Musical Search Engine With An Android Mobile

K. Vamshi Krishna
Embedded Systems Department
Aurora's Technological & Research Institute
Hyderabad, India

M. Pavani
Assoc. Professor in E.C.E
Aurora's Technological & Research Institute
Hyderabad, India

Abstract: Music information retrieval is a field of rapidly growing commercial interest. A Query By Humming system allows the user to find a song by humming art of the tune. No musical training is needed. Previous query by humming systems do not provide satisfactory results for various reasons. This paper describes, a query by humming based musical search engine, working on Android Platform like Android Mobile by accessing the web.

I. INTRODUCTION

Digital representations of music are becoming common for the storage and transfer of music over Internet. Many digital music archives are now available, making the content based retrieval of music a potentially powerful technology. The recent MPEG-7 audio standardization activity [1] seeks to develop tools for the description and intelligent searching of audio content. But in this system we collected a 10 of the most popular English songs by manual entry. These songs are segmented to some form of words. A Musical Search Engine based on acoustic querying would allow a user to hum or sing a short word of a song into a Android Mobile and then search and retrieve the song from the database.

II. APPLICATION ARCHITECTURE

Android is an open source operating system designed for mobile devices. It comes with a Java SDK and a valuable set of libraries to control on-board sensors and access web services. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It is built to be truly open.
"Android is built on the Open Linux kernel. Furthermore it utilizes a custom virtual machine that has been designed to optimize memory and hardware resources in mobile environment".

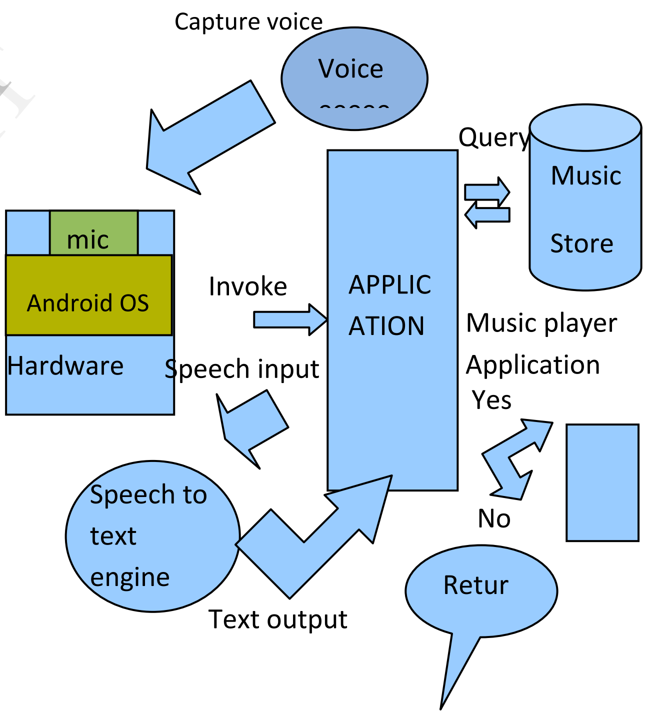

Figure 1. Architecture of Musical Search Engine.

Figure 1 shows the fundamental blocks of a basic query-by-hum based Musical Search Engine. The musical database is essentially an indexed set of some songs. The acoustic query, which is typically a few notes hummed or sung by the user, is processed to detect its approximate song line. The database is searches to find those songs that best match the query.

When the user is to hum/sung the song line , the user voice is processed of speech o text , the users voice is displayed on the mobile in the form of text. The Google has invoke the speech.

III. PROCESSING THE QUERY

A typical query by humming system includes three components:

- User humming: the input hum-query
- A database of music
- An index into the database for efficient retrieval of the hum-query

From the previous section we see that reliable note segmentation is a critical aspect of query processing. In order to simplify note segmentation, we currently require that the query be sung using a song tune instead of syllable.

In this, search engine we collected over 10 English songs by entering manually as a form of text. Upon these songs, the user can sung/hum any song of the word first it displayed on the mobile using speech recognition process.

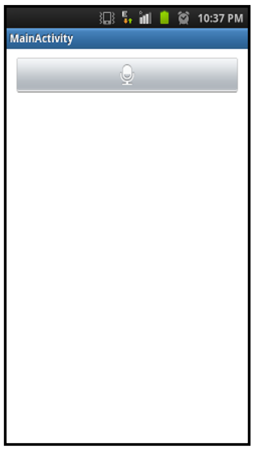

Figure 2: Musical search for Android Mobile.

In our interface we create an application by using java program, then loaded onto an Android Mobile.

Figure 1 shows the main activity slide in mobile. First the user press the music icon displayed on mobile.

Then it processed and appears like below in figure3.

