AN IMPLEMENTATION OF WEB BASED QUERY BY HUMMING SYSTEM

Lujia Chen\textsuperscript{1,2}, Bao-Gang Hu\textsuperscript{1,2}

\textsuperscript{1} National Laboratory of Pattern Recognition, Institute of Automation
\textsuperscript{2} Beijing Graduate School, Chinese Academy of Sciences,
{ljchen, hubg}@nlpr.ia.ac.cn

ABSTRACT

This paper presents a new prototype of web-based query by humming (QbH) system, which is designed specifically for music retrieval on web applications. Besides the humming input interface, the system also provides an alternative interface of note strike. Schemes for signal processing and music indexing, such as inverted index and contour snippet are discussed in detail. In addition, a preliminary test was also provided, showing that the performance is acceptable for web search service.

1. INTRODUCTION

In recent years, music information retrieval seems receiving more attentions. The conventional and most popular method to search music via Internet is using a keyword (such as title, album, pop star, etc.) in the web search engine. The limitation of this method is, it cannot work if one user can only remember music itself, rather than any text keyword. Query by Humming (QbH) system provides a new way to search music by music itself. QbH system enables user to hum a short part of the music, then it extracts some characters from the sound input signal, searches the music database by these characters, and finally returns some musics containing the specific part of melody which the user hummed.

In this work, we propose a web based music search engine, which takes a clip of melody as keyword. There are two ways to input melody: humming input and note strike input. Humming input is described before, which is the most nature way to input music. But it requires user's humming tone to be reasonably accurate, as well as a silent environment (high Signal to Noise Ratio, SNR). The note strike input method provides a piano keyboard on the screen for the user strikes the corresponding piano key by clicking mouse. This method has the most accuracy in pitch, as well as no necessary to perform a series of complex signal process operations, but it is difficult for non-professional user to click keys with mouse in accurate tempo since the note distance is different and the mouse cannot move fast enough to catch the tempo. In this system, we developed a unified solution to get rid of the tempo error problem, in both input modes. Therefore, the present system is designed for considerations of both cases.

A well-organized index is an essential part in a search engine. Index is a reorganization of raw data. A good index algorithm can remarkably reduce the time complexity of online search calculation. There are several indexing techniques in text IR field, such as Inverted Index, Suffix Tree, Suffix Array etc.\textsuperscript{[3456]} In this system, we employ Inverted Index concept into our indexing system, considering the difference between text and music, a “Contour Snippet” indexing method is developed.

The system uses the MIDI music format as the raw data, since the MIDI is digitalized music, it is easy to extract melody information.

The rest of the paper is organized as follows. In section 2, an overview of system architecture is given. In section 3, 4 and 5, signal process, index, and online search feature are introduced respectively. In section 6, an evaluation of the system is presented.

2. SYSTEM ARCHITECTURE

Like most Browser/Server (B/S) systems, the presented system is composed of server component and client component (shown in Fig. 1). The client component is provided to user after it is wrapped in a browser plugin. Java Applet technology is selected for its cross platform capability. Client component is in charge of collecting user's input, processing the sampled signal and extracting the melody note series, then it sends the series to server. It will also represent the search result to the user. The server component works on collecting raw data(MIDI file) and building index. It performs searching when the query request comes from clients. First, the Data Retrieve and Analysis module collects raw MIDI file from data source, which can be web crawler, static file, database, etc. The system also provides APIs for developers to extend the supported data source. Second, the Index module converts the massive raw data into a well-organized index format. The index enables Online Search module to get the target result very promptly. Another module is Online Search module. It responds client's query request, performs searching and feedback results.
3. SIGNAL PROCESS

The main task of Signal Process module in the client component is to analysis the wave input signal which represents the user humming and converts the wave signal into digitalized note series. This module works in 4 steps listed below.

3.1 Pre-processing on humming signal

Signal sampled from sound card usually contains DC offset, DC signal will bring error in frequency domain at low frequency area. In this step, we perform a filter to remove the DC component, then amplify and normalize the signal.

3.2 Spectrum power matrix generation

Human voice's base frequency has the range from 60Hz to 1000Hz. We select 8000Hz as the recording signal sample rate. At this step, we assume the humming voice signal is stationary in 100 millisecond, so we set the frame size 512 points, with 50% overlap. We perform FFT transform on every frame after a Hanning window is imposed. Then, the power of each complex value in spectrum is obtained. The spectrum power matrix is a Time-Frequency plane, every value on this plane represents the power at the specific frequency and the specific time(frame).

3.3 Pitch Tracking

Every music note has a fixed frequency and time duration. It means in the spectrum power matrix, the power will distribute at the base frequency and n-th harmonic of the corresponding note. The base frequency of the humming note can be derived as follow:

Define $P(t,f)$ as the power at the $t$-th time frame and frequency of $f$ Hz,

$$S(t,f) = \sum_{n=1}^{5} P(t, nf) \quad (f > 0) \quad (1)$$

$S(t,f)$ is at the $t$-th time frame, the the total power of base frequency and its harmonics. Here, we assume the harmonic higher than the 5-th order can be ignored, so we calculate the sum from $n=1$ to 5. If $f$ is the base frequency of the hummed note at the $t$-th frame, $S(t,f)$ should have a peak value.

We can get the $t$-th frame base frequency $f_{base}(t) :$

$$f_{base}(t) = \log_{2} \left( \frac{\arg\{ \max \{ S(t,f) \} \}}{f < 2000 \text{ Hz}} \right) \quad (2)$$

If there exist $k, m \in N$, such that

$$\max \left\{ f_{base}(k+i) - \bar{f} \right\} < \frac{1}{2} c , \quad (3)$$

where $i=0,1,2,..m-1$, $m$>3,

$$\bar{f} = \frac{1}{m} \sum_{i=0}^{m-1} f_{base}(k+i) , \quad (4)$$

and $c$ is a standard log frequency difference of a semitone. It is a threshold value to identify pitch transition. Then we can recognize the $k$-th, $(k+1)$-th, ..., $(k+m-1)$-th frames as a note, with the average frequency of $\bar{f}$, Fig. 2 is an example of $t$-frames plot.

3.4 Pitch Difference Calculation

Based on average frequencies of every note, pitch different $D_j$ can be calculated by:

$$D_j = \left\lfloor \frac{\bar{f}_j - \bar{f}_{j+1}}{c} \right\rfloor , \quad (5)$$

then the series of $D_j (j=1,2,3,...n)$ is the search keyword. In the note strike input method, the $D_j$ series can be derived by neighbored note distances.

Here, we ignored the time duration of each note, because the inaccuracy of note strike tempo.
4. MELODY INDEX

Most previous works[1][3][6][7] of QbH research use the linear scan method to match the target music, which is \text{O(n)} complexity algorithm. The search speed is unacceptable if the music database is large. In our system, inverted index algorithm is employed to improve the search performance in large database.

4.1 Contour Snippet

As we stated above, melody can be interpreted as Pitch Difference(PD) series. The whole music is a long PD series. We divide the long PD series into small clips, the cluster of similar clips is called Contour Snippet.

Contour Snippet can be understood as local illusion character of melody. For example, regardless the note time duration, “Do-Mi-So” is a 3 notes Contour Snippet, which appears in musics very frequently. “Do-Mi-Si” is another 3 notes Contour Snippet, which does not so frequently appear in music. Although there is only one note different between the above two Contour Snippets, the intuitive impressions of them are totally different.

4.2 Inverted Index

As [2] presented, Inverted Index algorithm is an effective one to improve text search performance. Conventionally, it is a “word based” index algorithm, which regroups the candidate items by common words included in each item. Regarding the most text article, the space is the natural word separator, so inverted index is easy to apply on text information. Our idea is to find a similar mechanism to divide music melody into “word”, then apply inverted index on melody. Music melody is different from text, it is constituted by several long strings of note, each note has two attributes: pitch and duration. The note series also includes a pause space. It seems to be similar to the text space, but the pause space in music melody is irregular, it is varied from different music, style, tempo, etc. Therefore, the pause space is not qualified to separate melody.

In Section 4.1, we introduced a concept called contour snippet, which is taken as the “word” of music in this work. Contour Snippet is composed of a few neighbored PDs, every contour snippet represents some character of the melody. The combination of several neighbored contour snippets will provide a clear character of the melody. Inverted index on music melody can be built based on contour snippet. The following paragraph will give an example to illustrate the step of building a melody inverted index.

Take 3 notes “Do-Mi-So” as the example. We count the distance \(d\) of each pair of notes (in semitone). In this case, we get two distance value \((4,3)\) according to Fig. 3. Then \((4,3)\) is a contour snippet. Note series in a music can be divided into a series of contour snippets. This is compared to word in text article. All methods of inverted index can be applied on contour snippet.

