Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 267–273

An Approach for Developing a Mobile Accessed Music Search Integration Platform

Marina Purgina
Institute of Computing and Control
St. Petersburg State
Polytechnical University
St. Petersburg, Russia, 194021
Email: mapurgina@gmail.com

Andrey Kuznetsov
Institute of Computing and Control
St. Petersburg State
Polytechnical University
St. Petersburg, Russia, 194021
Email: andrei.n.kuznetsov@gmail.com

Evgeny Pyshkin
Institute of Computing and Control
St. Petersburg State
Polytechnical University
St. Petersburg, Russia, 194021
Email: pyshkin@icc.spbstu.ru

***Abstract*—We introduce the architecture and the data model of the software for integrated access to music searching web services. We illustrate our approach by developing a mobile accessed application which allows users of Android running touch screen devices accessing several music searchers including *Musipedia*, *Music Ngram Viewer*, and *FolkTuneFinder*. The application supports various styles of music input query. We pay special attention to query style transformation aimed to fit well the requirements of the supported searching services. By examples of using developed tools we show how they are helpful while discovering citations and similarity in music compositions.**

I. Introduction

A VARIETY of multimedia resources constitutes considerable part of the present-day Web information content. The searching services usually provide special features to deal with different types of media such as books, maps, images, audio and video recordings, software, etc. Together with general-purpose searching systems, there are solutions using specialized interfaces adopted to the subject domains. Truly, quality of a searching service depends both on the efficiency of algorithms it relies on, and on user interface facilities. As shown in our previous work, such interfaces include special syntax forms, user query visualization facilities, interactive assisting tools, components for non-textual query input, interactive and "clickable" concept clouds, and so on [1]. Depending on searching tasks, specialized user interfaces may support different kinds of input like mathematical equations or chemical changes, geographic maps, XML-based resource descriptions, software source code fragments, editable graphs, etc.

In text searching such aspects as morphological and synonymic variations, malapropisms, spelling errors, and time dependency condition particular difficulties of a searching process. In the music searching domain there are specific complications like tonality changes, omitted or incorrectly played notes or intervals, time and rhythmic errors. Thus, although there are eventual similarities between text and music information retrieval, they differ significantly [2].

In our previous work (see [3]) we analyzed and developed an improved EMD algorithm used to compare single voice and polyphonic music fragments represented in symbolic form.

Human ability to recognize music is strongly interrelated to listener's experience which may be considered itself to be a product of music intelligent perception [4], [5]. Recently (see [6]) we also analyzed internal models of music representation (with most attention to a function-based representation) being the foundation of various algorithms for melody extraction, main voice recognition, authorship attribution, etc. Music processing algorithms use the previous user experience implicitly. As examples, we could cite the *Skyline* melody extraction algorithm [7] based on the empirical principle that the melody is often in the upper voice, or *Melody Lines* algorithms based on the idea of grouping notes with closer pitches [8].

The remaining text of the article is organized as follows. In section II we review music searching systems and approaches of the day. We also introduce our experience in the domain of human centric computing and refer to some recent related works. In section III we describe music query input styles and analyze possible transformations of music input forms so as to fit the requirements of searching services. Section IV contains the description of the developed Android application architecture. We show how it works and make an attempt to analyze the searching output from the point of view of a musicologist.

II. Background and Related Works

Apart from searching media by metadata descriptions (like author, title, genre, or production year information), main scenarios of music searching are the following:

1) Searching music information by existing audio fragment considered as an input.
2) Searching music by the written note score.
3) Searching compositions by human remembrance represented in a form of singed, hummed, tapped or anyhow else defined melody or rhythm fragment.

Searching by given audio fragments is supported by many specialized search engines such as *Audiotag*, *Tunatic* or *Shazam*[9]. As a rule, it is implemented on the basis of so called audio fingerprinting technique. The idea of this approach is to convert an audio fragment of fixed length to a low-dimensional vector by extracting certain spectral features from the input signal. Then this vector (being a kind of audio spectral fingerprint) is compared to fingerprints stored in some database [10], [11].

978-1-4673-4471-5/$25.00

TABLE I
ACCESSING MUSIC SEARCHING WEB SERVICES

Name	*Access*				
	GUI	*Micro*	*Text*	*Soft*	*Web*
Audiotag	+	–	–	–	+
Tunatic	+	+	–	–	–
Shazam	+	+	Partially	–	+
Midomi	+	+	–	–	+
Musipedia	+	+	Partially	SOAP	+
Ritmoteka	+	–	–	–	+
Songtapper	+	–	–	–	+
Music Ngram Viewer	+	–	+	REST/JSON	+
FolkTuneFinder	+	–	+	REST/JSON	+

The second scenario *Searching by note score* implicates two possibilities. The first one is to look for a given music fragment in the note sheet databases (this is beyond the scope of this paper). The main difficulty demanding using special music oriented algorithms is that the note score may contain user errors. The second possibility is to convert the melody into one of forms discussed hereafter so as to use existing searching facilities.

