GO FOR IT Q4

s336651 大垣慶介

2012年2月7日

q4_music_feature.py のドキュメント

```
================================================================================
q4_music_feature.py
--------------------------------------------------------------------------------

arguments
   |- 1.targetmusicstring
   |- (2.targetnote)
       |- default : None
output
   |- feature = $(feature)
   |- (if targetnote )
        |- answer melody is..
        |- $(melody)
example
   |- $ q4_music_feature.py
      3:4:,2:4:,1:4:,0:4:,1:4:,2:4:,3:4:,:4:,1:8:,0:4.:,-1:4:,-2:4:,:8:,-1:4.:,0:8:,1:8:

        |- output :
            |- feature= 24
   |- $ q4_music_feature.py
      3:4:,2:4:,1:4:,0:4:,1:4:,2:4:,3:4:,:4:,1:8:,0:4.:,-1:4:,-2:4:,:8:,-1:4.:,0:8:,1:8:

        |- output :
            |- feature= 24
            |- answer melody is...
            |- 3:4:,2:4:,3:4:,3:4:b,2:4:,3:4:,2:4:,:4:,4:8:s,4:4.:,3:4:b
               ,4:4:,:8:,3:4.:b,4:8:,3:8:
```

表 1: 環境

言語	python2.7
OS	Linux(Ubuntu10.10)
CPU	Intel(R) Core(TM) i5 CPU M 480 @ 2.67GHz
メモリ	3780088 kB

```
|- $$ q4_music_feature.py 3:4:,2:4:,3:4:,3:4:b,2:4:,3:4:,2:4:,:4:,4:8:s
    ,4:4.:,3:4:b,4:4:,:8:,3:4.:b,4:8:,3:8:
     |- output :
         |- feature= 24
```

```
for Python 2.7
K.Ogaki(ogaki@iis.u-tokyo.ac.jp)
2012/02/07
```

```
This program is released under GPL.
http://www.opensource.jp/gpl/gpl.ja.html.euc-jp
```

仕様

上記のドキュメントのように、 $q4_music_feature.py (targetstring) と入力することで targetstring の feature を計算する (1)(2)。さらに $q4_music_feature.py (targetstring) (targetnote) と入力することで tagetnote で終わり、 targetstring と同じ feature の文字列を返す。引数が 0 の場合ドキュメントを出力する。実行環境は表のとおりである。

アルゴリズム

問 (1)(2),feature の計算

feature を半音を 1 とした隣接音符との距離の旋律全体での総和とした。ただし休符はスキップする。擬似コードで以下に示す。

```
### pseudocode ###
feature = sum [ notesdistance( note[i],note[i+1]) for all notes]
```

結果は “出力” の章に示す

```
def notetonote(note1, note2):
    dist = 0
    for i in xrange(int(note1[0]), int(note2[0]) ):
        octnote = i%7
        if(octnote == 0 or octnote == 3):
            dist += 1
        else:
            dist += 2

    # --- accidental ---臨時記号の処理
    (...)
```

上のように愚直に note1 から note2 への 2 音符間の差をを計算している。さらにこれを note1,2 に対称性を持たせて距離とするための notesdistance(note1, note2) と和を取って feature とするための calcfeature(musicstring) を用意してある。
ちなみに出力の章より問 (ii) の答えは feature(B)=feature(D)

問 (3)

問題設定が非常に自由度が高かったので幾つか制約を追加した。

- targetnote として設定した音符で終わる旋律を探索する。
- 開始音は入力譜面と同一 (譜面 G が-2 からはじまっているため、この制約で問題の制約を満たす)
- 音符の長さ、休符は入力譜面と同一
- note[i] と note[i+1] の距離は入力譜面と同一 (この制約は必ず特徴量を同じにする)

これらの制約を満たす解を探索する問題に設定した。
制約 2,3,4 から、これは次の音符を上にシフトするか下にシフトするかの 2 分探索の問題となる。よって入力譜面の音符数を n として、$O(2^n)$。今回は targetnote(-2 など) に近づける問題となるので、この targetnote からの距離をコストとした priority queue を用いて探索を行なった。

```
class searchitem(object):
    def __init__(self, note, path):
        self.note = note
        self.path = path
        self.distance = notesdistance(self.note, [targetnote, '', ''] )
    def __le__(self, other):
        return self.distance < other.distance
```

