CS532 Web Science: Assignment 10

Finished on April 30, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

List of Figures

Listings

Problem 1

Question

Using the data from A8:

- Consider each row in the blog-term matrix as a 500 dimension vector, corresponding to a blog.

- From chapter 8, replace numpredict.euclidean() with cosine as the distance metric. In other words, you'll be computing the cosine between vectors of 500 dimensions.

- Use knnestimate() to compute the nearest neighbors for both:

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

for k=1,2,5,10,20.

Answer

A function for the cosine similarity was made based off the definition of the dot product. This was used with the nnestimate() function

```
#! /usr/bin/python

import math

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

def getdistances(data,vec1):
  distancelist=[]

  # Loop over every item in the dataset
  for i in range(len(data)):
    vec2=data[i]

    try:
      distancelist.append((cos_sim(vec1,vec2),i))
    except:
      pass

  # Sort by distance
  distancelist.sort()
  return distancelist

def knnestimate(data,vec1,k=5):
  # Get sorted distances
  dlist=getdistances(data,vec1)
```

```
    avg=0.0
    return dlist
vecs = {}

f = open("blogdata2.txt", "r")

for line in f:
    a = line.strip('\n').split('\t');
    b = a.pop(0)
    vecs[b] = a

print len(vecs)

fm = 'F-Measure'
ws = 'Web Science and Digital Libraries Research Group'

a = vecs[fm]
temp = vecs.values()
temp.pop(vecs.keys().index(fm))

a = knnestimate(temp,a,k=5)


k = [1, 2, 5, 10, 20]
print "-------F-Measure kNN-------"
for i in k:
  print "---k = "+str(i)
  for j in range(i):
    b = a[j][1]
    print vecs.keys()[b]

a = vecs[ws]
temp = vecs.values()
temp.pop(vecs.keys().index(ws))

a = knnestimate(temp,a,k=5)


k = [1, 2, 5, 10, 20]
print "------WS-DL kNN------"
for i in k:
  print "---k = "+str(i)
  for j in range(i):
    b = a[j][1]
    print vecs.keys()[b]
```

