CS595 Intro to Web Science, Assignment #10

Valentina Neblitt-Jones

December 12, 2013

Listings

List of Figures

Question 1

Choose a blog or newsfeed (or something with an Atom or RSS feed). It should be on a topic or topics or which you are qualified to provide classification training data. Find something with at least 100 entries.

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.

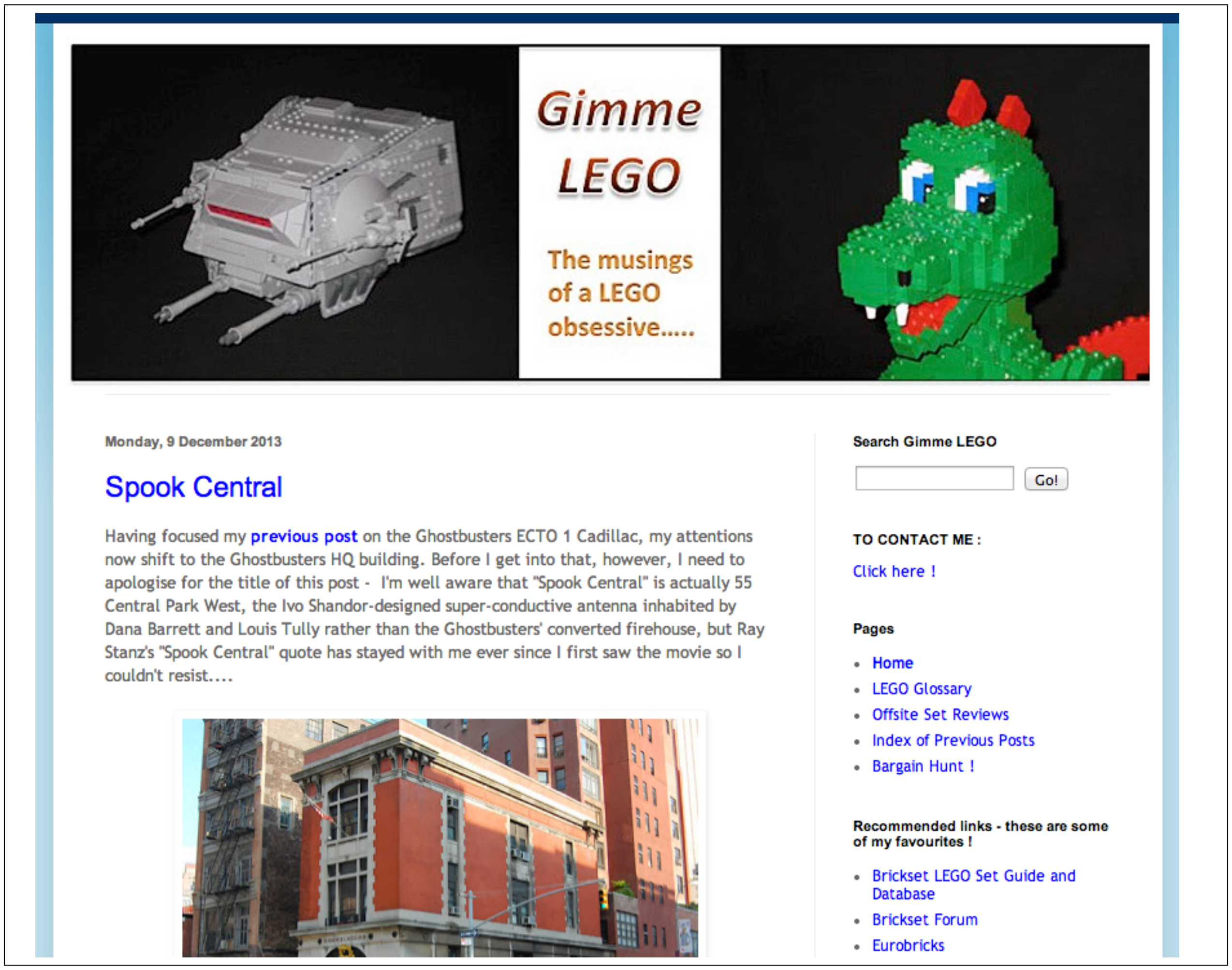
Answer to Question 1

Since I am an adult fan of LEGO (AFOL), I looked for a blog about LEGO and found *Gimme LEGO: The musing of a LEGO obsessive* at `http://gimmelego.blogspot.com/` (Figure 1). I am a founding member of a local LEGO club (HARDLUG) that has existed for 11 years and feel qualified to classify a blog about the hobby/obsession.

My categories were the following:

- original - originals meaning not a set; the correct term is MOC meaning "my own creation", but I felt that was unreasonable use of jargon
- sets - official sets release by The LEGO Group
- events - launch events, tours or conferences attended by the blogger
- awards - awards either in the blogger's opinion or his readers' opinions
- minifigures - focused on LEGO mini figures; The LEGO Group started releasing collectible minifigures in series in 2010
- market - posts about prices whether it was about deals/bargains or complaints
- summary - infrequently the blogger would write about blogging about LEGO and how it was going so far (e.g., successes, failures)

I believe these categories are suitable for a general audience. If my audience was other AFOLs, I would be more detailed and specific with my subject headings.