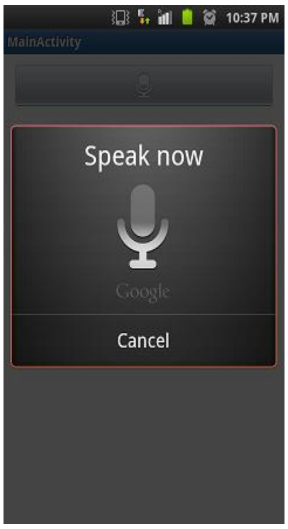

Figure 3. Musical search for Android Mobile.

Then the user can sung/hum the song (listed on the database) of first/any word during particular song. Then it is processed to detect its song line. The database is searched to find those songs the best match the query. The system returns a ranked set of matching melodies, which can be used to retrieve the desired original soundtrack.

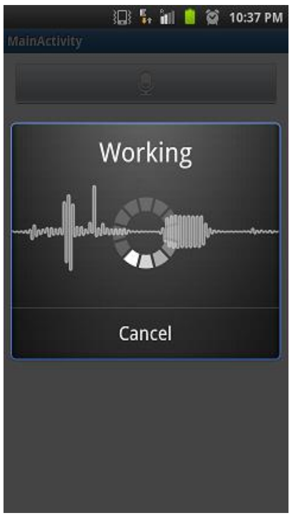

Figure 3. Musical search for Android Mobile.

IV. SPEECH RECOGNIGATION

ASR is short for Automatic Speech Recognition, which is also called Speech Recognition. It is generally performed by computer to convert audio waveform of speech to a sequence of readable text in real time.

Automatic speech recognition (ASR) can be defined as the independent, computer-driven transcription of spoken language into readable text in real time (Stuck less, 1994). In a nutshell, ASR is technology that allows a computer to identify the words that a person speaks into a microphone or telephone and convert it to written text.

Having a machine to understand fluently spoken speech has driven speech research for more than 50 years. Although ASR technology is not yet at the point where machines understand all speech, in any acoustic environment, or by any person, it is used on a day-to-day basis in a number of applications and services.

The ultimate goal of ASR research is to allow a computer to recognize in real-time, with 100% accuracy, all words that are intelligibly spoken by any person, independent of vocabulary size, noise, speaker characteristics or accent. Today, if the system is trained to learn an individual speaker's voice, then much larger vocabularies are possible and accuracy can be greater than 90%. Commercially available ASR systems usually require only a short period of speaker training and may successfully capture continuous speech with a large vocabulary at normal pace with a very high accuracy.

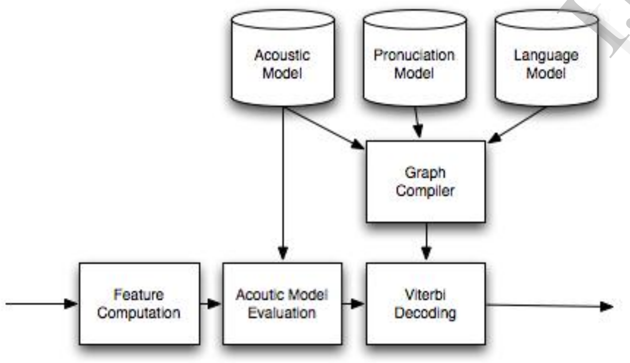

Figure 5: Basic block diagram of a speech recognizer.

We give an overview of Musical Search by Voice and our efforts to make speech input on mobile devices truly ubiquitous.

The explosion in recent years of mobile devices, especially web-enabled Smartphone's, has resulted in new user expectations and needs. Voice - a step toward our long term vision of ubiquitous access. While the integration of speech input into Musical search is a significant step toward more ubiquitous access, it posed many problems in terms of the performance of core speech technologies and the design of effective user interfaces.

It is an automated system that uses speech recognition and web search to help people find and listen music. Initially this system followed the well known model of first prompting the user for the songs followed by the desired listing of songs.

Mobile web search is a rapidly growing area of interest. Internet-enabled Smartphone's account for an increasing share of the mobile devices sold throughout the world, and most models offer a web browsing experience that rivals desktop computers in display quality. Users are increasingly turning to their mobile devices when doing web searches, driving efforts to enhance the usability of web search on these devices.

The goal of Musical search by Voice is to recognize any spoken search query. Hinting at the great diversity of inputs we must accommodate. Which is very domain-dependent?

Musical search by Voice must be capable of handling anything that Google search can handle. This makes it a considerably more challenging recognition problem, because the vocabulary and complexity of the queries is so large

Figure 5depicts the basic system architecture of the recognizer behind Musical search by Voice. For each key area of acoustic modeling and language modeling we will describe some of the challenges we faced as well as some of the solutions we have developed to address those unique challenges.

V. THE USER INTERFACE

According to the official introduction of Google, Android is a software platform for mobile devices such as telephone and tablet, which includes an operating system (OS), middleware and basic applications (Android, n. d.). Users can install the powerful software in accordance with their needs on Android system to extend the functions of the device.

Android platform, which consists of four major layers:

- Applications layer,
- Application Framework layer,
- Libraries and Android Runtime layer and
- Linux kernel layer.

Each layer is a collection of some program components, which provides the necessary services to its upper layer.

- The top layer of Android architecture is Applications layer. It should be particular pointed out that the applications installed in device can be run concurrently user can browse the website and listen to music in the meantime. Because all applications are developed based on the Java programming language,

- The application framework includes a large number of API classes and packages written in Java language to provide the direct interaction with the applications layer. Its purpose is to provide an effective and dependable way to "manage graphical user interfaces, access resources and content, receive notifications, or to handle incoming calls".

- The layer under the application framework consists of two significant parts which are called Libraries and Android Runtime. The first part libraries are all written in C and C++ language. Through being called by java interface, the libraries enable devices to process various data types and provide the core system features to applications (Maiaet al., 2010, p.4).

The libraries include some important components: libc is the most important component which provides standard C and C++ libraries; media framework provides a variety of media codecs to record and playback of different audio and video formats.

- Linux Kernel is located in the bottom layer of the Android architecture. It relies on Linux version 2.6 which was modified by Google in order to much more adapt to mobile devices.

The main function of this layer is to provide an abstraction layer between the hardware and above layers. The layers above the Linux Kernel do not
need to know how the hardware really works, they just need to send commands using API supported by the Linux Kernel, and then the Linux Kernel will actually interact with the hardware through providing the core system services. These most fundamental system services consist of memory management, process management, networking, security settings, driver model, etc (Android-App-Market, n. d.).

VI. CONCLUSION

Of immediate importance is increasing the number of songs in the database. This work is underway, and it is expected that a convincing demo on a realistic database will be presented at the conference. This is a step to further improve many applications like this. This can done in a local language also, the work is underway.

Database Songs	**10**
Queries	**50**
Mismatch	**1**
Conflicts	**1**
Success rate	**95%**

REFERENCES

[1] MPEG-7, http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm l
[2] M. Anand Raju, Preeti Rao, "Building a melody retrieval system", Proc.NCC, Mumbai, Jan 2002
[3] M. Anand Raju, Bharat Sundaram and Preeti Rao, "TANSEN: A QUERY-BY-HUMMING BASED MUSIC RETRIEVAL SYSTEM", Proc.NCC, Madras, Jan 2003
[4] Ghias A, Logan J, Chamberlin D, Smith B.C, "Query By Humming", Proc. ACM Multimedia, San Francisco, 1995
[5] McNab.R.J, Smith.L.A, Witten.I.H, Henderson.C.L, Cunningham.S.J, " Towards the Digital Music Library: Tune retrieval from acoustic input", Proc. ACM Digital Libraries, 1996.
[6] Gales, M. and Young, S. (2007) *The Application of Hidden Markov Models in Speech Recognition*, Hanover: now publishers Inc.
[7] Holmes, J. and Holmes W. (2001) *Speech Synthesis and Recognition. 3rd ed*. New York: Taylor & Francis.
[8] Jelinek, F. (1997) *Statistical Methods for Speech Recognition*, Massachusetts: The MIT Press.
[9] Green, P., Alexandersson, J., Hain, T., Moore, R. K. (n. d.) *Personal Adaptive Listeners – a Challenge for Speech Technology and Cognitive Systems.* [Unfinished work] (Personal communication with Prof. Phil Green, 2 August 2012).
[10] Christensen, h., Cunningham, S., Fox, c., Green, P., Hain, T., A comparative study of adaptive, automatic recognition of disordered speech. [Online] Natural Speech Technology. Available at: http://www.natural-speech-technology.org/node/149 [Accessed 2 August 2012].
[11] Gales, M. and Young, S. (2007), "The Application of Hidden Markov Models in Speech Recognition", Foundations and Trends in Signal Processing, Vol. 1, No. 3, pp. 197-206.
[12] Android, n. d. Android developer. [Online] Available at: http://developer.android.com/about/versions/index.html
[13] Android-App-Market, n. d. Android Architecture. [online] Available at: http://www.android-app-market.com/android-architecture.html [Accessed 10 August 2012].
[14] Android fragment, n. d. Android guide. [online] Available at: http://developer.android.com/guide/components/fragments.html

[15] .C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A general and e_cient weighted _nite-state transducer library. Lecture Notes in Computer Science, 4783:11, 2007.

[16] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope. Deploying GOOG-411: Early lessons in data, measurement, and test- ing. In Proceedings of ICASSP, pages 5260{5263, 2008.

[17] MJF Gales. Semi-tied full-covariance matrices for hidden Markov models. 1997.

[18] B. Harb, C. Chelba, J. Dean, and G. Ghemawhat. Back-o_ Language Model Compression. 2009.

[19] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. Journal of the Acoustical Society of America, 87(4):1738{1752, 1990.

[20] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. In CHI, pages 701{709, 2006.

[21] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In IEEE Transaction on Acoustics, Speech and Signal Processing, volume 35, pages 400{01, March 1987.

[22] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. Boosted MMI for model and feature-space discriminative training. In Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 2008.

IJERT