Finally, in the case of *Searching by human remembrance*, a system deals with main voice, rhythm, melody contour or interval sequence which is usually not perfectly defined. Hence, it's impossible to search directly within the binary contents of audio resources: we have no faithful audio fragment.

Thus, there are numerous ways to provide music input for a searching system. The extensive description of input styles used by music search engines may be found in [12]. Presently there are many searching web services allowing customers using one of several possible styles to input a music query including such services like *Midomi*, *Musipedia*, *Ritmoteka*, *Songtapper*, *Music Ngram Viewer*, and *FolkTuneFinder*.

Table I represents possible ways to access different music web searching services and pays attention to the following facilities:

- **GUI** Graphical user interface
- **Micro** Using microphone
- **Text** Text mode
- **Soft** Access for software applications by SOAP or similar software interaction protocols
- **Web** Web interface

As you can see, nowadays many services are accessible via browsers since they support Web interface features. Another important issue is the possibility to access some services from inside the software applications by using open protocols. It gives the way to create tools which allow users not to be limited by only one service at a time. Due to such tools we are able to use integrally different features of different searchers, to access additional resources such as *youtube* clips, and to deal with additional search attributes like genres, styles, time periods, etc.

Particularly, *Musipedia* service uses SOAP protocol described in [13]. *FolkTuneFinder* and *Music Ngram Viewer*

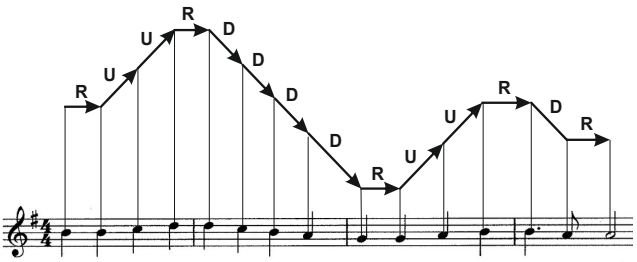

Fig. 1. Beethoven's "Ode to Joe" fragment represented in Parsons code

(both are also used in our work as target searching services) are based on the REST architectural style and their responses are wrapped in JSON format. Detailed description of the API usage rules and examples for *Music Ngram Viewer* service may be found in [14].

In respect to music inputs styles, existing tools support the following opportunities to define a music fragment:

- To sing or to hum the theme and to transfer the recording to the music search engine.
- To write notes by using one of known music notations directly (e.g. music score, Helmholtz or American pitch notation, MIDI notation, etc.).
- To tap the rhythm.
- To play the melody with the use of a virtual keyboard.
- To use MIDI-compatible instrument or it's software model.
- To enter Parsons code, or to set the melody contour by using "U, R, D" instructions [1] as shown in Figure 1.
- To define keywords or enter text query.

Table II lists some known music web searching services together with information about supported music query input styles, namely:

- **Audio** Audio fragment
- **Notes** Music score or pitch notation
- **Hum** Singing or humming
- **Rhythm** Tapping the rhythm
- **VKB** Virtual keyboard generating note sequence with rhythm
- **URD** Parsons code
- **MIDI** MIDI-piano
- **Text** Keywords or text query

III. MUSIC QUERY INPUT STYLES

As shown in the above section, we may define the music query by using different input styles. For a searching frame-

[1]Each pair of consecutive notes is coded as **U** ("sound goes Up") if the second note is higher than the first note, **R** ("Repeat") if the consecutive pitches are equal, and **D** ("Down") otherwise. Some systems use **S** ("the Same") instead of **R** to designate pitch repetition. Rhythm is completely ignored.

TABLE II
MUSIC SEARCHING WEB SERVICES INPUT STYLES

Name	*Web link*	*Input style*							
		Audio	*Notes*	*Hum*	*Rhythm*	*VKB*	*URD*	*MIDI*	*Text*
Audiotag	http://www.audiotag.info	+	–	–	–	–	–	–	–
Tunatic	http://www.wildbits.com/tunatic/	+	–	–	–	–	–	–	–
Shazam	http://www.shazam.com	+	–	–	–	–	–	–	+
Midomi	http://www.midomi.com	–	–	+	–	–	–	–	+
Musipedia	http://www.musipedia.org	–	+	+	+	+	+	+	+
Ritmoteka	http://www.ritmoteka.ru	–	–	–	+	–	–	–	–
Songtapper	http://www.bored.com/songtapper	–	–	–	+	–	–	–	–
Music Ngram Viewer	http://www.peachnote.com	–	–	–	–	+	–	–	–
FolkTuneFinder	http://www.folktunefinder.com	–	–	–	+	+	+	+	+

work, the important issue is not only featuring different input interfaces but transforming one query form to the another depending on searching service availability and it's communication schema.

