

```
27 """
28 Tutorial for the Million Song Dataset
29
30 by Thierry Bertin-Mahieux (2011) Columbia University
31 tb2332@columbia.edu
32 Copyright 2011 T. Bertin-Mahieux, All Rights Reserved
33
34 This tutorial will walk you through a quick experiment
35 using the Million Song Dataset (MSD). We will actually be working
36 on the 10K songs subset for speed issues, but the code should
37 transpose seamlessly.
38
39 In this tutorial, we do simple metadata analysis. We look at
40 which artist has the most songs by iterating over the whole
41 dataset and using an SQLite database.
42
43 You need to have the MSD code downloaded from GITHUB.
44 See the MSD website for details:
45 http://labrosa.ee.columbia.edu/millionsong/
46
47 If you have any questions regarding the dataset or this tutorial,
48 please first take a look at the website. Send us an email
49 if you haven't found the answer.
50
51 Note: this tutorial is developed using Python 2.6
52 on an Ubuntu machine. PDF created using 'pyreport'.
53 """
54
55 # usual imports
56 import os
57 import sys
58 import time
59 import glob
60 import datetime
61 import sqlite3
62 import numpy as np # get it at: http://numpy.scipy.org/
63
64 # path to the Million Song Dataset subset (uncompressed)
65 # CHANGE IT TO YOUR LOCAL CONFIGURATION
66 msd_subset_path='/home/thierry/Desktop/MillionSongSubset'
67 msd_subset_data_path=os.path.join(msd_subset_path, 'data')
68 msd_subset_addf_path=os.path.join(msd_subset_path, 'AdditionalFiles')
69 assert os.path.isdir(msd_subset_path), 'wrong path' # sanity check
70
71 # path to the Million Song Dataset code
72 # CHANGE IT TO YOUR LOCAL CONFIGURATION
73 msd_code_path='/home/thierry/Columbia/MSongsDB'
74 assert os.path.isdir(msd_code_path), 'wrong path' # sanity check
75
76 # we add some paths to python so we can import MSD code
77 # Ubuntu: you can change the environment variable PYTHONPATH
78 # in your .bashrc file so you do not have to type these lines
79 sys.path.append( os.path.join(msd_code_path, 'PythonSrc') )
80
81 # imports specific to the MSD
82 import hdf5_getters as GETTERS
83
84 # the following function simply gives us a nice string for
85 # a time lag in seconds
```

```
86 def strtimedelta(starttime, stoptime):
87     return str(datetime.timedelta(seconds=stoptime - starttime))
88
89 # we define this very useful function to iterate the files
90 def apply_to_all_files(basedir, func=lambda x: x, ext='.h5'):
91     """
92     From a base directory, go through all subdirectories,
93     find all files with the given extension, apply the
94     given function 'func' to all of them.
95     If no 'func' is passed, we do nothing except counting.
96     INPUT
97         basedir - base directory of the dataset
98         func    - function to apply to all filenames
99         ext     - extension, .h5 by default
100    RETURN
101        number of files
102    """
103    cnt = 0
104    # iterate over all files in all subdirectories
105    for root, dirs, files in os.walk(basedir):
106        files = glob.glob(os.path.join(root, '*' + ext))
107        # count files
108        cnt += len(files)
109        # apply function to all files
110        for f in files:
111            func(f)
112    return cnt
113
114 # we can now easily count the number of files in the dataset
115 print 'number of song files:', apply_to_all_files(msd_subset_data_path)
```

```
number of song files: 10000
```

```
88
89 # let's now get all artist names in a set(). One nice property:
90 # if we enter many times the same artist, only one will be kept.
91 all_artist_names = set()
92
93 # we define the function to apply to all files
94 def func_to_get_artist_name(filename):
95     """
96     This function does 3 simple things:
97     - open the song file
98     - get artist ID and put it
99     - close the file
100    """
101    h5 = GETTERS.open_h5_file_read(filename)
102    artist_name = GETTERS.get_artist_name(h5)
103    all_artist_names.add(artist_name)
104    h5.close()
105
106 # let's apply the previous function to all files
107 # we'll also measure how long it takes
108 t1 = time.time()
109 apply_to_all_files(msd_subset_data_path, func=func_to_get_artist_name)
110 t2 = time.time()
111 print 'all artist names extracted in:', strtimedelta(t1, t2)
```

```
all artist names extracted in: 0:01:35.553140
```

```
115
```

```
116
```

```
117
118
119 # let's see some of the content of 'all_artist_names'
120 print 'found', len(all_artist_names), 'unique artist names'

    found 4412 unique artist names

119 for k in range(5):
120     print list(all_artist_names)[k]

    Groundhogs
    Pale Forest
    The Real Kids
    JennyAnyKind
    Aswad

120
121 # this is too long, and the work of listing artist names has already
122 # been done. Let's redo the same task using an SQLite database.