Listing 1: Python script that computes kNN based on cosine similarity

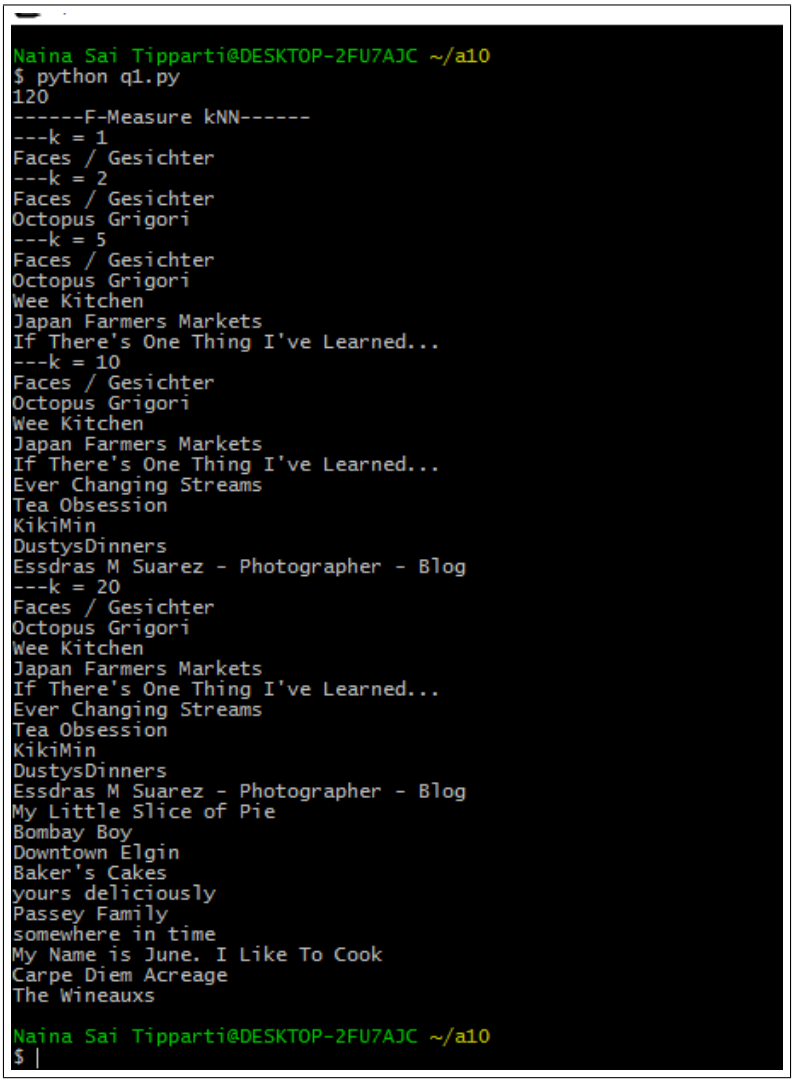

Figure 1: Clustings for the F-Measure blog

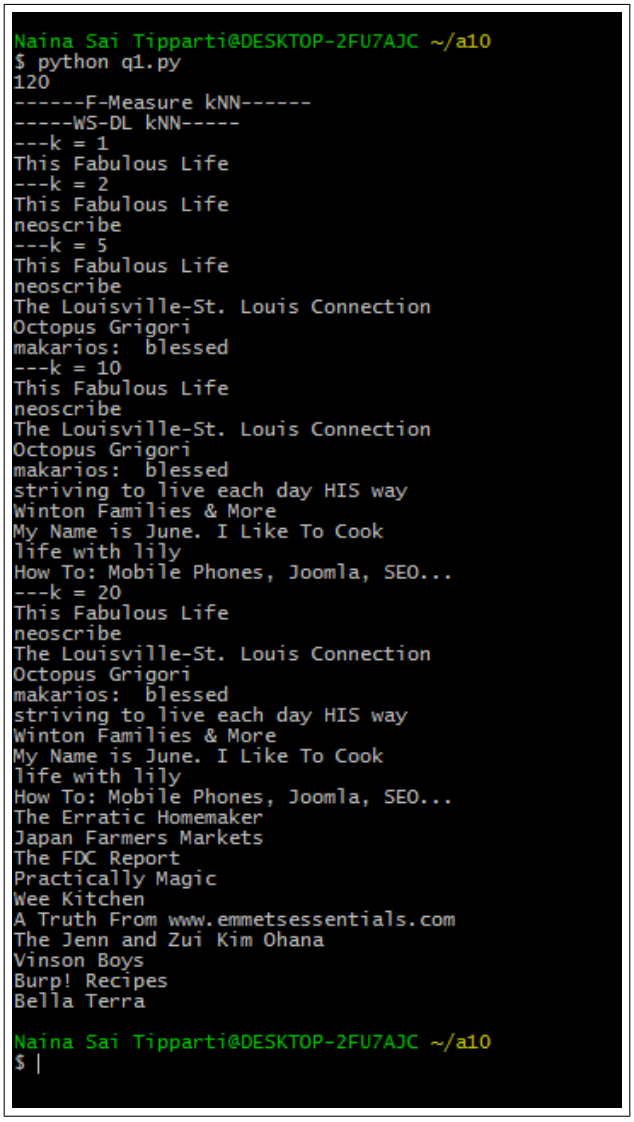

Figure 2: Clusterings for the WS-DL blogs

Problem 2

Question

Rerun A9, Q2 but this time using LIBSVM. If you have n categories, you'll have to run it n times. For example, if you're classifying music and have the categories:

metal, electronic, ambient, folk, hip-hop, pop

you'll have to classify things as:

metal / not-metal
electronic / not-electronic
ambient / not-ambient

etc.

Use the 500 term vectors describing each blog as the features, and your mannally assigned classifications as the true values. Use 10-fold cross-validation (as per slide 46, which shows 4-fold cross-validation) and report the percentage correct for each of your categories.

Answer

The `docclass.py` script was driven by the code shown in Listing 2.

```
entries = matrix.load_data(matrix.data_file)
cl = fisherclassifier(getwords)
cl.setdb('data.db')

T_HEAD = """\\begin{table}[h!]
\centering
\\begin{tabular}{| l | l | l | l |}
\hline
Entry Title & Actual & Predicted & cprob \\\\
\hline
"""

T_TAIL = """\hline
\end{tabular}
\caption{Question 2: Predictions }
\label{tab:mratings}
\end{table}
"""

def trainfrom(index=0):
  keys = training.keys()
  for key in keys[index:index+50]:
    cl.train(key, training[key])
  t = set(training.keys()[index:index+50])
  k = set(entries)
  rest = k - t
  predict = {}
  for item in rest:
    group, prob = cl.classify(item)
    predict[item] = (group, prob)
```

```
  with open('predict' + str(index), 'w') as outfile:
    outfile.write(T_HEAD)
    for item, tup in predict.iteritems():
      title = item.replace('&', '\\&').replace('#', '\\#')
240   row = ' & '.join([title, training[item], tup[0], str(tup[1])])
      outfile.write(row + ' \\\\\n')
    outfile.write(T_TAIL)

if __name__ == '__main__':
245 with open('training') as infile:
    training = {line.split('\t')[0]: line.split('\t')[1].strip() for line in infile}
  trainfrom(0)
  trainfrom(50)
```

