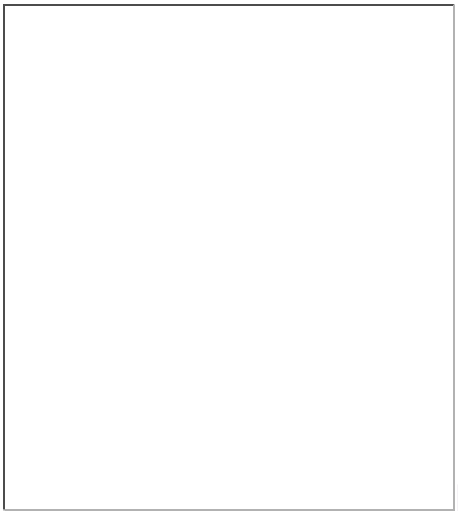
Counting Syllables Accurately in Python on Google App Engine

Posted on 29th June 2011 13th April 2015 by Danny Goodall



I wanted to be able to count syllables accurately in Python and looked around for existing code that I could re-use. I found one or two routines written in PHP that looked promising so I ported them to Python but was pretty disappointed with the accuracy.

I also found a Python routine that is part of the contributed code for NLTK that was not bad but again struggled with some words. You see, I had naively thought this would be a simple exercise. I hadn't realised that Syllable Counting in the English language is pretty difficult stuff with so many exceptions that it makes the most elegant algorithm convoluted and clumsy.

I then stumbled across this snippet of code by Jordan Boyd-Graper, via the excellent Running with Data site, and it seemed so elegant that I thought it must be too simplistic. But far from it, it is very accurate for the words it knows .

The code is shown here.

```
import curses
from curses.ascii
```

```
1 import curses
2 from curses.ascii import isdigit
3 import nltk
4 from nltk.corpus import cmudict
5
6 d = cmudict.dict()
```

```
7 def nsyl(word):
8 return [len(list(y for y in x if isdigit(y[-1]))) for x in d[word.lower()]]
```

It works by looking up the pronunciation of the word in the Carnegie Mellon University's pronunciation dictionary that is part of the Python-based Natural Language Toolkit (NLTK). This returns one or more pronunciations for the word. Then the clever bit is that the routine counts the stressed vowels in the word. The raw entry from the cmudict file for the word SYLLABLE is shown below.

```
SYLLABLE 1 S IH1 L
AH0 B AH0 L
```

```
1 SYLLABLE 1 S IH1 L AH0 B AH0 L
```

The stressed vowels are denoted by the string of letters ending in a number. They appear to represent the different individual pronunciations of the vowel sound. Anyway, for the words that the dictionary knows about (120,000+ I believe), this represents a very accurate method for obtaining the syllable count.

However, there is a problem. As my target environment is Google App Engine, that little line at the top of the code that says…

```
import nltk
```

```
1 import nltk
```

…ruins your entire afternoon.

You see NLTK and Google App Engine don't work well together due to NLTK's recursive imports. I spent some time trying to unwind the recursive imports on cmudict so that Google App Engine would work but to no avail.

So then I thought laterally and decided to build my own structure from the cmudict file (the raw text 3.6MB file that NLTK loads and wraps an object around). My plan was as follows:

1. Parse the raw cmudict file
2. For every word in the file call the above syllable count routine
3. Store the resultant syllable count in a word -> syllable lookup structure (a Python Dictionary)
4. Pickle the resultant dictionary
5. Un-pickle it where it is needed

And this seems to have worked quite well.

The code below builds the pickle file.

```
#!/usr/bin/env
python
```

```
1  #!/usr/bin/env python
2
3  from curses.ascii import isdigit
4  from nltk.corpus import cmudict
5  try:
6      import cPickle as pickle
7  except:
8      import pickle
9
10 #-----
11 # Create a shared dictionary key's on the word with the value as a list of
12 # possible syllable counts
13
14 GzzCMUDict = cmudict.dict()
15
16 GdcSyllableCount = {}
17
18 def CreatePickle(AlgQuiet=False):
19
20     def SyllableCount(AszWord):
21         """return the max syllable count in the case of multiple pronunciations"""
22
23         #http://groups.google.com/group/nltk-users/msg/81e70cb6704dc01e?pli=1
24
25         return [len([y for y in x if isdigit(y[-1])]) for x in GzzCMUDict[AszWord.lower()]]
26
27     try:
28         LhaInputFile = open('cmudict',r+')
29     except:
30         print "Could not open the cmudict file"
31         raise IOError
32
33     try:
34         for LszLine in LhaInputFile:
35
36             LszWord = LszLine.split(' ')[0].lower()
37
38             LliSyllableList = SyllableCount(LszWord)
39
40             if LszWord not in GdcSyllableCount:
41                 GdcSyllableCount[LszWord] = sorted(LliSyllableList)
42                 if not AlgQuiet:
43                     print "%-20s added %s" % (LszWord, LliSyllableList)
44             else:
45                 if not AlgQuiet:
46                     print "  -Word (%s) found twice. First count was %s, second was %s" % (LszWord, GdcSyllableCount[LszWord],
47 LliSyllableList)
48     except:
49         print "An error was encountered processing the file."
50         raise IOError
51
52     try:
53         #-----
54         # Now write the dictionary away to a new pickle file
```

```
55
56          LhaOutputFile = open('cmusyllables.pickle','w')
57
58          if not AlgQuiet:
59              print "Finished processing input filennNow dumping pickle filen"
60          pickle.dump(GdcSyllableCount, LhaOutputFile,-1)
61
62          if not AlgQuiet:
63              print "Pickle file cmusyllables.pickle has been created."
64      except:
65          print "An error was encountered writing the pickle file."
66          raise IOError
67
68  def main():
69      #-----
70      # Open the CMU file and for each entry create a dict with the resulting
71      # number of syallbles
72
73      CreatePickle()
74
75  if __name__ == '__main__':
      main()
```

