Song Search and Retrieval by Tapping

Geoffrey Peters, Caroline Anthony, and Michael Schwartz

School of Computing Science, Faculty of Business Administration, and Cognitive Science Program, Simon Fraser University
8888 University Drive, Burnaby, BC, Canada, V5A 1S6
gpeters@sfu.ca, canthony@sfu.ca, mpschwar@sfu.ca

Abstract

We present an interactive, web based system for musical song search and retrieval, using rhythmic tapping as the primary means of query input. Our approach involves encoding the input rhythm as a contour string, and using approximate string matching to determine the most likely match with songs in the database.

Introduction and Previous Work

It would be desirable and useful to have a content-based search mechanism for music that does not rely on knowledge of metadata such as name, composer, or related keywords. We present an online, web based system[1] that allows the user to tap a portion of the rhythm of a song in order to search for it.

The motivation behind research in this area ranges from the highly academic to practical commercial applications. Melucci et al (1999) emphasize that music is an important form of cultural expression, and with increasing digital access, librarians want more effective methods for organizing and retrieving it than current text-based ones. Kosugi et al (2000) describe a Karaoke machine that lets users select the song they want by singing part of it.

There are several commonalities among existing research into content-based music search and retrieval systems. The works that we examined each described some sort of matching algorithm, though some varied in which aspects of musical structure were used in the analysis (e.g. pitch, melody, rhythm). They also all tried to allow for and deal with errors to varying degrees. There was also much variation in the format of the input query for the content-based search. Some used MIDI input, while others used forms such as query by humming.

The overriding focus in most of these works is on the melody of songs, since that is the musical component most easily identifiable by musically untrained users (Melucci et al, 1999). Query-by-humming systems (Kline & Glinert, 2003; Kosugi et al, 2000) include the ability to process a simplified form of human vocal performance (in Kosugi et al's case, allowing only the syllable 'ta') to find the melody, which is just an extra step before they face the same problem as the others, which is how to analyze the songs and find matches. Here we see some differences. Melucci et al (1999) use the notion of melodic surface being segmented into phrases, where correspondences between songs can be found. Kline & Glinert (2003), as well as Kosugi et al (2000) focused on trying to improve robustness against specific types of user errors, and Kline & Glinert noted that the expected user error will exceed the abilities of most prior systems' limited error handling.

Some researchers have found ways to transform the problem domain from that of searching music to searching text. Tseng (1999), for example, encodes the melodic (pitch) contour as a string, so that existing approximate text matching algorithms can be leveraged. In our approach, we encode the rhythmic contour as a string, which is based on note durations instead of pitches. For the matching process itself we saw *n*-grams and Hidden Markov Models discussed, as well as minimum edit distance algorithms, of which we chose to use Sun & Manber's (1992) fast approximate string matching implementation.

Our Approach

User Input Method. By accessing our web page, visitors can tap the rhythm of a song's melody using the space bar on their computer keyboard. Our Java applet will generate a MIDI file, which is sent to our application server for analysis, and the database will be searched. The search results indicating the name of the songs best matching the input will be displayed in the user's web browser. Alternatively, a MIDI file can be created using a MIDI keyboard, and sent to our server for analysis. Once the MIDI input is received by our server, it will be passed to our MIDI analyzer module.

MIDI Analyzer Module. Our implementation takes a MIDI file containing a monophonic sequence of notes, and analyzes the rhythm to generate a rhythmic contour string. In this discussion, we consider a "rhythm" to be the sequence of note durations and rests in the MIDI file, and we consider a "beat" to be the length of time from the start of a note to the beginning of the next note. Our MIDI analyzer module is programmed to normalize the durations of the beats, in order to eliminate any global tempo dependence when searching for matches. To normalize the durations, the average duration of a beat is calculated, and

[1] Web: http://cgi.sfu.ca/~gpeters/tapper/tapper.cgi

then each beat's duration is divided by the average duration, resulting in an average note having a duration of 1. For a particular song that is being analyzed, the graph of the normalized duration can be plotted per beat, as beats progress through time. It can be observed that each song has a somewhat unique "duration function", by comparing these "duration plots" for various songs. An example duration plot for the children's song "Are You Sleeping" is shown in Figure 1.

