric to determine the similarities between dimensions of vectors. This was accomplished through the file of Cosine which is shown in Figure 2.

```
def Cosine(v1,v2):
    sumxx, sumxy, sumyy = 0, 0, 0
    for i in range(len(v1)):
        x = v1[i]; y = v2[i]
        sumxx += x*x
        sumyy += y*y
        sumxy += x*y
    return 1-(sumxy/math.sqrt(sumxx*sumyy))
```

Figure 2

The last step was to borrow the getdistances and knnestimate functions from the Programming Collective Intelligence book and make minor changes to them. These functions are shown in Figure 3. The knnestimate function calculates the group of nearest neighbors and uses the getdistances function to get the similarities of the two vectors being compared.

```
def getdistances(data,vec1):
    distancelist=[]
    for i in range(len(data)):
        vec2=data[i]
        distancelist.append((Cosine(vec1,vec2),i))
    distancelist.sort( )
    return distancelist

def knnestimate(data,vec1,k=3):
    # Get sorted distances
    dlist=getdistances(data,vec1)
    avg=0.0
    # Take the average of the top k results
    for i in range(k):
        idx=dlist[i][1]
        avg+= idx
        #avg+=data[idx]
    avg=avg/k
    return avg
```

Figure 3

The results of the knnestimate function is shown in the table in Figure 4 for K=1,2,5,10,and 20 where the F-Measure is 44 and ws-dl.blogspot is 50. As you keep the nearest neighbor doesn't range very much for for F-Measure for K=1 but the difference between K=1 and K = 2 and 5 is 7-11 more and continues to increase. Ws-dl.blogspot is much of the same way except the nearest neighbor for K=1 is a lot more compared to F-Measure.

Name	K = 1	K = 2	K = 5	K = 10	K = 20
http://f-measure.blogspot.com	43	36	32.8	28.3	25.15
http://ws-dl.blogspot.com/	54	56.33	44.6	33.9	29.9

Figure 4

Question 2:

In order to do the cross validation process and rerun A9 Question 2 using libsvm, we had to first start by collecting the data that would be used. The data we used in found in blogdata3.txt where I created a 500 term vector for each entry based off of 500 words. These were created from the combination of python scripts of A10Q2_1 (Figure 4) and A10Q2_2.py (Figure 5). I first went through the whole url and collected terms that could be used for each entry. Then, when I had the terms I searched through each of the entry and the summary and found the word count for each of the words in those entries.

```
def readfile(filename):
  #lines=[line for line in file(filename)]
  lines=[]
  for line in open(filename):
    lines.append(line)

  # 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

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 getwordcounts(url, words):
  # Parse the feed
  d=feedparser.parse(url)
  wording = {}
  # Loop over all the entries
  for e in d.entries:
    wc={}
    if 'summary' in e: summary=e.summary
    else: summary=e.description

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

blognames,words,data=readfile(r'C:\Users\Ryan\Documents\WebScience\Assignment10\blogdata2.txt')
wordcounts={}
wording = getwordcounts(r"http://theworldsfirstinternetbaby.blogspot.com/feeds/posts/default?max-
results=100&alt=rss", words)
wordcounts = wording

filename2 = r"C:\Users\Ryan\Documents\WebScience\Assignment10\blogdata3.txt"
out=open(filename2,'a')
out.write('Blog')
for word in words: out.write('\t%s' % word)
out.write('\n')
for blog,wc in wordcounts.items():
  try:
    print(str(blog))
    out.write(str(blog))
  except:
    out.write(str(blog.encode('utf-8')))
  for word in words:
    if word in wc: out.write('\t%d' % wc[word])
    else: out.write('\t0')
  out.write('\n')
```