Listing 2: docclass main

Problem 3

Question

Re-download the 1000 TimeMaps from A2, Q2. Create a graph where the x-axis represents the 1000 TimeMaps. If a TimeMap has "shrunk", it will have a negative value below the x-axis corresponding to the size difference between the two TimeMaps. If it has stayed the same, it will have a "0" value. If it has grown, the value will be positive and correspond to the increase in size between the two TimeMaps.

As always, upload all the TimeMap data. If the A2 github has the original TimeMaps, then you can just point to where they are in the report.

Answer

The python script in Listing 3 was used to retrieve the timemaps and then parse the returned html, traveling down the rabbit hole if the target URI has more than 1000 mementos.

```
# -*- encoding: utf-8 -*-
#! /usr/bin/python

import requests
import re

MW_URI = "http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/"

if __name__ == '__main__':
    with open('output', 'r') as f:
        output = open('site_mementos', 'w')
        mementos = {}
        for uri in f.read().split('\n'):
            if uri is '':
                continue
            count = 0
            target_uri = MW_URI + uri
            while True:
                result = requests.get(target_uri)
                if result.ok:
                    count = count + result.text.count('rel="memento"')
                last_line = result.text.split('\n')[-1]
                if 'rel="timemap"' not in last_line:
                    break
                sites = re.findall(r'<([^<|^>]+)>', last_line)
                target_uri = sites[1]
            mementos[uri] = count
            print 'found %d mementos for uri: %s' % (count, uri)
            output.write('%s %d\n' % (uri, count))
    output.close()
```

Listing 3: mementofinder.py

The dataset created Listing 3. A log scale was used along the y-axis to show more detail among the results. The script in Listing 4 was used to create the histogram in Figure 3, which shows the difference of mementos per site from the dataset.

```
#! /usr/bin/Rscript
```

```
data <- read.table("D:/cs532/a10/q3/results", header=TRUE, comment.char="")
counts <- table(data$Mementos)
pdf("histogram.pdf")
barplot(counts, log="y", ylim=c(.75, nrow(data)), ylab="Memento Count Difference", xlab="
    Sites", main="Memento Count Difference per Site")
dev.off()
```

Listing 4: build_histogram.r

```
https://twitter.com/asksparati/status/696367298525978624 0
https://twitter.com/History_Futbol/status/694720215087652864/photo/1 0
http://www.dainikbhaskar.info/sports/kids-should-learn-from-soaring-leicester-not-superstars
    -lionel-messi-and-cristiano-ronaldo-they-show-the-value-of-hard-graft/ 0
https://twitter.com/Juezcentral/status/696515201743659008 0
http://www.oldpicsarchive.com/selected-photos-part-4-33-rare-pics/4/?utm_content=buffer840ce
    &utm_medium=social&utm_source=twitter.com&utm_campaign=buffer 0
http://www.georgewalkerbush.net/bushfamilyfundedhitler.htm 79
http://odia.ig.com.br/noticia/rio-de-janeiro/2016-02-03/justica-proibe-venda-e-divulgacao-de
    -livro-escrito-por-adolf-hitler.html 0
http://www.telegraph.co.uk/news/worldnews/donald-trump/12038640/Who-said-it-Donald-Trump-or-
    Adolf-Hitler.html 7
https://www.youtube.com/watch?v=d3r70E6Dvfs&feature=youtu.be&a 16
https://www.youtube.com/watch?v=sI1E4Vs7cbk&feature=youtu.be&a 16
http://www.newsweek.com/adolf-hitler-black-holocaust-dark-secrets-423735?rx=us 2
http://www.flimper.com/events/4 0
https://www.facebook.com/robcaiafa/posts/10208639359731659 0
http://linkis.com/www.youtube.com/lrpOF 0
https://www.youtube.com/watch?v=QnpBN-ltVtE&feature=youtu.be 16
http://daveschlueteronline.com/the-penguin-updates-and-seo/ 3
https://twitter.com/MrJohnQZombie/status/664324668149493760/photo/1 0
https://twitter.com/MrJohnQZombie/status/664324668149493760 0
http://www.nzherald.co.nz/entertainment/news/article.cfm?c_id=1501119&objectid=11586082&ref=
    rss&utm_source=dlvr.it&utm_medium=twitter 0
http://frtyb.com/go/Og1_bnwHp_j4D2/DEFAULT 0
https://www.youtube.com/watch?v=cZKeqenZbJk&feature=youtu.be&a 16
```