Monday, 9 December 2013

Spook Central

Having focused my previous post on the Ghostbusters ECTO 1 Cadillac, my attentions now shift to the Ghostbusters HQ building. Before I get into that, however, I need to apologise for the title of this post - I'm well aware that "Spook Central" is actually 55 Central Park West, the Ivo Shandor-designed super-conductive antenna inhabited by Dana Barrett and Louis Tully rather than the Ghostbusters' converted firehouse, but Ray Stanz's "Spook Central" quote has stayed with me ever since I first saw the movie so I couldn't resist....

Search Gimme LEGO

Go!

TO CONTACT ME :

Click here !

Pages

- Home
- LEGO Glossary
- Offsite Set Reviews
- Index of Previous Posts
- Bargain Hunt !

Recommended links - these are some of my favourites !

- Brickset LEGO Set Guide and Database
- Brickset Forum
- Eurobricks

Figure 1: Gimme LEGO Blog

To obtain the XML file with the 100 entries, I clicked on the link to subscribe to posts via Atom and changed the max results to 100 (`http://gimmelego.blogspot.com/feeds/posts/default?max-results=100`) and saved the file as xml. The file is titled gimmelego.xml on GitHub at `https://github.com/vneblitt/cs595-f13/blob/master/assignment10/files/gimmelego.xml`. The blog titles are in Appendix C.

Question 2

Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries. Report the cprob() values for the 50 titles as well. From the title or entry itself, specify the 1-, 2-, or 3-gram that you used for the string to classify. Do not repeat strings; you will have 50 unique strings. For example, in these titles the string used is marked with *s:

- *Rachel Goswell* - “Waves are Universal” (LP Review)
- The *Naked and Famous* - “Passive Me, Aggressive You” (LP Review)
- *Negativland* - “Live at Lewis’s, Norfolk VA, November 21, 1992” (concert)
- Negativland - “*U2*” (LP Review)

Note how “Negativland” is not repeated as a classification string.

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

Answer to Question 2

This was disappointing. I managed to manually classify the first 50 entries and generate the database file using createTrainingData.py and docclass.py[1] (Listings 1 and 2). The database is on GitHub and titled gimmelego.db. My next set of error messages indicated that my program could not access the generated database to classify the second 50 entries. I do hope to eventually figure it out.

Question 3

Assess the performance of your classifier in each of your categories by computing precision and recall. Note that the definitions are slightly different in the context of classification; see: `http://en.wikipedia.org/wiki/Precision_and_recall#Definition_.28classification_context.29`

Answer to Question 3

Not reached due to problems with Q2.

Question 4 - Extra Credit (5 points)

Redo the questions above, but with the extensions on slide 26 and pp. 136-138.

Answer to Question 4

Not attempted.

A createTrainingData.py

Listing 1: createTrainingData.py

```
import os
import feedparser
import docclass

os.unlink('gimmelego.db')

# The first 50 entries manually classified
cl=docclass.classifier(docclass.getwords)

cl.setdb('gimmelego.db')

d = feedparser.parse('http://gimmelego.blogspot.com/feeds/posts/default?max-results=100')

cl.train(d.entries[0].content[0].value, 'original')
cl.train(d.entries[1].content[0].value, 'original')
cl.train(d.entries[2].content[0].value, 'books')
cl.train(d.entries[3].content[0].value, 'sets')
cl.train(d.entries[4].content[0].value, 'events')
cl.train(d.entries[5].content[0].value, 'events')
cl.train(d.entries[6].content[0].value, 'original')
cl.train(d.entries[7].content[0].value, 'sets')
cl.train(d.entries[8].content[0].value, 'sets')
cl.train(d.entries[9].content[0].value, 'books')
cl.train(d.entries[10].content[0].value, 'events')
cl.train(d.entries[11].content[0].value, 'sets')
cl.train(d.entries[12].content[0].value, 'original')
cl.train(d.entries[13].content[0].value, 'original')
cl.train(d.entries[14].content[0].value, 'events')
cl.train(d.entries[15].content[0].value, 'events')
cl.train(d.entries[16].content[0].value, 'sets')
cl.train(d.entries[17].content[0].value, 'sets')
cl.train(d.entries[18].content[0].value, 'minifigures')
cl.train(d.entries[19].content[0].value, 'original')
cl.train(d.entries[20].content[0].value, 'sets')
cl.train(d.entries[21].content[0].value, 'sets')
cl.train(d.entries[22].content[0].value, 'sets')
cl.train(d.entries[23].content[0].value, 'sets')
cl.train(d.entries[24].content[0].value, 'sets')
cl.train(d.entries[25].content[0].value, 'original')
cl.train(d.entries[26].content[0].value, 'sets')
cl.train(d.entries[27].content[0].value, 'original')
cl.train(d.entries[28].content[0].value, 'sets')
cl.train(d.entries[29].content[0].value, 'awards')
cl.train(d.entries[30].content[0].value, 'sets')
cl.train(d.entries[31].content[0].value, 'awards')
cl.train(d.entries[32].content[0].value, 'awards')
cl.train(d.entries[33].content[0].value, 'sets')
cl.train(d.entries[34].content[0].value, 'sets')
cl.train(d.entries[35].content[0].value, 'original')
cl.train(d.entries[36].content[0].value, 'sets')
cl.train(d.entries[37].content[0].value, 'sets')
cl.train(d.entries[38].content[0].value, 'market')
cl.train(d.entries[39].content[0].value, 'original')
cl.train(d.entries[40].content[0].value, 'events')
cl.train(d.entries[41].content[0].value, 'sets')
cl.train(d.entries[42].content[0].value, 'original')
cl.train(d.entries[43].content[0].value, 'sets')
cl.train(d.entries[44].content[0].value, 'minifigures')
cl.train(d.entries[45].content[0].value, 'original')
cl.train(d.entries[46].content[0].value, 'summary')
cl.train(d.entries[47].content[0].value, 'original')
cl.train(d.entries[48].content[0].value, 'sets')
cl.train(d.entries[49].content[0].value, 'market')
```

```
# The second 50 entries classified using Fisher

cl = docclass.fisherclassifier(docclass.getwords)

for i in range(50, 100):
        category = cl.classify(d.entries[i].content[0].value)
        print d.entries[i].title + ' : ' + category

for i in range(50, 100):
        for cat in categories:
                print cl.cprob(d.entries[i].title, cat)
```