Figure 4- A10Q2_1.py

```
import feedparser
import re

# Returns title and dictionary of word counts for an RSS feed
def getwordcounts(url):
  # Parse the feed
  d=feedparser.parse(url)
  wc={}
  entrytitle ={}
  # Loop over all the entries
  for e in d.entries:
    if 'summary' in e: summary=e.summary
    else: summary=e.description

    entrytitle.append(e.title.encode('utf-8'))
    # Extract a list of words
    words=getwords(e.title+' '+summary)
    for word in words:
      wc.setdefault(word,0)
      wc[word]+=1
  return entrytitle,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!='']


apcount={}
wordcounts={}
#filename1 = r"C:\Users\Ryan\Documents\WebScience\Assignment8\RSS.txt"
#filename1 = r"http://theworldsfirstinternetbaby.blogspot.com/feeds/posts/default?max-
results=100&alt=rss"
feedlist=[]
#for line in open(filename1):
feedlist.append("http://theworldsfirstinternetbaby.blogspot.com/feeds/posts/default?max-
results=100&alt=rss")


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)
print(len(apcount))

wordlist=[]
#get 500 words in the range
for w,bc in apcount.items():
  frac=float(bc)/len(feedlist)
  if len(wordlist) <= 500:
    #if frac>0.05 and frac<0.99:
    wordlist.append(w)
print(len(wordlist))

filename2 = r"C:\Users\Ryan\Documents\WebScience\Assignment10\blogdata2.txt"
out=open(filename2,'a')
out.write('Blog')
for word in wordlist: out.write('\t%s' % word)
out.write('\n')
for blog,wc in wordcounts.items():
  try:
    print(str(blog))
    out.write(str(blog))
  except:
    out.write(str(blog.encode('utf-8')))
  for word in wordlist:
    if word in wc: out.write('\t%d' % wc[word])
    else: out.write('\t0')
  out.write('\n')
```

Figure 5- A10Q2_2.py

Since I was able to get the data for each of the words, I could not begin the process of using libsvm, however, libsvm did not work on my computer. I found an alternative in scikit-learn which accomplished the same goal as libsvm. So I created the python script of A10Q2.py where it would read the data, as well as use the classifier information that I had to manually do for the first 50 to get the results of all the classifier guesses. Then, we plug the information and data into the support vector machine and then use cross validation tool that scikit-learn provides. Figure 6 shows the code used in this process.

```
import docclass
import feedparser
import re
```

```
from sklearn import svm
from sklearn import cross_validation
import numpy as np

def remove_html_tags(data):
  p = re.compile(r'<.*?>')
  return p.sub('', data)

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 readfile(feed, fisherclassifier):
        # Get feed entries and loop over them
        title =[]
        guess =[]
        actual=[]
        Yvalue = []
        f=feedparser.parse(feed)
        print(len(f))
        count = 0
        for entry in f['entries']:
                count = count + 1
                print(count)
                print ('-----')
                # Print the contents of the entry
                print ('Title: '+str(entry['title'].encode('utf-8')))
                print()
                fulltext = remove_html_tags(entry['title'])
                title.append(str(fulltext))
                #fulltext = fulltext +" " +remove_html_tags(entry['summary'])
                # Print the best guess at the current category
                if count < 20:
                        #print('Guess: '+str(fulltext.encode('utf-8')))
                        value = str(fisherclassifier.classify(fulltext))
                        print('Guess: '+ value)
                        if value == 'None':
                                Yvalue.append(0)
                        else:
                                Yvalue.append(int(value))
```

```
				# Ask the user to specify the correct category and train on that
				temp = raw_input('Enter Category:')
				print("value: ",temp)
				actual.append(int(temp))
				fisherclassifier.train(fulltext,temp)
			else:
				value1 = str(fisherclassifier.classify(fulltext))
				print(value1)
				actual.append(int(temp))
			print()

		return actual

def readVector(filename):
	#lines=[line for line in file(filename)]
	lines=[]
	for line in open(filename):
		lines.append(line)

	# 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

c2=docclass.fisherclassifier(docclass.getwords)
blognames,words,data=readVector(r'C:\Users\Ryan\Documents\WebScience\Assignment10\blogdata3.
txt')
Yvalue = readfile("http://theworldsfirstinternetbaby.blogspot.com/feeds/posts/default?max-
results=100&alt=rss", c2)


X_digits = np.array(data)
Y_digits = np.array(Yvalue)

clf = svm.SVC(kernel='linear', C=10)
clf.fit(X_digits, Y_digits)
scores = cross_validation.cross_val_score(clf, X_digits, Y_digits, cv = 10)
print(scores.mean())
for i in scores:
	print("Value:", i)
```

