```
'''
Created on Mar 13, 2013

@author: Masum
'''

import re
from math import log
from collections import defaultdict
import os
import sys

def replace_tag2(line_text, tag, pseudoword, line_no, context_size=2):  # <tag "532675">
    header = str(line_no) + ":" + pseudoword + ":" + str(tag) + "\n";
    line_text = re.sub('<tag "\d+">\s*\w+</>', pseudoword, line_text, 1)
    line_text = re.sub('[.|"|,|;|:|!|(|)|?|`|\']', '', line_text);
    parsed = [tok.lower() for tok in line_text.split(' ') if len(tok)>2];
    contexts = []

    index = 0;
    try:
        index = parsed.index(pseudoword);
    except:
        #print line_text+'\n'
        return None;

    #add left tokens
    if (index>=context_size):
        contexts.extend(parsed[index-context_size : index])
    #add right tokens
    if len(parsed)>(index+context_size):
        contexts.extend( parsed[index+1:index+1+context_size])
    #print contexts
    line_text = header + (' ').join(contexts) + "\n"
    return line_text;

def process_file(file_in, max_lines, file_out, pseudoword, sense, start_line, context_size):
    mode = 'w+' if (start_line==0) else 'a';
    f_train = open(file_out+".train.txt", mode);
    f_test = open(file_out+".test.txt", mode);
    curr_line = 0;
    tokenized_line="";

    with open(file_in, 'r') as f:
        line_text = "";
        within_a_line = False
        for line in f:
            if re.match("\n", line):
                # if start_line>3:break;
                within_a_line = False;
                tokenized_line = replace_tag2(line_text, sense, pseudoword, start_line,
  context_size)
                if tokenized_line:
                    if curr_line<max_lines*.8:
                        f_train.write(tokenized_line);
                    else:
                        f_test.write(tokenized_line);
```

```
57                      start_line += 1;
58                  curr_line+=1;
59                  line_text = ""
60                  continue;
61
62              if re.match('\d+', line):
63                  within_a_line = True;
64                  continue;
65
66              if within_a_line:
67                  line_text += line.strip() + " ";
68      f_train.close();
69      f_test.close();
70      return start_line;
71  # end
72
73  def build_corpus(file_in1, line_no1,file_in2, line_no2,file_out, pseudoword, contex_size):
74      # 1st file
75      line_no = process_file(file_in1, line_no1,file_out, pseudoword , 0, 0, contex_size);
76      # 2nd file
77      process_file(file_in2, line_no2,file_out, pseudoword , 1, line_no, contex_size);
78  # end;
79
80
81  def run_traning(file_in, pseudoword):
82      hashes = [defaultdict(float), defaultdict(float)]
83      sense_types= [0,1];
84      count_senses=[0.0, 0.0];
85      current_sense = -1
86
87      with open(file_in, 'r') as f:
88          line_no=0;
89          for line in f:
90              #if line_no>3:break;
91              m = re.match("\d+:"+pseudoword+":(\d)", line) #0:accidentwooden:0
92              if m:
93                  current_sense = int(m.group(1))
94                  count_senses[current_sense]+=1;
95                  line_no += 1;
96              else:
97                  for context in  line.strip().split(" "):
98                      hashes[current_sense][context]+=1;
99          #priors
100         priors = [1.0*count_senses[0]/(count_senses[0]+count_senses[1]), 1.0*count_senses[1]/
    (count_senses[0]+count_senses[1])];
101
102         #conditionals
103         for sense in sense_types:
104             for context in hashes[sense]:
105                 hashes[sense][context] = 1.0*hashes[sense][context]/count_senses[sense];
106
107     #print hashes;
108     return sense_types, priors,hashes ;
109
110
111 def run_disambiguation(file_train, file_test, pseudoword, context_size):
112     sense_types, priors, conditionals =  run_traning(file_train, pseudoword);
```

```
113     actual_sense = -1;
114     predicted_sense = -1;
115     scores  = [0.0, 0.0];
116 
117     #print "priors: ", priors;
118     #print "sense types: ",sense_types;
119     #print "training instances: ",count_senses
120     #print conditionals[1];
121 
122     factor = 10000;
123     count_corrects=0;
124     count_wrong= 0;
125 
126     with open(file_test, 'r') as f:
127         line_no=0;
128         for line in f:
129             m = re.match("\d+:"+pseudoword+":(\d)\n", line) #0:accidentwooden:0
130             if m:
131                 actual_sense = int(m.group(1))
132                 line_no += 1;
133             else:
134                 contexts = line.strip().split(" ");
135                 for sense in sense_types:
136                     scores[sense] = log(priors[sense]);
137 
138                     for context in contexts:
139                         if context in conditionals[sense]:
140                             prob = conditionals[sense][context]