4.3 Index building

The raw data of the system is MIDI music files. In a search engine database, there are thousands of MIDI music. Before the online search service is provided, the index must be built in advance. The system will traverse the MIDI database, extract the contour snippets in every MIDI file, then add the MIDI file to every contour snippet music list. The first time building the index may take several minutes, about 0.5 sec per file. The system will save the index structure and data on the disk, once the system is restarted, just reload the index data from disk. If new music is added to the MIDI database, just insert it into the index list, without rescanning all the music in the database.

5. ONLINE SEARCH

Our system is a web based system, and it will serve the client's query simultaneously. And the server must response the client's query request very promptly. It requires the online search module to reach a high performance in order to retrieve the search result within an acceptable delay.

5.1 Candidate filter

As we have stated previously, the system uses contour snippet as the key in the index. In this case, we must convert any user query input into contour snippet.

In note strike input method, we can get the contour snippet directly from the note distance shown in Fig. 3. In section 3, we introduced the signal process method. In this procedure, we finally get a series of Pitch Differences. Pitch Difference is equivalent to note distance. So 2–3 neighbored Pitch Differences will constitute a contour snippet. On average, user hums 8 notes in 10 second, if contour snippet length is set as 2, we can get 6 contour snippets. Generally, the number of contour snippets \(c\) we can get from the user input is

\[ c = h - l \]

where \(h\) is the note count of the user input, and the \(l\) is the contour snippet length.

In the inverted index structure, every contour snippet key maps to a music set, all the elements in the set is related to the key. From each query, we can get \(c\) keys, so we get \(c\) related music sets.
\[ S_i \quad (i=1, 2, \ldots, c) \]
Then, we perform a union operation on these sets,
\[ R = S_1 \cup S_2 \cup \ldots \cup S_c \]
(7)
Set \( R \) is candidate music set.
At this step, all the non-related musics are excluded, and the search scale is reduced by a simple set union operation.

5.2 Candidate ranking

Ranking the search result is an important feature in search engine systems. Rank represents the similarity between the query keyword and the candidate item. In our melody search engine, hit rate of the contour snippet is employed to calculate the rank of the candidate music.

Contour snippet hit rate is defined as the number of sets which contain \( \nu \):
\[ r = \frac{1}{c} \sum_{i=1}^{c} L(\nu, S_i) \]  
(8)
where the \( L \) is the boolean function:
\[ L(\nu, S) = \begin{cases} 1 & \nu \in S \\ 0 & \nu \notin S \end{cases} \]  
then \( r \) is the hit rate. Hit rate represents the similarity between the query and candidate. In our system, the search result set is sorted by the hit rate in decent order, ensuring most similar candidate will appear at the top of list. If the user input is not accurate enough, for example, one or two notes missing or with error, the target music is still in our candidate list.

6. PERFORMANCE EVALUATION

Before our test, we collected a music library with 435 MIDI files, and one humming query with 10 notes. We test both the client and the server components on one PC, which is equipped Intel P4 3GHz CPU, 1G RAM, Linux operation system and Java Runtime Environment 1.5.

First, the client component is tested by a humming input. Second, the server component is tested using the query generated by the client. In this test, we performed a full index building operation, so the system traverses the whole MIDI dataset, then uses the index to generate the ranked result set. Table 1 shows the data of the test.

**Table 1:** Test on the system

<table>
<thead>
<tr>
<th>Client</th>
<th>Signal Format</th>
<th>8000 Hz 16 bits wave</th>
<th>Time duration</th>
<th>7.2 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Signal Process Time Cost</td>
<td>3.1 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recognized Note Count</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual Note Count</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td>Number of MIDI files</td>
<td>435</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Building index time cost</td>
<td>49 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Online search time cost</td>
<td>0.1 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of candidates</td>
<td>79</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Target music position</td>
<td>Top 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The search result set is sorted by the ranking point, Fig. 4 shows the ranking points of top 10 candidates in the result list, and the first 2 candidates are two versions of target music. It shows the rank of the candidate can remarkably identify the target and non-target music.

![Candidate Ranking Points](image)

Fig. 4: Candidate Ranking Points.

7. SUMMARY

In this work, an implementation of a melody-based music search engine system is proposed. The performance evaluation shows that online search in 435 files costs only 0.1 sec, which is acceptable for a search engine. In addition, inverted index algorithm ensures the time complexity of online search to be better than \( O(n) \). The future work will focus on making the system to be able to search in larger database, and making the humming input recognition more robust.

8. REFERENCES