Figure 6

After the information was plugged in to the support vector machine, the results are shown in Figure 7 for the each of the categories. Each of these categories were ran such as Rock N Roll and Not Rock N Roll.. etc. As you can see from the table, Metal had the higgest percentages across the board for each of the categories while Rock N Roll and Other followed, and Alternative with the lowest percentages.

Type	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Attempt 6	Attempt 7	Attempt 8	Attempt 9	Attempt 10
Rock N Roll	0.9	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Metal	0.909	0.909	0.909	1	1	1	1	1	1	1
Other	0.909	0.909	0.909	0.909	1	1	1	1	0.888	1
Alternative	0.818	0.7272	0.9	0.9	0.9	0.9	0.8	0.8	0.77	0.88

Figure 7

Question 3:

I reran question A2Q2 and the python script again in order to download the 1000 TimeMaps to observe a change in the TimeMaps. All the information is stored in the excel file Histogram2.xls and displays the old and new URIs next to each other as well as the difference in the number of mementos. I took the difference information and created the graph as shown in Figure 8 where the x-axis is the uri number and the y-axis is the difference. From the majority of my URIs, it seemed like they stayed the same but more negative values are present. This change from the beginning of the semester to where we are now just shows how the URIs have changed in a matter of a few months. The TimeMap information is stored in A2Q2 folder where is shows the list of URIs I used for this assignment.

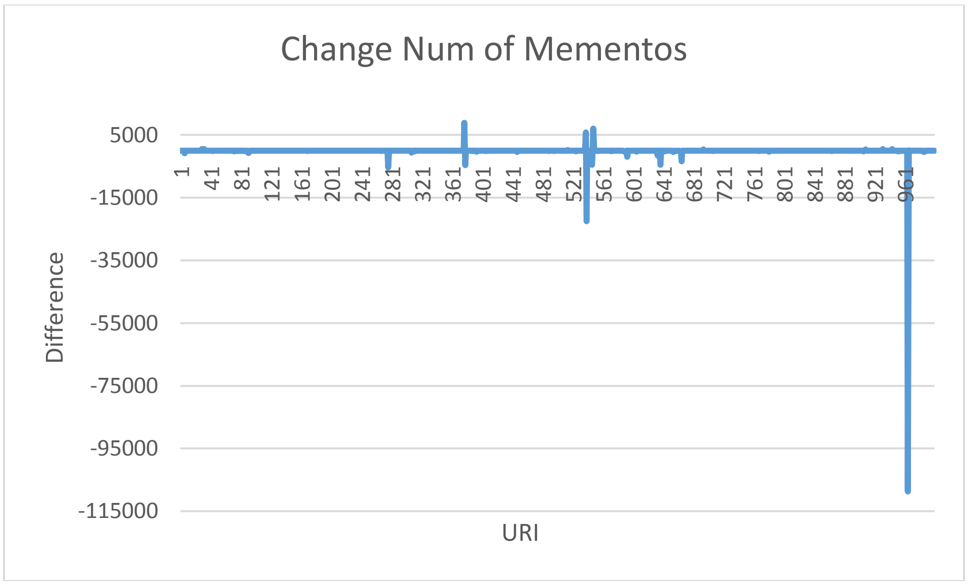

Figure 8

Question 4: