CS532 Web Science: Assignment 9

Finished on April 21, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

List of Tables

Listings

Problem 1

Question

Choose a blog or a newsfeed (or something similar with an Atom or RSS feed). Every student should do a unique feed, so please "claim" the feed on the class email list (first come, first served). It should be on a topic or topics of which you are qualified to provide classification training data. Find something with at least 100 entries (or items if RSS).

Create between four and eight different categories for the entries in the feed:

examples:

work, class, family, news, deals

liberal, conservative, moderate, libertarian

sports, local, financial, national, international, entertainment

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

Download and process the pages of the feed as per the week 12 class slides.

Be sure to upload the raw data (Atom or RSS) to your github account.

Answer

To obtain the blog entries required for this assignment, the `matrix.py` script was again used to download the blog entries from "Kevin's XL & Disc Golf Chronicles", a blog written by Kevin Morrow about his motorcycling and disc golfing exploits. The categories I came up with for each blog entry are as follows:

1. game: recreational game(s) of disc golf
2. tourney: tournament round(s)
3. motorcycles: anything related to riding/owning motorcycles
4. event: community events/cookouts
5. diy: disc dyes/graphic design

The script downloaded entries from the atom feed [1] of the blog until a total of 100 entries were retrieved. It parsed each entry's title and saved them as a list to the `blog_content` file which will later be used for training the fisher classifier in Question 2.

```
import feedparser
import futures
import math
import md5
import re
import sys
import json

blog_uri = 'http://kevinmorrow.blogspot.com/feeds/posts/default'
data_file = 'blog_content'

def get_next(d):
    for item in d.feed.links:
        if item['rel'] == u'next':
            return item['href']
    return None

def parse_entries(entries, uri):
    print('processing {}'.format(uri))
    next = uri
    while next is not None:
        feed = feedparser.parse(next)
        next = get_next(feed)
        print('next {}'.format(next))
        for entry in feed.entries:
            if entry.title in entries:
                continue
            entries.append(entry.title)
            if len(entries) >= 100:
                next = None
                break
    return entries

def load_data(filename):
    entries = []
    with open(filename) as infile:
        return [entry.strip() for entry in infile]

if __name__ == '__main__':
    old_entries = load_data(data_file)
    entries = parse_entries(old_entries, blog_uri)
    with open(data_file, 'w') as outfile:
        for entry in entries:
            outfile.write(entry + '\n')
```

Listing 1: matrix.py

Problem 2

Question

Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries.

Create a table with the title, predicted category, actual category,
and cprob() and fisherprob() for the actual category.

Answer

The `docclass.py` script was driven by the code shown in Listing 3. Before the script was run each of the 100 blog entries was classified manually to be used for training data later. This training is stored in the `training` file. The `docclass` main driver uses the first 50 entries as training for classifying the second 50 and then swaps each of these two sets. Tables 1 and 2 are the compiled training results with their actual classification and the cprob and fisherprob as calculated by the Fisher classifier. Refer to 4 in Appendix A for the full script. Each of the entry titles was used in full as the training feature since most of them are only a few words.

```
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('#', '\\#')
      row = ' & '.join([title, training[item], tup[0], str(tup[1])])
      outfile.write(row + ' \\\\\n')
    outfile.write(T_TAIL)
```

```
if __name__ == '__main__':
  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

Entry Title	Actual	Predicted	cprob	fisherprob
Disc Girl	diy	diy	0.139554246962	0.01247214
2013 DGCR Hawk Hollow Weekend	event	tourney	0.135398562817	0.577164396
Hey! I'm Back!	personal	motorcycles	0.742810969879	0.232819247
Last Ride of the Season	motorcycles	motorcycles	0.569769009772	0.01247214
Promoting the Sport	event	news	0.215734958342	0.217699063
Dyeing For Some New Discs....	diy	personal	0.127742032263	0.220540108
It's Been a Weird Sportster Year	motorcycles	diy	0.459029096789	0.055958898
Frank Lloyd Wright Day Trip	news	game	0.519935002549	0.034491001
Mach III Re-fit....	diy	diy	0.327231205524	0.986168775
Mike Sale: The Quest for 2,500	event	news	0.348489346703	0.05796196
2014 Chili Cook-Off	tourney	motorcycles	0.568479771654	0.304934209
Betty Queen Open	tourney	tourney	0.28731988169	0.553480606
2013 Hawk Hollow Open: Ams	tourney	tourney	0.0528373296205	0.011404427
Feelin' Lucky? Punk!....	news	tourney	0.655185013039	0.007793036
Red Oak Rumble	tourney	tourney	0.655185013039	0.001339982
Beginners Guide for Disc Dyes (Long Post)	diy	diy	0.195308043476	0.047201907
Snow Round at Loriella	game	game	0.476013302099	0.005693106
PDGA World's, the rest of it...	tourney	tourney	0.230722681279	0.614089489
Great Way to Start the Year	tourney	news	0.466829424722	0.056752429
2013 Hawk Hollow Open... Pros	tourney	tourney	0.0802708824856	0.03880296
Latest Disc Dye...	diy	diy	0.0574532219591	0.006505884
Winter Round	game	tourney	0.59657359028	0.636468946
Sporty Surprise...	motorcycles	tourney	0.59657359028	0.126113688
Deluxe Retractable Birdie Bead Scoring System	diy	tourney	0.759831170994	0.031327544
Latest Dye...	diy	diy	0.154721589649	0.000680295
Seneca Sun Seeker	tourney	tourney	0.655185013039	0.183933662
2013 Loriella Challenge	tourney	tourney	0.257589575924	0.082880791
First Day in Charlottesville	tourney	game	0.476013302099	0.082480313
Bayville Bash IX	tourney	tourney	0.59657359028	0.006962763
State of the Sporty	motorcycles	news	0.327231205524	0.001493909
Mornin' Round at Loriella	game	game	0.476013302099	0.062402682
Hawk Hollow Open	tourney	tourney	0.0672478176833	0.147750004
2nd Annual LoCo Open	tourney	tourney	0.254631706407	0.038507178
SOMD Classic	tourney	tourney	0.59657359028	0.683828022
Winchester IFO	tourney	tourney	0.59657359028	0.284807165
D Day Ride...	motorcycles	game	0.38493019271	0.126327062
Virginia Team Invitational	event	tourney	0.166468597895	0.016803297
Good & Bad day of DG	game	game	0.257589575924	0.993453532
Vincent & Jules....	diy	tourney	0.59657359028	0.007586851
All Ready for the World's	news	diy	0.0772653501085	0.852708891
The Season is Over...	news	tourney	0.215734958342	0.122180252
A Full Day Saturday	game	game	0.257589575924	0.037194075
Last Day of Vacation	motorcycles	game	0.476013302099	0.043672214
Turkey Day Doubles	game	game	0.384502787693	0.042393655
New Improved Putter	diy	diy	0.215734958342	0.320768231
Fall is coming...	motorcycles	tourney	0.59657359028	0.006163205
Lost Another One...	diy	diy	0.327231205524	0.3675231
First rounds at the Worlds	tourney	news	0.351391490133	0.022771071
Ace Race Fun....	event	personal	0.476013302099	0.449376649
Day Two at the World's	tourney	diy	0.122142469344	0.009538327

Table 1: Question 2: Predictions 1-50

Entry Title	Actual	Predicted	cprob	fisherprob
Maryland vs Virginia Ice Bowl Battle IV	event	event	0.0979229226451	0.080994041
Had Some Fun This Morning...	personal	personal	0.0605701708213	0.109631434
2014 River City Open	tourney	tourney	0.108800373877	0.978217096
Saturday in Staunton	tourney	tourney	0.174085576264	0.023708116
Doin' a dye, dye, dye & dye...	diy	diy	0.0732870294308	0.999943702
Bored at work = evental disc dye	diy	diy	0.02046431997	0.0000267
Spotsy SuperDubs	tourney	tourney	0.23578679514	0.001336632
2014 Hawk Hollow Open	tourney	tourney	0.0204836924925	0.096398553
It's Been Awhile...	diy	diy	0.400936622169	0.981799679
Pretty Good Disc Golf Day	game	game	0.031850199739	0.943089161
Bored = Dye Some Plastic...	diy	diy	0.0290173583845	0.215734958
Almost There...	motorcycles	motorcycles	0.886142331508	0.000884766
All Hail the Disc!	diy	diy	0.0497016027828	0.693526509
Knocked for a Loop Today...	news	news	0.0717591100714	0.057087801
Orlando and Some Disc Golf	personal	personal	0.0190818350443	0.007715421
Disc Golf & ChiliâĂę	event	event	0.0257577881041	0.009017303
It's Been A While...	diy	diy	0.303191637893	0.017392693
Santa's Little Helper....	diy	diy	0.137680696132	0.457547314
Hawk Hollow Open - Ams	tourney	tourney	0.00985086638406	0.0003178
#830RatedforLife....	tourney	tourney	0.25	0.169717444
Blast From My Past	diy	diy	0.215734958342	0.693945055
Skyline Drive or Last Place?	motorcycles	motorcycles	0.981220963209	0.162140906
New DG Hobby... Para Cord	diy	diy	0.102330524752	0.00497828
Battle in the Blue Ridge	tourney	tourney	0.14611299113	0.221964681
My Friend is a World Champ	tourney	tourney	0.089580585114	0.019803994
Another Day... Another Dye	diy	diy	0.0629356658608	0.021785892
2013 Battlefield Open	tourney	tourney	0.0291686553403	0.729853987
'Merica, Fuck Yeah!	game	game	0.215734958342	0.098144926
Too Hot to Play...	game	game	0.166468597895	0.023551902
I won't be posting for a while...	personal	personal	0.0717591100714	0.21318711
Facebook is Making my World a Little Smaller...	diy	diy	0.0725730164948	0.076723525
Mach III Re-Paint...	diy	diy	0.127217607055	0.177771567
Some days are Just Better...	game	game	0.0979229226451	0.999480271
Multi-Color Disc Dyes	diy	diy	0.0252331502301	0.000111479
2013 Virginia Team Invitational	tourney	event	0.0345067265265	0.004929646
Hell Hath Frozen Over...	tourney	tourney	0.155662943557	0.000716775
My Best Buddy Died today...	personal	personal	0.155662943557	0.728713968
Gettin' Ready....	event	event	0.174085576264	0.013419206
2 Rounds @ Loriella	game	game	0.0849057427037	0.012498733
New Backpack	news	news	0.101483354394	0.203721368
Building a Course...	news	news	0.23578679514	0.139554247
Getting it Back Together...	motorcycles	motorcycles	0.81216172237	0.932671336
DGCR Mid-Atlantic Meet	event	event	0.155662943557	0.174085576
Lost a Friend This Week	personal	personal	0.0950728157762	0.918752268
Shaving Cream Disc Dyes	diy	diy	0.0440390153459	0.163829249
New Putter Dye...	diy	diy	0.0261190230258	0.991422156
Do a Little Dye, Play a Little Golf	event	event	0.0202847913416	0.010090002
First Ride of 2014	motorcycles	motorcycles	0.801415273277	0.860680626
Promoting the Club	news	news	0.155177975467	0.235706362
Multi-Color 2nd Attempt	diy	diy	0.0950728157762	0.406330125
Day Two at the World's	tourney	news	0.171383513075	0.633731408

Table 2: Question 2: Predictions 51-100

Problem 3

Question

Assess the performance of your classifier in each of your categories by computing precision, recall, and F-measure.

Answer

To calculate the *precision*, *recall* and *F-Measure* the `assess.py` script was used. This script parsed the pipe separated table stored in the file `predict_raw`, which contains all the predictions and cprob values for each item. The table is separated on each category.

