Volume 5, Issue 2, February 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Ease of Music Retrieval Using Singing/Humming Based Querying

Pritesh Patil*, Shruti Mengade, Prapti Kolpe, Vaishanvi Gawande, Kalyani Budhe

AISSMS' IOIT, Department of Information Technology, S.P. Pune University,

Pune, Maharashtra, India

Abstract— User will feel more delightful, if he/she can search the song by just singing or humming section of song. This can be helpful when user is unable to recall the attributes of the song like song title, name of artist, name of album. In the existing systems, results re-ranking has not been integrated into the query by singing/ humming systems. Our method is divided into two phases. In the first phase, we find the possible search results. This is similar to the conventional singing/humming query process. During the second phase, the musical preference of the user is utilized to rank the possible search results again. Songs that are most likely to be queried would be positioned at the front of the list in the search results. In current query by singing/humming systems, when query results are returned, a user's preference or previous search history are often not considered. In proposed system, we use the information from a user's search history, as well as the properties of genres common to users with similar backgrounds, to estimate the genre or style, the current user may be interested in, based on a probability calculation. The accuracy from querying by singing/humming system is improved.

Keyword -- Query by humming/singing, Music Information Retrieval, query result re-ranking, edit distance, contour string generation.

I. INTRODUCTION

With the advent of new astonishing technologies, consumers can now use internet to play any type of music anywhere, anytime by searching it very easily. Automatic playlist creation, music recommendation and music search are related problems. Searching song by singing/humming section of it is the most natural way to search the song.

For searching the song, user uses a song's metadata (for example, song title, artist, or publication date, etc.), or the content of a music file (for example, melody). A user can easily and naturally search for a song through a voice recognition system. In order to perform a search, the user can call out the song title or artist. However, people often cannot recall the song title or the name of the artists, and only part of the melody is remembered. Currently there are many search engines on internet which allow user to search the songs using song related details like song title, artist name, movie name etc. But there are very few systems which do search using acoustic query as input. Many search engines based on acoustic input search song using same details of song mentioned earlier but by using voice recognition technology which gives more comfort to user by reducing his/her typing task. There are very few systems currently available which give freedom to user to search the song by singing/humming segment of song.

The main task for Query by Singing/Humming (QbSH) system is to do song searching based on query in the form of melody sang or hummed by the user. Songs often have similar melodies, and melodies that are sang or hummed are often inaccurate. Result is a large amount of query results returned and many of them may be wrong. If a large number of query returned then, it would bother users and make the system difficult to operate, especially when a user is unable to operate the device. Proposed system is focused on research on filtering unnecessary query results to reduce the operating procedures required by the user. This is achieved by use of probability calculations regarding various similarity factors of users' likings.

II. PROPOSED QBSH APPROACH

This section introduce the processing of query given by user, generation of contour string from MIDI file of the songs in database, match processing, re-ranking methods for better performance.

A. Query Transcription:

As illustrated in figure 1 we first take the input from user using microphone, this input is saved in the .wav file format. After recording we need to convert in such a form, so that the representation will become compatible of match processing. Firstly recorded file will be passed through low pass filter to reduce the level of noise in the recorded file and then it is converted into MIDI file [5]. Once this MIDI file is generated, this is analysed by our contour string generation methodology.