This results in a dictionary lookup that gives an accurate syllable count (or counts because some words have multiple pronunciations and therefore syllable counts) for the words it has in it's dictionary.

Words not in the Dictionary

But what about words that the dictionary doesn't know about? Well the way I handled that is to build a fallback routine into the code. The best (most accurate) mechanical routine I found was PHP-based and is part of Russel McVeigh's site:

http://www.russellmcveigh.info/content/html/syllablecounter.php

I ported Russel's code to Python and I added a couple of other exceptions that I found. Most of the mechanical syllable calculation routines I found, work on the following basic syllable rules:

1. Count the number of vowels in the word
2. Subtract one for any silent vowels such as the e at the end of a word
3. Subtract any additional vowels in vowel pairs/triplets (ee, ei, eau, etc.) i.e. each group of multiple vowels scores only one vowel

The number you have left is the number of syllables. However there then follows a series of adjustments where if certain patterns are recognised in the word, syllables are added in or taken away and then finally you end up with the correct syllable count. But, even with all this adjustment it's never accurate. But perhaps good enough for those words not in the cmudict.

So the code I've developed is really simple. It looks up syllable counts in the cmudict and returns the results if found and if not has a guess at the syllable count instead. I'd really like to share the code with you but something in my wordpress theme or the syntax highlighter that I use objects to something in the code. Perhaps, as I'm not a proper programmer it doesn't like my

esoteric, bastardised Hungarian notation variable names?

So I can't post it here at the moment but will try to get that fixed. If you're interested contact me and I'll happily share it.