```
def load_data(filename):
    data = {}
    with open(filename) as infile:
        for line in infile:
            entry, actual, predicted, cprob = line.split('|')
            data[entry] = {'actual': actual, 'predicted': predicted, 'cprob': float(cprob.
                strip())}
    return data

def assess(data, categories):
    results = {}
    for category in categories:
        tp, fp, fn = float(0), float(0), float(0)
        for entry, items in data.iteritems():
            if data[entry]['actual'] != category:
                continue
            if not data[entry]['predicted']:
                fn += 1
            elif data[entry]['actual'] == data[entry]['predicted']:
                tp += 1
            elif data[entry]['actual'] != data[entry]['predicted']:
                fp += 1
        prec = tp / (tp + fp)
        recall = tp / (tp + fn)
        f1 = 2 * (prec * recall) / (prec + recall)
        results[category] = {'p': str(prec), 'r': str(recall), 'f1': str(f1)}
    return results

categories = ['game', 'tourney', 'motorcycles', 'event', 'diy']

T_HEAD = """\\begin{table}[h!]
\centering
\\begin{tabular}{| l | l | l | l |}
\hline
Category & Precision & Recall & F-Measure \\\\
\hline
"""

T_TAIL = """\hline
\end{tabular}
\caption{Question 3: Assessments }
\label{tab:assess}
\end{table}
"""

data = load_data('predict_raw')
res = assess(data, categories)
with open('assess', 'w') as outfile:
    outfile.write(T_HEAD)
```

```
    for cat, table in res.iteritems():
        outfile.write(' & '.join([cat, table['p'], table['r'], table['f1']]) + ' \\\\\n')
52  outfile.write(T_TAIL)
```

Listing 3: docclass main

Category	Precision	Recall	F-Measure
event	0.5	1.0	0.666666666667
game	0.909090909091	1.0	0.952380952381
tourney	0.785714285714	1.0	0.88
diy	0.884615384615	1.0	0.938775510204
motorcycles	0.454545454545	1.0	0.625

Table 3: Question 3: Assessments

Appendix A

```
from sqlite3 import dbapi2 as sqlite
import re
import math
import matrix

def getwords(doc):
  splitter=re.compile('\\W*')
  # Split the words by non-alpha characters
  words=[s.lower() for s in splitter.split(doc)
         if len(s)>2 and len(s)<20]

  # Return the unique set of words only
  return dict([(w,1) for w in words])

class classifier:
  def __init__(self,getfeatures,filename=None):
    # Counts of feature/category combinations
    self.fc={}
    # Counts of documents in each category
    self.cc={}
    self.getfeatures=getfeatures

  def setdb(self,dbfile):
    self.con=sqlite.connect(dbfile)
    self.con.execute('create table if not exists fc(feature,category,count)')
    self.con.execute('create table if not exists cc(category,count)')


  def incf(self,f,cat):
    count=self.fcount(f,cat)
    if count==0:
      self.con.execute("insert into fc values ('%s','%s',1)"
                       % (f,cat))
    else:
      self.con.execute(
        "update fc set count=%d where feature='%s' and category='%s'"
        % (count+1,f,cat))

  def fcount(self,f,cat):
    res=self.con.execute(
      'select count from fc where feature="%s" and category="%s"'
      %(f,cat)).fetchone()
    if res==None: return 0
    else: return float(res[0])

  def incc(self,cat):
    count=self.catcount(cat)
    if count==0:
      self.con.execute("insert into cc values ('%s',1)" % (cat))
    else:
      self.con.execute("update cc set count=%d where category='%s'"
                       % (count+1,cat))

  def catcount(self,cat):
    res=self.con.execute('select count from cc where category="%s"'
                         %(cat)).fetchone()
    if res==None: return 0
    else: return float(res[0])

  def categories(self):
    cur=self.con.execute('select category from cc');
    return [d[0] for d in cur]

  def totalcount(self):
    res=self.con.execute('select sum(count) from cc').fetchone();
```

```
    if res==None: return 0
    return res[0]

  def train(self,item,cat):
    features=self.getfeatures(item)
    # Increment the count for every feature with this category
    for f in features:
      self.incf(f,cat)