このような, これまでの path('+','+','-'... など) と今の音符、今の targetnote からの距離 distance をもつ searchitem クラスとして、探索の葉ノードを格納している。これを distance で優先度順に取り出して、targetnote に至ったら終了とする。__le__が示しているのは何を priority の比較に用いるかである。以下に擬似コードを示す。

```
### pseudocode ###
def searchmelody( targetmusic ):
    queue = [searchitem(targetmusic[0], [])]
    while(1):
        nowsearchitem = heappop(queue)
        if(len(nowsearchitem.path)==len(targetdists)): ## leaf node
            if(nowsearchitem.note[0] == targetnote): ## find !!
                break
        else:
            shift = targetdists[len(nowsearchitem.path)]
            heapq.heappush(queue, searchitem( shiftnote(nowsearchitem.note, +
                shift), nowsearchitem.path+["+"] ) )
            heapq.heappush(queue, searchitem( shiftnote(nowsearchitem.note, -
                shift), nowsearchitem.path+["-"] ) )
```

出力の章より、問 (iii) の答えは “-2:8.:,-1:16:,-2:8.:,-1:16:,-2:4:,-4:4:,-4:8.:,-3:16:,-3:8.:s,-3:16:,-3:4:s,-1:4:s,-2:8:b,:8:,-1:8.:s,-1:16:b,-2:8:b,:8:,-1:8.:s,-1:16:b,-2:8.:b,-2:16:b,-1:8.:,-1:16:,-4:8.:,-3:16:s,-2:4:s” となる

出力

```
# feature(A)
$ ./q4_music_feature.py -3:4:,-2:4:,-1:4:,0:4:b,-1:4:,-2:4:,-3:4:,:4:,-1:4:,0:4:
    b,1:4:,2:4:,1:4:,0:4:b,-1:4:,:4:
feature= 24
# feature(B)
$ ./q4_music_feature.py 3:4:,2:4:,1:4:,0:4:b,1:4:,2:4:,3:4:,:4:,1:4:,0:4:b
    ,-1:4:,-2:4:,-1:4:,0:4:b,1:4:,:4:
feature= 26
# feature(C)
$ ./q4_music_feature.py
    3:4:,2:4:,1:4:,0:4:,1:4:,2:4:,3:4:,:4:,1:8:,0:4.:,-1:4:,-2:4:,:8:,-1:4.:,0:8:,1:8:

feature= 24
# feature(D)
$ ./q4_music_feature.py
    -6:8:,-6:8:,-6:8:,-4:8:,-2:8:,-2:8:,-2:8:,:8:,-5:8:,-5:8:,-5:8:,-3:8:,-2:8:,-2:8:,-2:8
    b,-1:8:,-1:8:,-1:8:,-1:8:b,-2:4:,0:4:,1:4:,:4:
feature= 26
# feature(E)
$ ./q4_music_feature.py
    -6:8.:,-7:16:,-6:8:,-5:8:,-4:8:,-4:8:,-4:4:,-3:8.:,-4:16:,-5:8:,-6:8:,-5:4:,:4:,-3:8:,-

feature= 28
# feature(F)
$ ./q4_music_feature.py
    -6:2:,-5:4:,-6:8:,-5:8:,-4:4:,-2:4:,-4:4:,:4:,-5:4:,-5:4:,-6:4:,-5:4:,-4:2.:,:4:

feature= 22
# feature(G)
$ ./q4_music_feature.py
    -2:8.:,-1:16:,-2:8.:,-1:16:,-2:4:,-4:4:,-4:8.:,-3:16:,-4:8.:,-3:16:,-4:4:,-6:4:,-4:8:,

feature= 51

#melody from G
$ ./q4_music_feature.py
    -2:8.:,-1:16:,-2:8.:,-1:16:,-2:4:,-4:4:,-4:8.:,-3:16:,-4:8.:,-3:16:,-4:4:,-6:4:,-4:8:,
    -2
```

```
feature= 51
answer melody is...
-2:8.:,-1:16:,-2:8.:,-1:16:,-2:4:,-4:4:,-4:8.:,-3:16:,-3:8.:s,-3:16:,-3:4:s
    ,-1:4:s,-2:8:b,:8:,-1:8.:s,-1:16:b,-2:8:b,:8:,-1:8.:s,-1:16:b,-2:8.:b,-2:16:b
    ,-1:8.:,-1:16:,-4:8.:,-3:16:s,-2:4:s
#check feature(melody from G)
$ ./q4_music_feature.py $(./q4_music_feature.py
    -2:8.:,-1:16:,-2:8.:,-1:16:,-2:4:,-4:4:,-4:8.:,-3:16:,-4:8.:,-3:16:,-4:4:,-6:4:,-4:8:,
    -2 2|tail -1)
feature= 51
```