Assignment 10

CS 532: Introduction to Web Science

Spring 2017

Grant Atkins

Finished on May 1, 2017

1

Question

```
1.  Using the data from A8:

- Consider each row in the blog-term matrix as a
1000 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 1000 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

To solve this question I used the blog data I previously received in assignment 8 **blogdata.txt** and the code provided by the Programming Collective Intelligence book [2]. To calculate the cosine distance metric **numpredict.py** had to be modified to accompany this new function. The methods that changed in this file were the *knnestimate* and *getdistances* functions. In the *getdistances* function I simply swapped out the euclidean function for cosine as shown in Listing 1. In the *knnestimate* function I removed the average value and instead return a list of sorted distances in descending order.

I then went filtered the F-measure and Web Science research group's blogs and create separate vectors with their values. These vectors were used for the cosine measurement against the vector with all blogs. This is shown in Listing 2.

```
1
2  def cosine(v1, v2):
3      '''
4      cosine function for 1000 blog matrix. zip v1, v2
5      for array of tuples
6      '''
7      sumab = sum([a * b for a, b in zip(v1, v2)])
8      suma = sum([a * a for a in v1])
9      sumb = sum([b * b for b in v2])
10     return sumab / math.sqrt(suma * sumb)
11
12
13 def getdistances(data, vec1):
14     distancelist = []
15
16     # Loop over every item in the dataset
17     for i, vals in enumerate(data):
18         vec2 = vals
19
20         # Add the distance and the index
21         distancelist.append((cosine(vec1, vec2), i))
22
23     # Sort by distance
24     distancelist.sort(reverse=True)
25     return distancelist
26
27
28 def knnestimate(data, vec1, k=5):
29     # Get sorted distances
30     distances = getdistances(data, vec1)
31
32     # avg = 0.0
```

```
33      # Take the average of the top k results
34      # for i in range(k):
35      #     idx = dlist[i][1]
36      #     avg += data[idx]['result']
37      # avg = avg / k
38      return distances
```

Listing 1: Python script with included cosine function and knnestimate changes

```
1
2   from numpredict import *
3
4
5   def estimate(vectorValues, fmeasureVector, rgroupVector):
6       nn = knnestimate(vectors.values(), fmeasureVector)
7       print("==========" * 2)
8       print("K nearest neighbors of F-Measure")
9       print("==========" * 2)
10      kvals = [1, 2, 5, 10, 20]
11      for k in kvals:
12          print('k =', k)
13          for j in range(k):
14              print('%s\t%.6f' % (list(vectors.keys())[nn[j][1]],
                    nn[j][0]))
15
16          print("----------" * 2)
17      print()
18
19      print("==========" * 2)
20      print("K nearest neighbors of Web Science and Digital
            Libraries Research Group")
21      print("==========" * 2)
22      nn = knnestimate(vectors.values(), rgroupVector)
23      for k in kvals:
24          print('k =', k)
25          for j in range(k):
26              print('%s\t%.6f' % (list(vectors.keys())[nn[j][1]],
                    nn[j][0]))
27
28          print("----------" * 2)
29
30
31  def getData():
32      '''
33      get blogdata in tuples and add to specified blog arrays, or
34      new dictionary with blog as key
35      '''
36      fmeasure = 'F-Measure'
```

```
37      wlblog = 'Web Science and Digital Libraries Research Group'
38      vectors = {}
39      fmeasureVals = []
40      webrVals = []
41      with open("data/blogdata.txt", 'r') as f:
42          allLines = f.readlines()
43          for i, line in enumerate(allLines):
44              if i == 0:
45                  # skip header
46                  continue
47              tuples = line.strip().split('\t')
48              if tuples[0] == fmeasure:
49                  for i in range(1, len(tuples)):
50                      fmeasureVals.append(float(tuples[i]))
51              elif tuples[0] == wlblog:
52                  for i in range(1, len(tuples)):
53                      webrVals.append(float(tuples[i]))
54              else:
55                  vectors[tuples[0]] = []
56                  for i in range(1, len(tuples)):
57                      vectors[tuples[0]].append(float(tuples[i]))
58
59      return vectors, fmeasureVals, webrVals
60
61
62  if __name__ == "__main__":
63      vectors, vectorfm, vectorwb = getData()
64      estimate(vectors.values(), vectorfm, vectorwb)
```

Listing 2: Python script to find KNN values