Listing 5: Sample of Memento Links

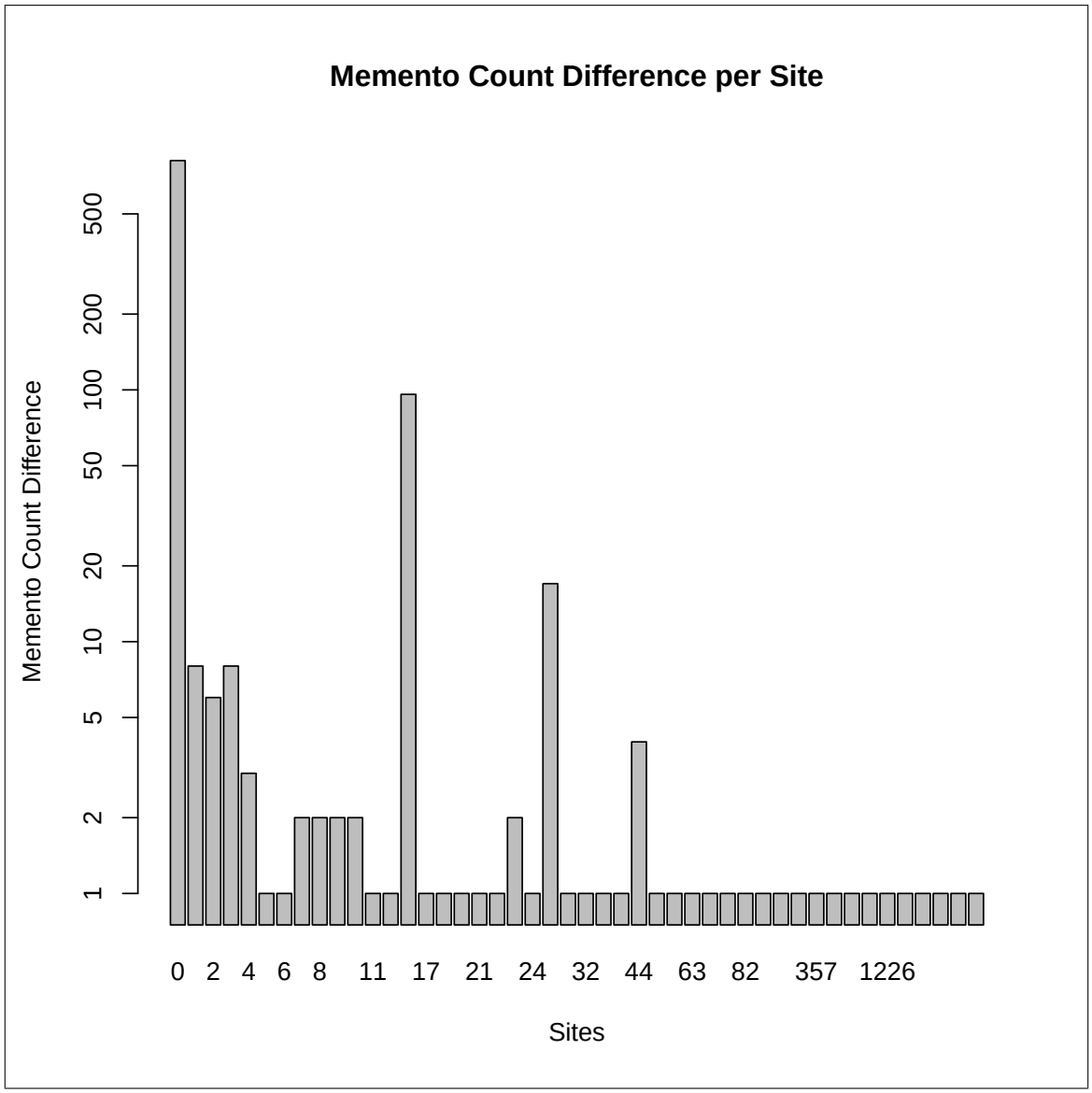

Figure 3: Histogram of Mementos Count Difference

Problem 4

Question

Repeat A3, Q1. Compare the resulting text from February to the text you have now. Do all 1000 URIs still return a "200 OK" as their final response (i.e., at the end of possible redirects)?