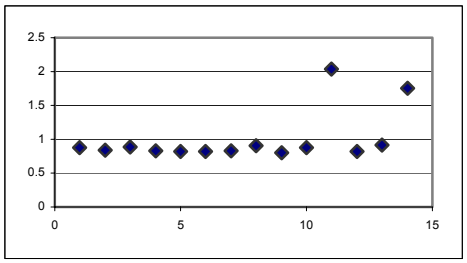

Figure 1 – "Are You Sleeping" Duration Plot

The goal of the MIDI Analyzer Module is to encode this duration information as a rhythmic contour string, so approximate string matching can be used. The rhythmic contour is calculated by finding the difference in duration of each pair of consecutive beats. It may be illustrative to examine a plot of the rhythmic contour for an example song (Figure 2), and compare it to the duration plot for the same song (Figure 1).

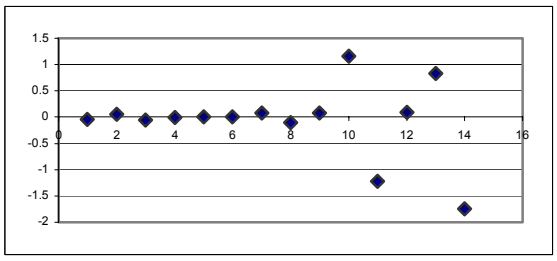

Figure 2 – "Are You Sleeping" Rhythmic Contour Plot

Once the rhythmic contour has been calculated, it is encoded as a string using the algorithm described in Figure 3. The result of the algorithm is that a song of n+1 beats is encoded into a string with n characters, with the three symbols *s*, *d*, and *u* (meaning the duration stays the same, goes down, or up, respectively).

Each song in the library database has been pre-processed and a contour string was generated and stored for each. A contour string is also generated for the search input. Then, the approximate string matching algorithm from Sun & Manber (1992) is used to calculate the edit distance between the input string and each string in the database. The edit distance is defined as the number of operations needed to transform one string into another string. The song in the database with a corresponding contour string that has the least edit distance to the input's contour string is considered to the most likely song for which the user is searching.

Algorithm for duration contour string generation.

Input: duration contour values (c1…cn), threshold T
Output: contour string (s)

For each duration contour value (ci) do:

If abs(ci) < T then	Append "s" to string s
Else if ci < 0 then	Append "d" to string s
Else if ci > 0 then	Append "u" to string s

Figure 3

Results. By experimentation we found that around 15 notes should be entered to get accurate search results with a database of 30 children's songs. The algorithm used allows for successful matching even by non-musicians who make errors in their search input. However, the system is only able to find songs which have been previously added to the database, so a training feature will be added in the future to allow visitors to train the system.

Conclusion

Our interactive web site demonstrates that rhythmic tapping can be an effective means of song search and retrieval, at least for a database with a limited number of songs. Such a tool could potentially be useful in a children's game which teaches the concept of rhythm, and on a larger scale could be used as a search method for an online music store. Our intention is to expand the software's capabilities by adding a training feature, but in its current form it is sufficient as a "proof of concept".

References

Kline, R.L., and Glinert, E.P. 2003. Music: Approximate matching algorithms for music information retrieval using vocal input. In *Proceedings of the eleventh ACM international conference on Multimedia,* 130-139. New York, NY: ACM Press.

Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M., and Kushima, K. 2000. A practical Query-by-humming system for a large music database. In *Proceedings of the eighth ACM international conference on Multimedia,* 333-342. New York, NY: ACM Press.

Melucci, M., and Orio, N. 1999. Musical information retrieval using melodic surface. In *Proceedings of the fourth ACM conference on Digital libraries,* 152-160. New York, NY: ACM Press.

Sun, W., and Manber, U. 1992. Fast text searching: allowing errors. In *Communication of the ACM. 35(10),* 83-91. New York, NY: ACM Press.

Tseng, Y.H. 1999. Content-based retrieval for music collections. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 176-182. New York, NY: ACM Press.