141                             scores[sense] += log((prob*factor+1.0)/(factor+context_size));
142                         else:
143                             scores[sense] += log(1.0/factor);
144                         #end inner for
145                     #end sense
146                 #end outer for
147                 if scores[0]>scores[1]:
148                     predicted_sense = 0;
149                 else:
150                     predicted_sense = 1;
151 
152                 #calculate accuracy
153                 if predicted_sense==actual_sense:
154                     count_corrects+=1;
155                 else:
156                     count_wrong+=1;
157             #print actual_sense, predicted_sense;
158         #print 'accuracy: ', count_corrects*100.0/(count_corrects+count_wrong), '\n';
159         print context_size, ' ',count_corrects*100.0/(count_corrects+count_wrong);
160 
161 
162 def run_pair(w1, line_no1, w2, line_no2, context_count):
163     build_corpus('wsd/'+w1+'.cor', line_no1,'wsd/'+w2+'.cor', line_no2,'wsd/'+w1+w2+'.bag',
    w1[2:]+w2[2:], context_count)
164     run_disambiguation('wsd/'+w1+w2+'.bag.train.txt',
    'wsd/'+w1+w2+'.bag.test.txt',w1[2:]+w2[2:], context_count);
165 
166 
167 def hw3():
```

```
168     for i in range(1,20):
169         #print 'for run', i,':';
170         run_pair('4.amaze', 319,'4.behaviour', 1003, i);
171         #run_pair('3.sack', 296,'3.sanction', 101, i);
172         #run_pair('2.knee', 477,'2.onion', 29, i);
173         #run_pair('1.accident', 1303,'1.wooden', 370, i);
174 #end
175 
176 
177 def get_file_chars(fname):
178     alphabet =
    ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x
    ','y','z','0','1','2','3','4','5','6','7','8','9'];
179     alpha_dict = defaultdict(float);
180 
181     total=0
182     with open(fname, 'r') as f:
183         for line in f:
184             for c in line.lower():
185                 total+=1;
186                 if c in alphabet:
187                     alpha_dict[c] += 1.0;
188     return alpha_dict;
189 #end
190 
191 def entropy(alpha_dict):
192     info = 0.0
193     total = 0. + sum(alpha_dict.values())
194     for c in alpha_dict:
195         #print c, ' ', alpha_dict[c]
196         p = alpha_dict[c]/total;
197         info += -p*log(p, 2);
198     print 'entropy is: ',info;
199 
200 def kldiv(_s, _t):
201     if (len(_s) == 0):return 1e33
202     if (len(_t) == 0):return 1e33
203 
204     ssum = 0. + sum(_s.values())
205     tsum = 0. + sum(_t.values())
206 
207     vocabdiff = set(_s.keys()).difference(set(_t.keys()))
208     lenvocabdiff = len(vocabdiff)
209 
210     """ epsilon """
211     epsilon = min(min(_s.values())/ssum, min(_t.values())/tsum) * 0.001
212 
213     """ gamma """
214     gamma = 1 - lenvocabdiff * epsilon
215 
216     """ Check if distribution probabilities sum to 1"""
217     sc = sum([v/ssum for v in _s.itervalues()])
218     st = sum([v/tsum for v in _t.itervalues()])
219 
220     if sc < 9e-6:
221         print "Sum P: %e, Sum Q: %e" % (sc, st)
222         print "*** ERROR: sc does not sum up to 1. Bailing out .."
```

```
223         sys.exit(2)
224     if st < 9e-6:
225         print "Sum P: %e, Sum Q: %e" % (sc, st)
226         print "*** ERROR: st does not sum up to 1. Bailing out .."
227         sys.exit(2)
228 
229     div = 0.
230     for t, v in _s.iteritems():
231         pts = v / ssum
232         ptt = epsilon
233         if t in _t:
234             ptt = gamma * (_t[t] / tsum)
235         ckl = (pts - ptt) * log(pts / ptt)
236         div +=  ckl
237     return div
238 #end of kldiv
239 
240 
241 
242 def get_corpus_count(corpus_path):
243     total_freq = defaultdict(float);
244 
245     for filename in os.listdir (corpus_path):
246         #print filename
247         file_count = get_file_chars(os.path.abspath(corpus_path)+os.path.sep+filename);
248         for k in file_count:
249             total_freq[k] += file_count[k];
250         #print '\n'
251     #get_file_chars("kl/English1/test.txt");
252     #for k in total_freq:
253         #print k, ' ', total_freq[k];
254     return total_freq
255 
256 
257 corpus_path1 = "kl/English1/"
258 corpus_path2 = "kl/English2/"
259 corpus_path3 = "kl/French1/"
260 
261 print "KL-divergence <1,2>:",
    kldiv(get_corpus_count(corpus_path1),get_corpus_count(corpus_path2));
262 print "KL-divergence <2,1>:",
    kldiv(get_corpus_count(corpus_path2),get_corpus_count(corpus_path1));
263 
264 print "KL-divergence <1,3>:",
    kldiv(get_corpus_count(corpus_path1),get_corpus_count(corpus_path3));
265 print "KL-divergence <3,1>:",
    kldiv(get_corpus_count(corpus_path3),get_corpus_count(corpus_path1));
266 
267 print "KL-divergence <2,3>:",
    kldiv(get_corpus_count(corpus_path2),get_corpus_count(corpus_path3));
268 print "KL-divergence <3,2>:",
    kldiv(get_corpus_count(corpus_path3),get_corpus_count(corpus_path2));
269 
270 
271 
```