Create two graphs similar to that described in Q3, except this time the y-axis corresponds to difference in bytes (and not difference in TimeMap magnitudes). For the first graph, use the difference in the raw (unprocessed) results. For the second graph, use the difference in the processed (as per A3, Q1) results.

Of the URIs that still terminate in a "200 OK" response, pick the top 3 most changed (processed) pairs of pages and use the Unix "diff" command to explore the differences in the version pairs.

Answer

Using the python script in Listing 6, 1000 unique URIs were dereferenced and their raw contents were stored in the `html/raw/` folder as a file with the filename as the md5-hashed URI. These were then stripped of all html elements and their processed contents were stored in the `html/processed/` folder as the same md5-hashed filename. For reference, the URIs were written as the first line of each of their content files.

```
#! /usr/bin/python

import requests
import concurrent.futures
import md5
from bs4 import BeautifulSoup
import pickle

def convert(uri):
    return md5.new(uri).hexdigest()

def get_html(uri):
    print('Getting {}'.format(uri))
    response = requests.get(uri)
    return response.url, response.status_code, response.content

if __name__ == '__main__':
    with open('links') as infile:
        uris = [uri.rstrip('\n') for uri in infile]

    with concurrent.futures.ThreadPoolExecutor(max_workers=8) as executor:
        uri_futures = [executor.submit(get_html, uri) for uri in uris]
        for future in concurrent.futures.as_completed(uri_futures):
            try:
                uri, status_code, content = future.result()
            except Exception as exc:
                print('{} generated an exception: {}'.format(uri, exc))
                continue
            if status_code == 200:
                hashed_uri = convert(uri)
```

```
                print('Writing {} as {}'.format(uri, hashed_uri))
                try:
                    with open('html/raw/' + hashed_uri, 'w') as outfile:
                        outfile.write(uri + '\n')
                        outfile.write(content)
                    with open('html/processed/' + hashed_uri + '.processed.txt', 'w') as
                        outfile:
                        outfile.write(uri + '\n')
                        outfile.write(BeautifulSoup(content).get_text().encode('utf8'))
                except Exception as e:
                    print '**** ERROR **** ---- ' + uri
                    print e
            else:
                print('Not writing {}, bad status code: {}'.format(uri, status_code))
```

Listing 6: get_html.py

```
import os, sys

savefile = open('new_processed_size', 'a')
path = "html/processed/"

for filename in os.listdir(path):
    filepath = os.path.join(path, filename)
    size = os.path.getsize(filepath)
    savefile.write(str(size))
    savefile.write('\n')
    print size

savefile = open('new_raw_size', 'a')
path = "html/raw/"

for filename in os.listdir(path):
    filepath = os.path.join(path, filename)
    size = os.path.getsize(filepath)
    savefile.write(str(size))
    savefile.write('\n')
    print size

savefile = open('old_processed_size', 'a')
path = "old/html/processed/"

for filename in os.listdir(path):
    filepath = os.path.join(path, filename)
    size = os.path.getsize(filepath)
    savefile.write(str(size))
    savefile.write('\n')
    print size

savefile = open('old_raw_size', 'a')
path = "old/html/raw/"

for filename in os.listdir(path):
    filepath = os.path.join(path, filename)
    size = os.path.getsize(filepath)
    savefile.write(str(size))
    savefile.write('\n')
    print size
```

Listing 7: get_size.py

diff analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file in order to make it identical to the second file.