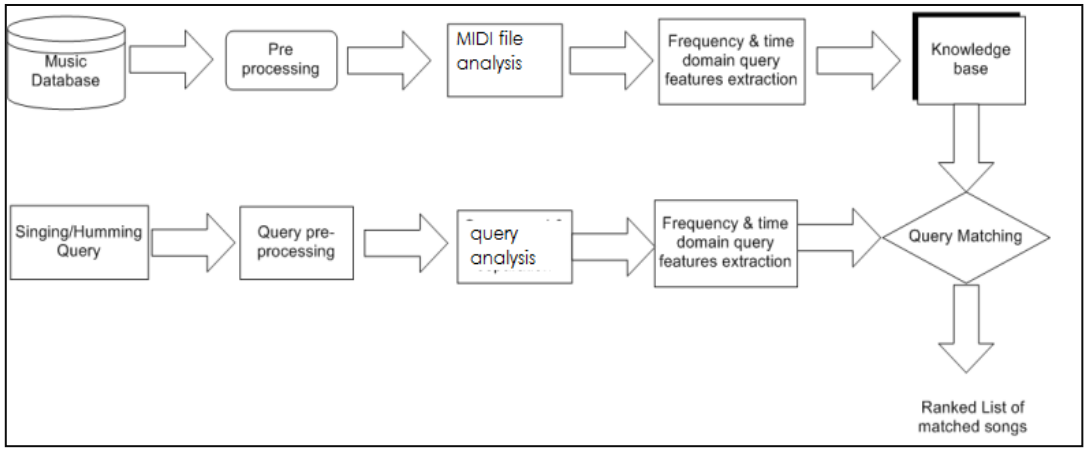

Figure 1 Architecture of the system.

B. Contour String Generation:

Our implementation takes a MIDI file containing a monophonic string of notes, and analyse the rhythm so that rhythmic contour string can be generated. 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. Each beat in the MIDI form is represented as a triplet (v, u, d), where v is the pitch scale, u is onset time and d is offset time. Then 'v' is represented in the form of pitch scale (v_1, $v_{2,\ldots}$,v_n) based on which contour string is generated. Contour string will be as shown in table:
For song "Twinkle, Twinkle, Little Star" Following contour string is generated:

Table I An example of a pitch string and its associated contour string

Pitch String	67 67 64 65 62 62 60 62 64 65 67
Contour String	0 -3 1 -3 0 -2 2 2 1 2

C. Match Processing:

Once we get both user query and song in database in the contour string form, we can readily compare user query with each song stored in database. For query matching dynamic programming is the best way[4]. For comparison purpose, user query contour string is denoted as p(p_1, p_2,.., p_m) and database song contour string is represented as q(q_1, q_2,… , q_m). Edit distance is Calculated using these two strings [1]. Edit distance (ED) is a common distance metric to measure the similarity between two contour strings. On one hand our QBSH system employs melody strings to represent the input query and entries in database, on the other hand in order to balance user singing or errors in humming, the relative pitch can be better to represent the coarse melody contour. Therefore the ED method is suitable to calculate similarity between transcribed query and indexed melodies with relative pitch. The edit distance between two compared sequences can be defined as the minimal transitional number of note inclusion, deletion and replacement which are necessary operations to transform source sequence into target sequence. The selection of appropriate cost function, make the ED to take user errors into consideration. Inclusion cost covers extra notes hummed by user, while the deletion cost accounts for notes skipped by user. To avoid a song with two or more segments matching a query, and all of these segments becoming the search results, for every song, we only consider the segment with the smallest edit distance to be the determining factor used to decide if this segment can be a search result or not. For the search process, songs with small edit distances are considered to be the potential results. They are called 'k' candidates. These k candidates are used for re-ranking purpose, so that optimized results are generated by QBSH system. This re-ranking makes our QBSH different from other QBSH systems which do not consider user history for result generation.

D. Re-ranking of the result:

How the re-ranking takes place for optimized result is introduced by this section. There are three sections as Ranking based on similar queries, Ranking based on user preference[2], Ranking based on similar user records.

1) Ranking based on Similar Users' Records :

If a query is performed byan user for a song, then it does not matter that how many times that song has been searched, the segment sang/ hummed by the user often shows similar signals. If that user is searching for a song that has been searched earlier, then to avoid repeating of the time spent on transferring and detecting the similar song, here we record every query made by the user and the corresponding results that are need to be corrected by the user.