```
================================================================================
K nearest neighbors of F-Measure
================================================================================
k = 1
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA      0.651976
--------------------------------------------------------------------------------
k = 2
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA      0.651976
Revolver USA Distribution & Midheaven mailorder 0.480294
--------------------------------------------------------------------------------
k = 5
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA      0.651976
Revolver USA Distribution & Midheaven mailorder 0.480294
Indie Top 20 - The Blog!        0.475896
The World's First Internet Baby 0.419790
.       0.418521
--------------------------------------------------------------------------------
k = 10
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA      0.651976
Revolver USA Distribution & Midheaven mailorder 0.480294
Indie Top 20 - The Blog!        0.475896
The World's First Internet Baby 0.419790
.       0.418521
MTJR RANTS & RAVES ON MUSIC     0.410171
On Warmer Music 0.398600
Eli Jace | The Mind Is A Terrible Thing To Paste        0.388213
DaveCromwell Writes     0.386476
Pithy Title Here        0.359736
--------------------------------------------------------------------------------
k = 20
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA      0.651976
Revolver USA Distribution & Midheaven mailorder 0.480294
Indie Top 20 - The Blog!        0.475896
The World's First Internet Baby 0.419790
.       0.418521
MTJR RANTS & RAVES ON MUSIC     0.410171
On Warmer Music 0.398600
Eli Jace | The Mind Is A Terrible Thing To Paste        0.388213
DaveCromwell Writes     0.386476
Pithy Title Here        0.359736
Steel City Rust 0.356205
A to Zappa - Song of the day    0.354112
Some Call It Noise....  0.353264
The Music Binge 0.351249
The Jeopardy of Contentment     0.346409
Encore  0.344751
The Girl at the Rock Show       0.340610
www.doginasweater.com Live Show Review Archive  0.339225
turnitup!       0.328807
Did Not Chart   0.304550
--------------------------------------------------------------------------------
```

Figure 1: F-Measure blog's KNNestimate output

```
========================================================================
K nearest neighbors of Web Science and Digital Libraries Research Group
========================================================================
k = 1
macthemost      0.395647
------------------------------------------------------------------------
k = 2
macthemost      0.395647
ORGANMYTH       0.363448
------------------------------------------------------------------------
k = 5
macthemost      0.395647
ORGANMYTH       0.363448
MarkFisher's-MusicReview        0.346194
ΜΕΣΑ ΣΤΗ ΒΡΩΜΙΑ 0.324367
juanbook        0.293966
------------------------------------------------------------------------
k = 10
macthemost      0.395647
ORGANMYTH       0.363448
MarkFisher's-MusicReview        0.346194
ΜΕΣΑ ΣΤΗ ΒΡΩΜΙΑ 0.324367
juanbook        0.293966
Eli Jace | The Mind Is A Terrible Thing To Paste        0.278686
Diagnosis: No Radio     0.266199
Pithy Title Here        0.259391
Myopiamuse      0.258883
Mile In Mine    0.255558
------------------------------------------------------------------------
k = 20
macthemost      0.395647
ORGANMYTH       0.363448
MarkFisher's-MusicReview        0.346194
ΜΕΣΑ ΣΤΗ ΒΡΩΜΙΑ 0.324367
juanbook        0.293966
Eli Jace | The Mind Is A Terrible Thing To Paste        0.278686
Diagnosis: No Radio     0.266199
Pithy Title Here        0.259391
Myopiamuse      0.258883
Mile In Mine    0.255558
Avidd Wallows' Blog     0.253298
hello my name is justin.        0.247136
Cherry Area     0.245448
A2 MEDIA COURSEWORK JOINT BLOG  0.245424
On Warmer Music 0.233990
Steel City Rust 0.233324
Revolver USA Distribution & Midheaven mailorder 0.230158
Some Call It Noise....  0.229648
.       0.226684
MTJR RANTS & RAVES ON MUSIC     0.225070
------------------------------------------------------------------------
```

Figure 2: Web Research group's KNNestimate output

2

Question