It does not actually change the files; however, it can optionally generate a script (with the -e option) for the program ed (or ex which can be used to apply the changes.
The -e option tells diff to output a script, which can be used by the editing programs ed or ex, that contains a sequence of commands. The commands are a combination of c (change), a (add), and d (delete) which, when executed by the editor, will modify the contents of file1 (the first file specified on the diff command line) so that it matches the contents of file2 (the second file specified).

```
diff -e file1 file2 > output
```

Listing 8: diff command

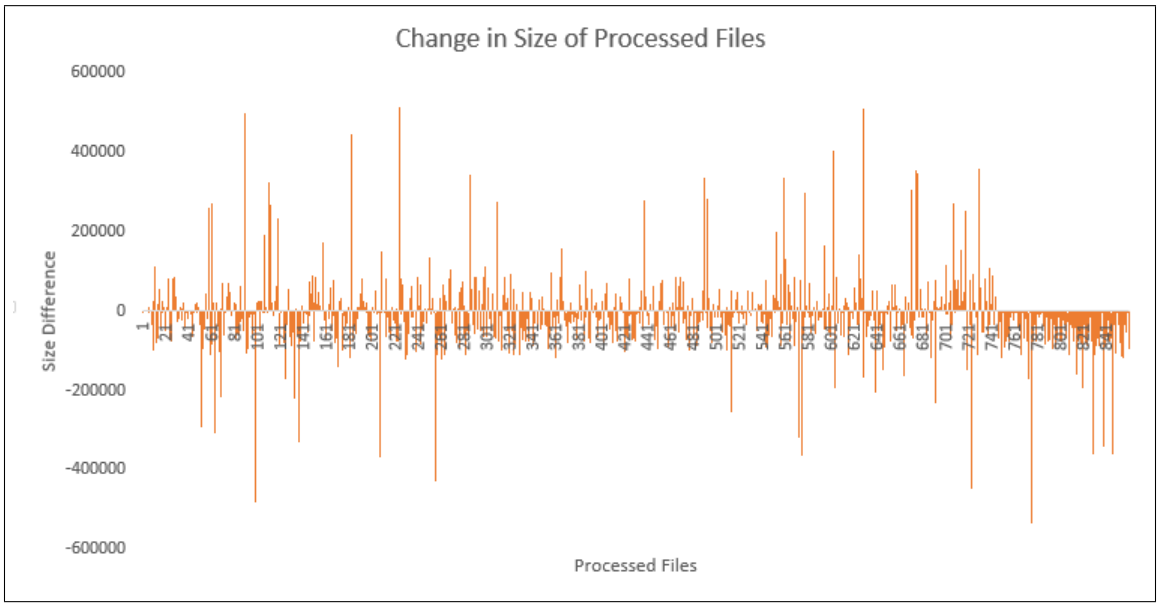

Figure 4: Change in Size of Processed Files

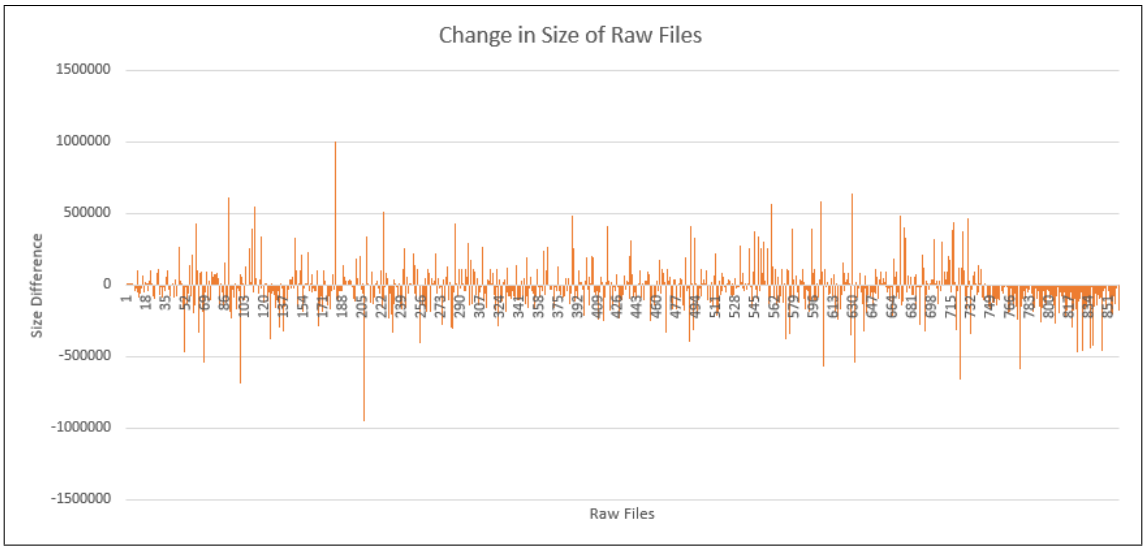

Figure 5: Change in Size of Raw Files