2) Ranking based on User's Preference:

During the first search stage, the query is compared with the previous search results by us. In the ranking process that is followed, we consider the preferences of user. The user often searches a song in which they are interested. For

example, if a user likes Rock music, then their probability of searching the Rock song is higher than that of the classical song. According to this hypothesis in the second searching stage, we will be also comparing the query with previous history and we also consider the genre of music files that were already searched by the user previously.

3) Ranking based on Similar Users' Records:

The search results ranking based on user preference is affected by the number of queries performed by individuals. We can also say that, if only a few queries are performed, then it is not easy to estimate a user's music preference. This is called as the cold-start problem. But to resolve this, we have designed the collaborative method for ranking the songs in candidates. We have determined that different age groups have distinct music preferences from our observations of the music behaviour of average users. Also the different education levels affect the music types that people used to listen. We can consider an example that elders in southern Taiwan prefer the Japanese style songs in the native Taiwanese language, but the young people in northern Taiwan like to search for the Chinese Pop songs. While designing our collaborative method, the basic information collected from each user is used and preference analysis is performed on each attribute. Also the method called the naïve Bayesian prediction method is then used to estimate the user preference for the music genre.

III. USER INTERFACE

QBsH should have a web-enabled user interface too[3]. It is possible to upload a previously recorded audio input file based on the currently available technology, and also to do the required query signal processing (either on the server side or client side) and use the finally generated text string to search an indexed database of songs on the server side. At last, the first best matched songs are returned by means of links to the corresponding audio soundtracks as it is shown in the sample output page.

For the implementation of the desired user interface, http response writing, file upload, we could have used either JSP or Java Servlets running on the http server. Due to their superior performance, ability to handle multiple requests effectively, portability for the code and good security, the servlets got chosen. Java Servlets can be run on any Java enabled server supporting servlets. The server was installed and run from a Windows7 platform. The client-side operations are as follows: recording of the query to a standard audio format; and query file upload process. The server-side operations are: reading of the uploaded file at the server; query signal processing of the uploaded file; displaying the contour of pitch; searching for the indexed database; printing the ranked matches on the client's page.

IV. CONCLUSION

The query by singing/humming system using the user's singing/humming, not the text input, which attracts the attention. QBSH system also provides a simple and natural way of searching a song. Person does not need to remember the title of song or the artist of the song to perform a search. In our project, we use previous search history of the user as well users from the similar background to perform probability calculations, this method improves the rate of success. In our study we combine the probability calculation with the query by singing/humming. We used the users search history to perform a ranking calculation. The search system id hindered by cold start effect, when user perform a little search or no search in the past, it will gives the less accurate result. In this system, it takes long time to calculate the edit distance between the each string. This process causes significant delays while returning the result, user may discontinue the use of this system because of the delay. All our experiments are currently performed in quite environment. In real world it is often that all the devices will be operated in noisy environment. In future study, we will study how to reduce the impact of cold start effect on the system and also study the filtering of noise to reduce the errors during the counter string conversion.

REFERENCES

[1] Ning-Han Liu "Effective Results Ranking for Mobile Query by Singing/Humming Using a Hybrid Recommendation Mechanism", IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 16, NO. 5, AUGUST 2014

[2] Siwei Lai ,Yang Liu ,HuxiangGu ,LihengXu ,Kang Liu ,Shiming Xiang ,Jun Zhao"Hybrid Recommendation Models for Binary User Preference Prediction Problem"Workshop and Conference Proceedings 18:137{151, 2012 Proceedings of KDD-Cup 2011 competition}

[3] Chai-Jong Song, Hochong Park, Chang-Mo Yang, Sei-Jin Jang, and Seok-Pil Lee, "Implementation of a Practical Query-by-Singing/Humming (QbSH) System and Its Commercial Applications" IEEE Transactions on Consumer Electronics, Vol. 59, No. 2, May 2013

[4] Jyh-Shing Roger Jang, Ming-Yang Gao, "A Query-by-Singing System based on Dynamic Programming"