123 # We connect to the provided database: track_metadata.db
124 conn = sqlite3.connect(os.path.join(msd_subset_addf_path,
125                                     'subset_track_metadata.db'))
126 # we build the SQL query
127 q = "SELECT DISTINCT artist_name FROM songs"
128 # we query the database
129 t1 = time.time()
130 res = conn.execute(q)
131 all_artist_names_sqlite = res.fetchall()
132 t2 = time.time()
133 print 'all artist names extracted (SQLite) in:', strtimedelta(t1, t2)

    all artist names extracted (SQLite) in: 0:00:00.011670

133 # we close the connection to the database
134 conn.close()
135 # let's see some of the content
136 for k in range(5):
137     print all_artist_names_sqlite[k][0]

    !!!
    (hed) p.e.
    089 Clique feat. Minnesota Snipe & Skinny Cueball
    089 Clique feat. Prophet
    1. Futurologischer Congress

139
140 # now, let's find the artist that has the most songs in the dataset
141 # what we want to work with is artist ID, not artist names. Some artists
142 # have many names, usually because the song is "featuring someone else"
143 conn = sqlite3.connect(os.path.join(msd_subset_addf_path,
144                                     'subset_track_metadata.db'))
145 q = "SELECT DISTINCT artist_id FROM songs"
146 res = conn.execute(q)
147 all_artist_ids = map(lambda x: x[0], res.fetchall())
148 conn.close()
149
150 # The Echo Nest artist id look like:
151 for k in range(4):
152     print all_artist_ids[k]

    AR009211187B989185
    AR00A6H1187FB5402A
    AR00LNI1187FB444A5
    AR00MBZ1187B9B5DB1
```

```

153
154 # let's count the songs from each of these artists.
155 # We will do it first by iterating over the dataset.
156 # we prepare a dictionary to count files
157 files_per_artist = {}
158 for aid in all_artist_ids:
159     files_per_artist[aid] = 0
160
161 # we prepare the function to check artist id in each file
162 def func_to_count_artist_id(filename):
163     """
164     This function does 3 simple things:
165     - open the song file
166     - get artist ID and put it
167     - close the file
168     """
169     h5 = GETTERS.open_h5_file_read(filename)
170     artist_id = GETTERS.get_artist_id(h5)
171     files_per_artist[artist_id] += 1
172     h5.close()
173
174 # we apply this function to all files
175 apply_to_all_files(msd_subset_data_path, func=func_to_count_artist_id)
176
177 # the most popular artist (with the most songs) is:
178 most_pop_aid = sorted(files_per_artist,
179                      key=files_per_artist.__getitem__,
180                      reverse=True)[0]
181 print most_pop_aid, 'has', files_per_artist[most_pop_aid], 'songs.'
```

AROIHOI122988FEB8E has 13 songs.

```

182
183 # of course, it is more fun to have the name(s) of this artist
184 # let's get it using SQLite
185 conn = sqlite3.connect(os.path.join(msd_subset_addf_path,
186                                     'subset_track_metadata.db'))
187 q = "SELECT DISTINCT artist_name FROM songs"
188 q += " WHERE artist_id='"+most_pop_aid+"'"
189 res = conn.execute(q)
190 pop_artist_names = map(lambda x: x[0], res.fetchall())
191 conn.close()
192 print 'SQL query:', q
```

SQL query: SELECT DISTINCT artist_name FROM songs WHERE artist_id='
AROIHOI122988FEB8E'

```

193 print 'name(s) of the most popular artist:', pop_artist_names

name(s) of the most popular artist: [u'Mario Rosenstock']
```

```

194
195 # let's redo all this work in SQLite in a few seconds
196 t1 = time.time()
197 conn = sqlite3.connect(os.path.join(msd_subset_addf_path,
198                                     'subset_track_metadata.db'))
199 q = "SELECT DISTINCT artist_id, artist_name, Count(track_id) FROM songs"
200 q += " GROUP BY artist_id"
201 res = conn.execute(q)
202 pop_artists = res.fetchall()
203 conn.close()
204 t2 = time.time()
```

```
205 print 'found most popular artist in', strtimedelta(t1, t2)
```

```
found most popular artist in 0:00:00.055763
```

```
206 print sorted(pop_artists, key=lambda x:x[2], reverse=True)[0]
```

```
(u'AROIHOI122988FEB8E', u'Mario Rosenstock', 13)
```