```
2.  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 1000 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

After reading many stackoverflow posts on LIBSVM, I decided to use the scikit-learn library to solve this question [3]. I think this was a better option due to the ease of training and performing cross validation. When creating the blog matrix I combined the 100 selected feeds title and summary to create the new term matrix. Due to the short amount of text in each description it should be noted that my matrix creating program, **createMatrix.py** shown in Listing 3, only created **898** terms instead of 1000.

The actual SVM code for is shown in Listing 4. It first iterates through the newly created feed data matrix and matches my manual classification. A dictionary is created to hold all the values, it is then passed to the *execSVM* function where it checks if the items have the category and performs SVM assigning values with 1 or -1 based if category was or was not the category assigned. Then finally performs cross validation folding up to 10 times.

Category	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Mean
Android	0.818182	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.888889	0.810707
iOS	0.818182	0.900000	0.600000	0.500000	0.900000	0.800000	0.800000	0.800000	0.900000	0.666667	0.768485
Realm News	0.727273	0.727273	0.727273	0.800000	0.800000	0.800000	0.800000	0.777778	0.777778	0.777778	0.771515
React Native	0.909091	0.909091	0.909091	0.909091	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.963636
Nodejs	0.909091	0.909091	0.909091	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.972727
Xamarin	0.909091	0.909091	0.909091	0.909091	0.909091	1.000000	1.000000	1.000000	1.000000	1.000000	0.954545
Databases	0.909091	0.909091	0.909091	0.909091	0.909091	1.000000	1.000000	1.000000	1.000000	1.000000	0.954545

Table 1: SVM with 10-fold cross validation

```
1
2  import feedparser
3  import re
4  import html
5
6
7  def remove_img_tags(data):
8      '''
9      Helper to remove img tags from description
10     '''
11     p = re.compile(r'<img.*?/>')
12     return p.sub('', data)
13
14
15 def remove_emojis(data):
16     '''
17     Helper to remove emojis from description
18     '''
19     emoji_pattern = re.compile("["
20                                u"\U0001F600-\U0001F64F"  #
                                       emoticons
21                                u"\U0001F300-\U0001F5FF"  #
                                       symbols & pictographs
22                                u"\U0001F680-\U0001F6FF"  #
                                       transport & map symbols
23                                u"\U0001F1E0-\U0001F1FF"  # flags
                                       (iOS)
24                                "]+", flags=re.UNICODE)
25     d = emoji_pattern.sub(r'', data)
26     return d
27
28
29 def getwordcounts(title, descrip):
30     wc = {}
31
32     # Extract a list of words
33     words = getwords(title + ' ' + descrip)
34     for word in words:
35         wc.setdefault(word.strip(), 0)
36         wc[word] += 1
37     return title, wc
38
39
40 def getwords(html):
41     # Remove all the HTML tags
42     txt = re.compile(r'<[^>]+>').sub('', html)
43
44     # Split words by all non-alpha characters
```

```
45      words = re.compile(r'[^A-Z^a-z]+').split(txt)
46
47      # Convert to lowercase
48      return [word.lower() for word in words if word != '']
49
50
51  apcount = {}
52  wordcounts = {}
53  f = feedparser.parse('data/feed.xml')
54  counter = 0
55  for i, entry in enumerate(f.entries):
56      t = html.unescape(entry["title"])
57      t = t.replace('"', '')
58      d = html.unescape(entry["description"])
59      d = remove_img_tags(d)
60      d = remove_emojis(d)
61      d = d.replace('"', '')
62
63      title, wc = getwordcounts(t, d)
64      wordcounts[title] = wc
65      for word, count in wc.items():
66          apcount.setdefault(word, 0)
67          if count > 1:
68              apcount[word] += 1
69      counter += 1
70      if counter >= 100:
71          break
72
73  wordlist = []
74  for w, bc in apcount.items():
75      frac = float(bc) / 100
76      # if frac>0.01 and frac<0.5:
77      wordlist.append(w)
78      if len(wordlist) >= 1000:
79          break
80
81  out = open('data/feedData.txt', 'w')
82  out.write('Blog')
83  for word in wordlist:
84      out.write('\t%s' % word)
85  out.write('\n')
86  for blog, wc in wordcounts.items():
87      # print blog
88      try:
89          out.write(blog)
90      except:
91          out.write(str(blog.encode('utf-8')))
92      for word in wordlist:
93          if word in wc:
```

```
94              out.write('\t%d' % wc[word])
95          else:
96              out.write('\t0')
97      out.write('\n')
```