Different input styles are useful since the user music qualification differs. Melody definition by using a virtual or real keyboard is one of the most exact ways to represent the query, since it accumulates most melody components. However it is not common that users are skilled enough to use the piano keyboard as well as to write adequate note score.

Contrariwise, tapping a rhythm seems to be relatively simple way to define music searching query. The problem is that the number of possible rhythm patterns is evidently less than the number of compositions. It means that even if we succeed to tap the rhythm correctly, we may apparently have a list of thousands titles in return [6].

For the melody contour schema it is possible to choose pitch and time quantization so that the pitch-time representation is equivalent to piano-roll notation [15].

Symbolic pitch notations may be useful if we are limited by only text user controls.

The development of mobile devices with touch screens affects strongly the usage aspects of music searching interfaces. Such devices make possible simulating many kinds of music instruments, although the virtual piano-style keyboard remains the most popular interface. MIDI standards supports transferring of the simulated playing signals between applications regardless the kind of simulated musical instrument. In contrast to the work [16] we don't propose a novel music search engine based on improved music input technologies. Our research is focused on creating middleware application which helps to communicate with existing searchers.

A. Input Styles Transformations

We represent relationships between music query input styles in form of an oriented graph shown in Figure 2. Every transition arc shows the possible transformation from one input style to another, together with indication which note information has to be extracted.

Since the virtual keyboard based query implicitly includes such note attributes as it's duration and it's pitch, there is no much difficulty to transform the keyboard input into the

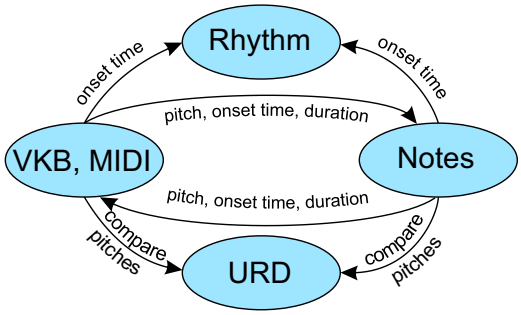

Fig. 2. Graph of music query transformations

rhythm or pitch notation. Next we are able to get the music contour from the sequence of sound pitches.

Clear that in such a way we restrict the user query, and therefore it seems we couldn't expect better searching results. However such transformations may have sense for at least two reasons:

- We attempt to emphasize the meaning of special melody attributes.
- We would like to try to connect a searching service which probably uses quite different music database (e.g. specialized on some music genre[2]) although it supports only restricted input methods (e.g. rhythm or pitch notation).

Regarding to the user interface issues, the ability to move from one input style to another renders possible to switch easily between different searching systems within the framework of one mobile or web application without re-entering the query.

B. Models We Use to Represent Queries

Despite the fact that in our earlier works we argued for the function based representation as one of the best way to describe music, this form seems to be too complicated to be employed directly at a user interface level. Usually user queries are relatively short (and it is true not only for the case of

[2]We turn our attention to the example of such a case in the following section of this paper.

music [17]), so we use sequence of *Note* objects to represent the searching query.

The attributes of a *Note* object are the following:

- *Note name* according to the American pitch notation
- Its *Octave number*
- Its *Onset time*
- Its *End time*

Values necessary for different input styles (such as a note duration or its MIDI value) may be computed on the base of above information. A series of onset time values may be used to generate a rhythm sequence.

IV. Introducing Android Application for Access to Music Searching Services

Nowadays, people are happy to use their mobile devices to access different searching services at any time from any place. They use different types of such devices which may have different input mechanisms like phone keys, *qwerty*-keyboards, touch screens, voice recognition devices, and so on. The variety of devices running on Android operating system is rapidly increasing during last years, so we decided to use Android platform for our music searching application.

For our implementation we selected some music searchers which may be accessed programmatically, particularly: *Musipedia*, *Music Ngram Viewer*, and *FolkTuneFinder*. For three searching systems we implemented four user query input styles:

- Note score editor supporting one voice definition
- Parsons code
- Rhythm tapping
- Piano style virtual keyboard with additional representation of American pitch notation[3]

A. Application Architecture

Despite general application construction ideas are common for various operating platforms, there is obvious specificity of the Android applications: the application architecture and its activities life cycle is governed by the operating system and the Android API. So the solution being discussed in this article isn't platform independent. However let us note that the Android application can serve as a model for implementing flexible human centric interface which is oriented to present-day style of using hardware and software facilities of various mobile devices.