Danny Goodall

Edit – It looks like I *might* have solved that problem by using a different syntax highlighter.

```
#!/usr/bin/env
python
```

```
1   #!/usr/bin/env python
2   try:
3       import cPickle as pickle
4   except:
5       import pickle
6
7   import re
8
9   class cmusyllables(object):
10
11      def __init__(self):
12
13          #-----
14          # Record the mode of the syllable count - manual / lookup
15
16          self.szMode = None
17
18          self.dcSyllableCount = None
19
20          #-----
21          # New structures for the SyllableCount3 routine
22
23          self.dcSyllable3WordCache = {}
24
25          self.liSyllable3SubSyllables = [
26              'cial',
27              'tia',
28              'cius',
29              'cious',
30              'uiet',
31              'gious',
32              'geous',
33              'priest',
34              'giu',
35              'dge',
36              'ion',
37              'iou',
38              'sia$',
39              '.che$',
40              '.ched$',
41              '.abe$',
42              '.ace$',
```

```
43              '.ade$',
44              '.age$',
45              '.aged$',
46              '.ake$',
47              '.ale$',
48              '.aled$',
49              '.ales$',
50              '.ane$',
51              '.ame$',
52              '.ape$',
53              '.are$',
54              '.ase$',
55              '.ashed$',
56              '.asque$',
57              '.ate$',
58              '.ave$',
59              '.azed$',
60              '.awe$',
61              '.aze$',
62              '.aped$',
63              '.athe$',
64              '.athes$',
65              '.ece$',
66              '.ese$',
67              '.esque$',
68              '.esques$',
69              '.eze$',
70              '.gue$',
71              '.ibe$',
72              '.ice$',
73              '.ide$',
74              '.ife$',
75              '.ike$',
76              '.ile$',
77              '.ime$',
78              '.ine$',
79              '.ipe$',
80              '.iped$',
81              '.ire$',
82              '.ise$',
83              '.ished$',
84              '.ite$',
85              '.ive$',
86              '.ize$',
87              '.obe$',
88              '.ode$',
89              '.oke$',
90              '.ole$',
91              '.ome$',
92              '.one$',
93              '.ope$',
94              '.oque$',
95              '.ore$',
96              '.ose$',
97              '.osque$',
98              '.osques$',
```

```
99              '.ote$',
100             '.ove$',
101             '.pped$',
102             '.sse$',
103             '.ssed$',
104             '.ste$',
105             '.ube$',
106             '.uce$',
107             '.ude$',
108             '.uge$',
109             '.uke$',
110             '.ule$',
111             '.ules$',
112             '.uled$',
113             '.ume$',
114             '.une$',
115             '.upe$',
116             '.ure$',
117             '.use$',
118             '.ushed$',
119             '.ute$',
120             '.ved$',
121             '.we$',
122             '.wes$',
123             '.wed$',
124             'yse$',
125             'yze$',
126             '.rse$',
127             '.red$',
128             '.rce$',
129             '.rde$',
130             '.ily$',
131             '.ely$',
132             '.des$',
133             '.gged$',
134             '.kes$',
135             '.ced$',
136             '.ked$',
137             '.med$',
138             '.mes$',
139             '.ned$',
140             '.[sz]ed$',
141             '.nce$',
142             '.rles$',
143             '.nes$',
144             '.pes$',
145             '.tes$',
146             '.res$',
147             '.ves$',
148             'ere$'
149         ]
150
151         #global $split_array;
152         self.liSyllable3AddSyllables  = [
153             'ia',
154             'riet',
```

```
155             'dien',
156             'ien',
157             'iet',
158             'iu',
159             'iest',
160             'io',
161             'ii',
162             'ily',
163             '.oala$',
164             '.iara$',
165             '.ying$',
166             '.earest',
167             '.arer',
168             '.aress',
169             '.eate$',
170             '.eation$',
171             '[aeiouym]bl$',
172             '[aeiou]{3}',
173             '^mc',ism',
174             '^mc',asm',
175             '([^aeiouy])1l$',
176             '[^l]lien',
177             '^coa[dglx].',
178             '[^gq]ua[^auieo]',
179             'dnt$'
180         ]
181
182         #-----
183         # Create a list of the compiled regex
184
185         self.liSyllable3RESubSyllables = []
186         self.liSyllable3REAddSyllables = []
187
188         for LszRegEx in self.liSyllable3AddSyllables:
189             LreRegEx = re.compile(LszRegEx)
190             self.liSyllable3REAddSyllables.append(LreRegEx)
191
192         for LszRegEx in self.liSyllable3SubSyllables:
193             LreRegEx = re.compile(LszRegEx)
194             self.liSyllable3RESubSyllables.append(LreRegEx)
195
196     def Load(self, AszFile = 'cmusyllables.pickle'):
197         try:
198             LhaPickleFile = open(AszFile,'rb')
199
200             self.dcSyllableCount =  pickle.load(LhaPickleFile)
201             #print "LOADED SYLLABLES"
202         except:
203             return( False )
204
205         return( True )
206
207     def GetRawDict(self):
208         return(self.dcSyllableCount)
209
210     def NonCMUSyllableCount(self, AszWord):
```

```
211
212         #LszWord = self._normalize_word( AszWord.lower() )
213         LszWord = AszWord
214
215         #-----
216         # If we've already seen this before then return the syllables
217
218         if LszWord in self.dcSyllable3WordCache:
219             return(self.dcSyllable3WordCache[LszWord])
220
221         #-----
222         #Split into parts on vowels and vowel sounds
223
224         LliWordParts = re.split(r'[^aeiouy]+', LszWord)
225
226         #-----
227         # Combine the valid parts of the word
228
229         LliValidWordParts = []
230
231         for LszValue in LliWordParts:
232             if LszValue &lt;&gt; '':
233                 LliValidWordParts.append(LszValue)
234
235         LinSyllables = 0
236
237         #-----
238         # Loop through the compiled regexs looking for matches
239
240         for LreSylRE in self.liSyllable3RESubSyllables:
241             LinMatch = 0 if LreSylRE.search(LszWord) is None else 1
242             LinSyllables -= LinMatch
243
244         for LreSylRE in self.liSyllable3READdSyllables:
245             LinMatch = 0 if LreSylRE.search(LszWord) is None else 1
246             LinSyllables += LinMatch
247
248         #-----
249         # Now compute the syllable count by the number of vowels
250
251         LinSyllables += len(LliValidWordParts)
252
253         #-----
254         # If we've not found any there must be at least 1
255
256         LinSyllables = 1 if LinSyllables == 0 else LinSyllables
257
258         #----
259         # Record this result in the word cache
260
261         self.dcSyllable3WordCache[LszWord] = LinSyllables
262
263         #-----
264         # Return the result
265
266         return(LinSyllables)
```

```
267
268     def SyllableCount(self, AszWord, AszMode = 'max', AlgFallBack=True):
269
270         if AszMode.lower() not in ['min','max','ave','raw']:
271             LszMode = 'max'
272         else:
273             LszMode = AszMode
274
275         LszWord = AszWord.lower()
276
277         if len(LszWord) == 0 or LszWord not in self.dcSyllableCount:
278             self.szMode = None
279             if len(LszWord) == 0 or not AlgFallBack:
280                 if AszMode in ['min','max']:
281                     return(0)
282                 elif AszMode in ['ave']:
283                     return(0.0)
284                 elif AszMode in ['raw']:
285                     return([])
286             else:
287                 LliSyllableList = list((self.NonCMUSyllableCount(LszWord),))
288                 self.szMode = 'manual'
289         else:
290             LliSyllableList = self.dcSyllableCount[LszWord]
291             self.szMode = 'lookup'
292
293         if LszMode == 'min':
294             return(min(LliSyllableList))
295         elif LszMode == 'max':
296             return(max(LliSyllableList))
297         elif LszMode == 'ave':
298             return(float(float(sum(LliSyllableList))/float(len(LliSyllableList))))
299         elif LszMode == 'raw':
300             return(LliSyllableList)
301         else:
302             return(None)
303
304     def GetSyllableMode(self):
305         #-----
306         # Return either None, manual or lookup depending on how the last
307         # syllable count was arrived at
308
309         return(self.szMode)
310
311 def main():
312
313     LzzSyllableCounter = cmusyllables()
314
315     LzzSyllableCounter.Load()
316
317     LliList =
318 ['','theatre','productized','productised','pumblechook','everything','altogether','particular','opportunity','everybody','cooeed','cueing']
319
320     for LszWord in LliList:
321         print "'%s' has max(%d), min(%d), ave(%3.2f), raw(%s) syllables - Calculated by (%s)" %
322 (LszWord,LzzSyllableCounter.SyllableCount(LszWord), LzzSyllableCounter.SyllableCount(LszWord,
```

```
    AszMode='min'),LzzSyllableCounter.SyllableCount(LszWord, AszMode='ave'),LzzSyllableCounter.SyllableCount(LszWord,
    AszMode='raw'),LzzSyllableCounter.GetSyllableMode())
323
    if __name__ == '__main__':
        main()
```