Listing 3: Python script to create term matrix from Realm RSS feed 100 items

```
1
2  from sklearn import svm
3  from sklearn import cross_validation
4  import numpy as np
5
6
7  def execSVM(cat, data):
8      X = []
9      Y = []
10     for unit in data:
11         vec = data[unit]['vector']
12         X.append(vec)
13         if data[unit]['actual'] != cat:
14             Y.append(-1)
15         else:
16             Y.append(1)
17
18     dataX = np.array(X)
19     dataY = np.array(Y)
20     svc = svm.SVC(C=10)
21     svc.fit(dataX, dataY)
22     score = cross_validation.cross_val_score(svc, dataX, dataY,
           cv=10)
23     return score
24
25
26 def createTabular(cats, newsItems):
27     with open("../docs/crossVal.tex", 'w') as f:
28         f.write("\hline\n")
29         for cat in cats:
30             data = execSVM(cat, newsItems)
31             outStr = cat + " & "
32             mean = 0.0
33             for score in data:
34                 outStr += "%.6f & " % (score)
35                 mean += score
36             mean = mean / len(data)
37             outStr += "%.6f " % (mean)
38             outStr += "\\\\ \n"
39             f.write(outStr)
40
```

```
41
42 if __name__ == "__main__":
43     newsItems = {}
44     with open("data/feedData.txt") as f, open("data/
           classifiedFeeds.txt") as cf:
45         allLines = f.readlines()
46         cats = cf.readlines()
47         for i, line in enumerate(allLines):
48             if i == 0:
49                 # skip header
50                 continue
51             tuples = line.strip().split('\t')
52             feedTitle = tuples[0]
53             newsItems[feedTitle] = {}
54             newsItems[feedTitle]['vector'] = [float(wc) for wc
                   in tuples[1:]]
55             newsItems[feedTitle]['actual'] = cats[i - 1].rstrip
                   ()
56
57     cats = ['Android', 'iOS', 'Realm News',
58                 'React Native', 'Nodejs', 'Xamarin', '
                       Databases']
59
60     createTabular(cats, newsItems)
```

Listing 4: Python script to perform cross validation with SVM classifier

3

Question

(3 points extra credit)