    # Increment the count for this category
    self.incc(cat)
    self.con.commit()

  def fprob(self,f,cat):
    if self.catcount(cat)==0: return 0

    # The total number of times this feature appeared in this
    # category divided by the total number of items in this category
    return self.fcount(f,cat)/self.catcount(cat)

  def weightedprob(self,f,cat,prf,weight=1.0,ap=0.5):
    # Calculate current probability
    basicprob=prf(f,cat)

    # Count the number of times this feature has appeared in
    # all categories
    totals=sum([self.fcount(f,c) for c in self.categories()])

    # Calculate the weighted average
    bp=((weight*ap)+(totals*basicprob))/(weight+totals)
    return bp



class naivebayes(classifier):

  def __init__(self,getfeatures):
    classifier.__init__(self,getfeatures)
    self.thresholds={}

  def docprob(self,item,cat):
    features=self.getfeatures(item)

    # Multiply the probabilities of all the features together
    p=1
    for f in features: p*=self.weightedprob(f,cat,self.fprob)
    return p

  def prob(self,item,cat):
    catprob=self.catcount(cat)/self.totalcount()
    docprob=self.docprob(item,cat)
    return docprob*catprob

  def setthreshold(self,cat,t):
    self.thresholds[cat]=t

  def getthreshold(self,cat):
    if cat not in self.thresholds: return 1.0
    return self.thresholds[cat]

  def classify(self,item,default=None):
    probs={}
    # Find the category with the highest probability
    max=0.0
    for cat in self.categories():
      probs[cat]=self.prob(item,cat)
```

```
      if probs[cat]>max:
        max=probs[cat]
        best=cat

    # Make sure the probability exceeds threshold*next best
    for cat in probs:
      if cat==best: continue
      if probs[cat]*self.getthreshold(best)>probs[best]: return default
    return best

class fisherclassifier(classifier):
  def cprob(self,f,cat):
    # The frequency of this feature in this category
    clf=self.fprob(f,cat)
    if clf==0: return 0

    # The frequency of this feature in all the categories
    freqsum=sum([self.fprob(f,c) for c in self.categories()])

    # The probability is the frequency in this category divided by
    # the overall frequency
    p=clf/(freqsum)

    return p
  def fisherprob(self,item,cat):
    # Multiply all the probabilities together
    p=1
    features=self.getfeatures(item)
    for f in features:
      p*=(self.weightedprob(f,cat,self.cprob))

    # Take the natural log and multiply by -2
    fscore=-2*math.log(p)

    # Use the inverse chi2 function to get a probability
    return self.invchi2(fscore,len(features)*2)
  def invchi2(self,chi, df):
    m = chi / 2.0
    sum = term = math.exp(-m)
    for i in range(1, df//2):
        term *= m / i
        sum += term
    return min(sum, 1.0)
  def __init__(self,getfeatures):
    classifier.__init__(self,getfeatures)
    self.minimums={}

  def setminimum(self,cat,min):
    self.minimums[cat]=min

  def getminimum(self,cat):
    if cat not in self.minimums: return 0
    return self.minimums[cat]
  def classify(self,item,default=None):
    # Loop through looking for the best result
    best=default
    max=0.0
    for c in self.categories():
      p=self.fisherprob(item,c)
      # Make sure it exceeds its minimum
      if p>self.getminimum(c) and p>max:
        best=c
        max=p
    return best, p

def sampletrain(cl):
  cl.train('Nobody owns the water.','good')
  cl.train('the quick rabbit jumps fences','good')
```

```
  cl.train('buy pharmaceuticals now','bad')
  cl.train('make quick money at the online casino','bad')
  cl.train('the quick brown fox jumps','good')

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('#', '\\#')
      row = ' & '.join([title, training[item], tup[0], str(tup[1])])
      outfile.write(row + ' \\\\\\n')
    outfile.write(T_TAIL)

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

Listing 4: docclass.py

References

[1] Internet Engineering Task Force (IETF). Rfc-4287 the atom syndication format. https://tools.ietf.org/html/rfc4287, 2005.