Posted in Language and Text Processing, Tips and tagged cmudict, gae, google app engine, nltk, notaproperprogrammer, python, syllables.

Post navigation

← FileZilla, SFTP and Amazon EC2
Inserting Google Chart Tools Visualizations... →

5 Comments

1. *Steve Hanov*
   29th March 2012 at 8:30 pm

   The Rhymebrain API (RhymeBrain API) also includes syllable counting, and uses the CMU dictionary. As a fallback, when it doesn't contain the pronunciation of a word, it derives the CMU using machine learning and then counts the number of syllables.

   - *Danny Goodall*
     30th March 2012 at 9:49 am

     Sounds interesting and it looks like a great site. I'll check it out Steve when I get moment.

     Dan

2. 
   richard
   20th June 2013 at 12:40 am

   You missed `d = cmudict.dict()` from your snippet of Jordan Boyd-Graber's code.

3. Pingback: Counting Syllables in the English Language Using Python | 42?

4. *stowe*
   21st August 2013 at 6:41 pm

as much as rhyming goes, i think http://www.prime-rhyme.com is the most accurate.

Comments are closed.

Search

Recent Posts

- Heroku Versus AppEngine and Amazon EC2 – Where Does it fit in?
- Inserting Google Chart Tools Visualizations into WordPress
- Counting Syllables Accurately in Python on Google App Engine
- FileZilla, SFTP and Amazon EC2
- arcanicity.appspot.com – How much jargon does your text contain?

Recent Comments

- Meitar on Inserting Google Chart Tools Visualizations into WordPress
- stowe on Counting Syllables Accurately in Python on Google App Engine
- Counting Syllables in the English Language Using Python | 42? on Counting Syllables Accurately in Python on Google App Engine
- richard on Counting Syllables Accurately in Python on Google App Engine
- cewood on FileZilla, SFTP and Amazon EC2

Archives

- April 2012
- September 2011
- June 2011
- May 2011

Categories

- Arcanicity
- Charting
- Cloud / Platform as a Service
- Cloud Stuff
- General
- Language and Text Processing
- Natural Language Processing (NLP)
- PHP
- Python
- Software Development
- Tips
- WordPress

Meta

- Log in
- Entries feed
- Comments feed
- WordPress.org

Connect