3. 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

NOT ATTEMPTED

4

Question

```
(3 points extra credit)

4.  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 same code from A3 I downloaded the html files with the same URI list, which is also in my assignment 10 repository. The files to download the pages and process them are **download.sh** and textbfprocessHtml.py which aren't listed but are also in my repository. The only real file that required more work was done in two files, **fileSizeDiff.R** for histogram creation and **getFileSizes.py** which calculate the bytes for the old files processed and raw html files, as well as the new processed and raw html files.

I unfortunately did not have enough time to keep track of which URIs were actually still responding with a 200 response code. However I only included URIs in the charts below that still returned content and matched the final URI of the previous one found in assignment 3. The bar charts below indicate the old file size in bytes minus the new file size in bytes.

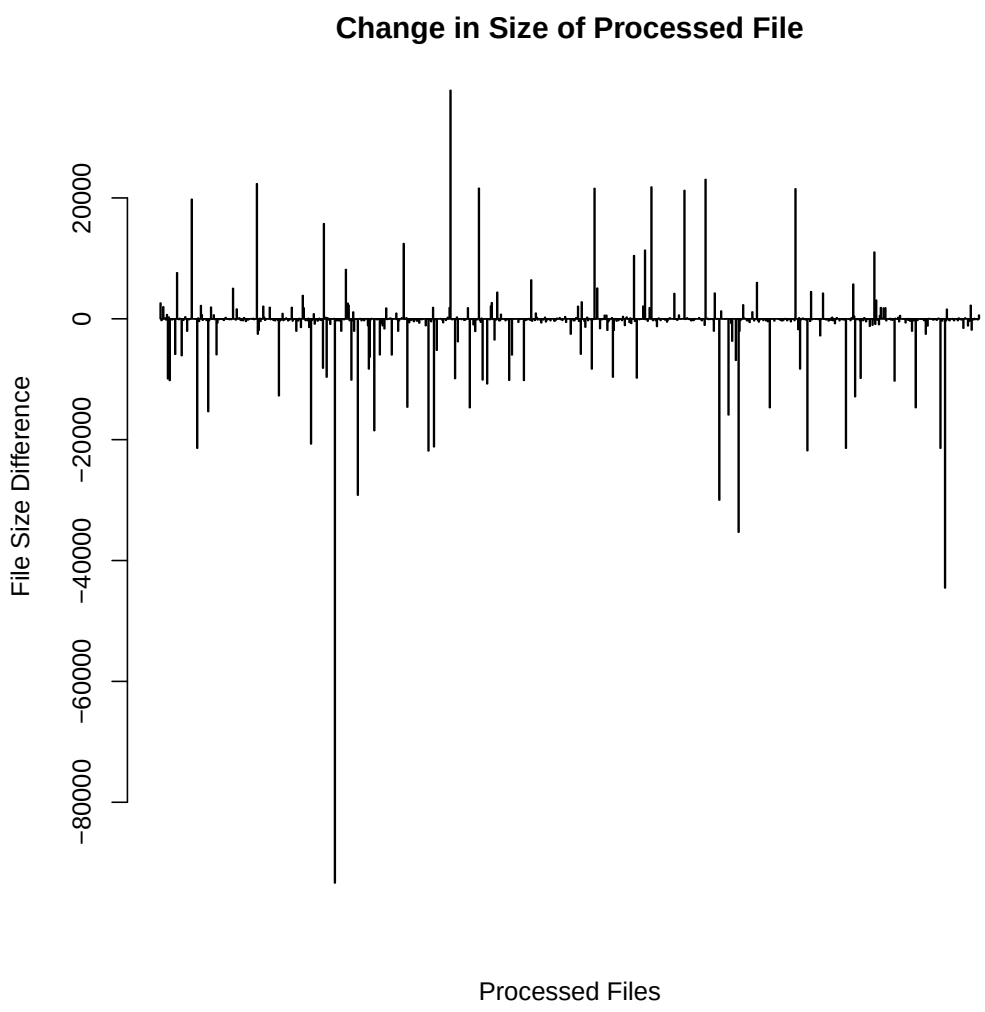

Figure 3: Barplot of Processed files in bytes, the new value is old byte size minus new byte size

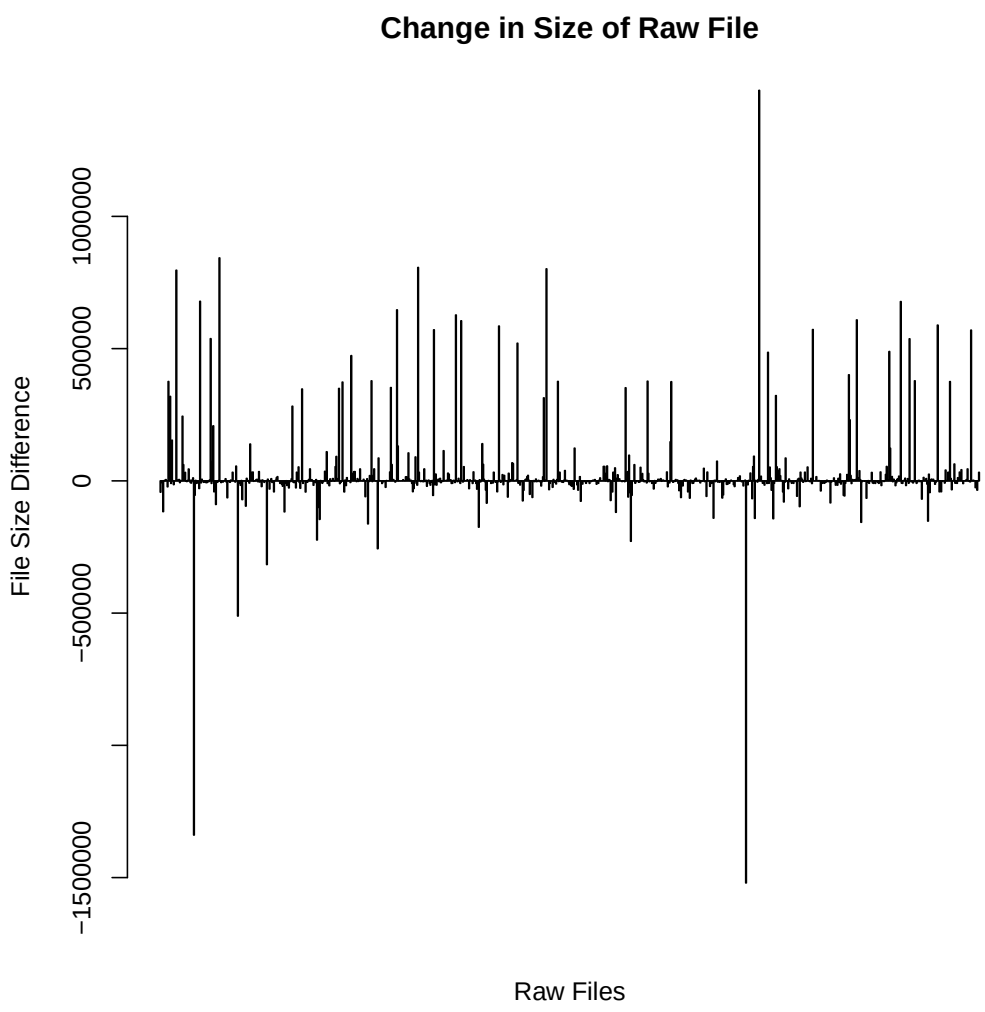

Figure 4: Barplot of Raw files in bytes, the new value is old byte size minus new byte size

References

[1] Atkins, Grant. "CS532 Assignment 10 Repository" Github. N.p., 23 March 2017. Web. 23 March 2017.https://github.com/grantat/cs532-s17/tree/master/assignments/A10.

[2] Segaran, Toby. "Programming Collective Intelligence". O' Reilly, 2007. Web. 6 April 2017. http://shop.oreilly.com/product/9780596529321.do.

[3] "How to setup LIBSVM for Python". n.p., n.d. Web. 1 May 2017. http://stackoverflow.com/questions/15755130/how-to-setup-libsvm-for-python