B docclass.py

Listing 2: docclass.py

```
from pysqlite2 import dbapi2 as sqlite
import re
import math

def getwords(doc):
  splitter=re.compile('\'\\W*')
  print doc
  # 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
```

C 100 Blog Titles from GimmeLEGO

1. Spook Central
2. Who Ya Gonna Call ?
3. American Beauty
4. A Wretched Hive of Scum and Villainy....
5. STEAMrollered
6. Full STEAM Ahead
7. Gerry Anderson Returns....?
8. Shanghai Surprise
9. Galaxy Squad - Like a Prayer
10. Spoiled for Choice
11. Ready for Launch
12. Designer Robot
13. UCS AT-AT : They think it's all over...
14. UCS AT-AT : The Home Straight
15. LEGO Inside Tour 2013 part 2
16. LEGO Inside Tour 2013
17. Rocky Horror ?
18. Yellows !
19. Gold Rush
20. UCS AT-AT : Raising the Roof
21. Jabba Dabba Doo !
22. Hi Ho Silver Lining
23. Cowabunga !
24. Night at the Museum
25. A-wing and a Prayer

26. UCS AT-AT : Beside Myself...
27. The Real Classic Space ?
28. UCS AT-AT : Heady Days
29. Technic Temptation....
30. “And the Gimme LEGO Readers’ Choice Award for Best Set of 2012 goes to....”
31. Moonlighting
32. 2012 Readers Choice Award - the Nominations
33. The Gimme LEGO Awards 2012
34. “They’re for sale if you want them”
35. One Hundred Percent
36. UCS AT-AT : Once More Unto the Breach....
37. Monsters, Inc.
38. Blast from the Past : Set 657 Executive Jet
39. Bargain Hunt Lives !
40. UCS AT-AT : Body Beautiful
41. Blown Away
42. Van-tastic
43. UCS AT-AT : Tricky
44. Bling
45. Still Going Strong
46. UCS AT-AT : the build begins....
47. Happy Birthday to Me !
48. Building the Perfect Beast - Anatomy of an AT-AT
49. Going for Gold
50. Scalpers Rejoice !
51. Building the Perfect Beast - the UCS AT-AT
52. Hell’s Bells
53. Ancient Treasures
54. I Want My Mummy !
55. New York, New York
56. What’s that coming over the hill....?
57. Block-tastic
58. Dangerous

59. Spaced Out
60. Fangs for the Memories...
61. Home Truths
62. No Consolation
63. United in Manchester
64. Blast from the Past : Set 695 Racing Car
65. Dreams Can Come True...
66. Apologies...
67. Headache
68. Whoops...
69. 3-in-1
70. Sneak Peek part 2 : Star Wars Miniland
71. Sneak Peek
72. “The damage doesn’t look as bad from out here”
73. Baubles
74. So Bad it’s Almost Good
75. Once upon a time......
76. The Joy of Six
77. Dino
78. A Different World
79. One step back and two steps forward....
80. Taking the Plunge
81. The 2011 Gimme LEGO Readers Choice Award.....
82. Christmas Car-nage
83. 25th of December it is.....
84. The Gimme LEGO Awards 2011
85. The best things....
86. Better late than never
87. Shuttle Love
88. Bucket List Reloaded
89. Bucket List
90. Charidee
91. Girl’s Stuff

92. Blasphemy
93. Here comes The Sun
94. Here there be Dragons...
95. One Year On....
96. Revealed : Kingdoms Set 10223 Royal Joust
97. Midi-chlorians
98. Neo- Neo- Classic Space ?
99. Favourite Sets #6 : Tractor
100. Ultimate ?

References

[1] Segaran, T. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly, 2007.