Query by Humming System

Dong In Lee

MA/MST ’07
System Overview

- Pitch detection
- Note segmentation
- Get note interval
- Make query string
- Compare with DB
- Show song name
Pitch Detection

► YIN algorithm

Based on the well known autocorrelation method for detecting a signal’s periodicity.

Since the basic autocorrelation method has high error rates, YIN introduces several modifications to reduce the errors while maintaining simplicity and low-latency in its implementation.
Note Segmentation

Set the threshold value for slope between two pitches
Note Interval & Query String

►► People can’t hum in original key for any song. Thus, instead of using absolute key note, I used note interval.

Ex) C C G G A A G  =>  0 7 0 2 0 -2

►► I made simple scheme for converting numbers to string

Ex) 0 7 0 2 0 -2  =>  0 g 0 b 0 B
Database specification

► Same scheme was used to make DB

Ex) Happy birthday song

0bBeAD0bBgBE0lCDABh0ADAa
String Compare Algorithm

1st trial: String Edit Distance Algorithm

Principle
- Query: 0bDA
- Database: 0bBeA
- Distance: 2

Problem
- Query: 0bDA
- Intended DB: 0bBeAD0bBgBE0lCDABh0ADAa
- Distance: 21
- Vicious DB: hhhhh
- Distance: 4!
String Compare Algorithm Con’t

2\textsuperscript{nd} trial : LCS Algorithm

- Principle

  Query : 0bDA

  Intended DB : 0bBeAD0bBgBE0lCDABh0ADAa

  => LCS Length : 3

  Vicious DB : hhhh

  => LCS Length : 1

- Problem

  More Vicious DB : xx0xxxxbxxA

  LCS Length : 3
3rd trial: Modified LCS Algorithm

- Principle
  Check adjacent characters

If common characters are found, modified algorithm check to see if the adjacent characters are also verified to common characters.

=> The more adjacent common characters there exists, the higher value the query has.

Cf. Time complexity: $O(m*n)$ with Dynamic programming technique
String Compare Algorithm Con’t

- Modified LCS Algorithm

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>b</th>
<th>B</th>
<th>e</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Value of intended DB
0 -> 3 -> 2 -> 1 ->
0 -> 3 -> 6
String Compare Algorithm Con’t

- Modified LCS Algorithm

Value of vicious DB

0 -> 3 -> 2 -> 1 ->
0 -> 3 -> 2 -> 1 ->
0 -> 3
Conclusion

►► Evaluation
This QBH system shows good performance when the user makes “good” query

 ► Future Work
- Fast transition detection
- More elaborated algorithm for string matching