That's why we skip the description of the specific Android implementation details like how to define resources, or how to support different screen resolutions and orientations. We don't discuss component layout control and internal data models either. Being limited by the paper scope, we only describe the principal Android application activities, their interaction and their connection to user interface components.

Figure 3 represents main components of our music searching helper application.

[3]In fact, it means that we support symbolic pitch notation input too.

The main activity *InputStyleSelection* provides the interface for input style choice. According to the selected input style the respective activity (*MelodyContour*, *MusicScore*, *VirtualPiano*, or *RhythmTapper*) opens and provides the corresponding input interface. The user input is stored as a list of *Note* objects used to construct the query as required. Classes *FolkTuneFinder*, *PeachNote* and *Musipedia* transfer the user input to one of supported search engines and filter their outputs.

B. Web Protocol Adapters

With respect to searching services' application interfaces mentioned in section II, the web information exchange protocol adapters have been implemented as Figure 4 illustrates.

The SOAP protocol is not recommended for mobile devices since it uses verbose XML format and may be considerably slower in comparison with other middleware technologies. Unfortunately it is the only way to communicate with the *Musipedia* system. In our case, the mentioned SOAP disadvantages shouldn't case concern since the exchange occurs relatively rarely, only when the respective button is pressed by a user, and there is small amount of information being transferred. We use *org.ksoap2* Java package [18] containing classes required for handling SOAP envelopes and literal XML content. To implement interaction with other searching services (based on the REST architecture and wrapping their responses in JSON format which is typically more compact in comparison with XML) we use *Google Gson* Java library [19]. It allows converting Java objects into their JSON representation as well as backward converting JSON strings to equivalent Java objects.

C. Usage Example

The application starts with a welcome screen for the preferred input style selection (Figure 5).

Then the respective activity starts as shown in Figure 6 representing the example of a virtual keyboard interface. As described earlier (see section III) the user actions are being stored in form of a note sequence with respect to the following related data:

- A *pitch* represented in the American pitch notation (note name and octave number
- The pitch *onset time*
- The pitch *end time*

Other properties may be computed depending on the requirements of a music searcher. Let us illustrate this by the input represented in form of a simplified timing chart (with respect to the note names rather than sound frequencies). The chart in Figure 7 represents some first notes of the well known Russian folk song "Birch Tree".

For the reason that *Musipedia* searcher requires a sequence of triplets containing an onset time, a MIDI pitch and its duration, the user input shown in Figure 7 is converted to the following query data:

0.0, 76, 0.54; 0.66, 76, 0.47; 1.21, 76, 1.43; 1.72, 76, 0.50; 2.41, 74, 0.98; 3.57, 72, 0.27; 3.89, 72, 0.52; 4.62, 71, 0.81; 5.57, 69, 0.75;

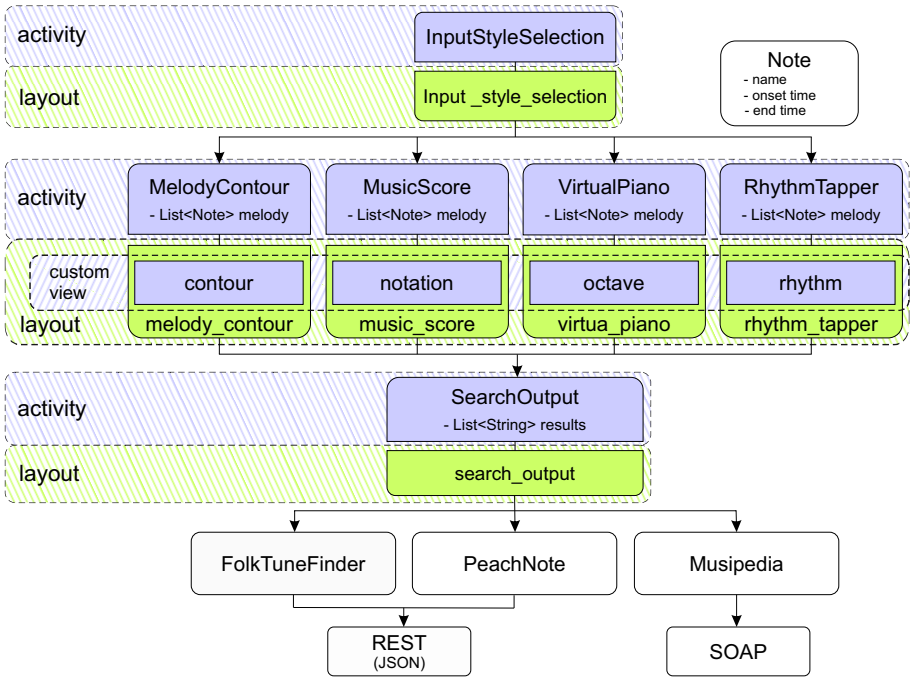

Fig. 3. Android music searching application architecture

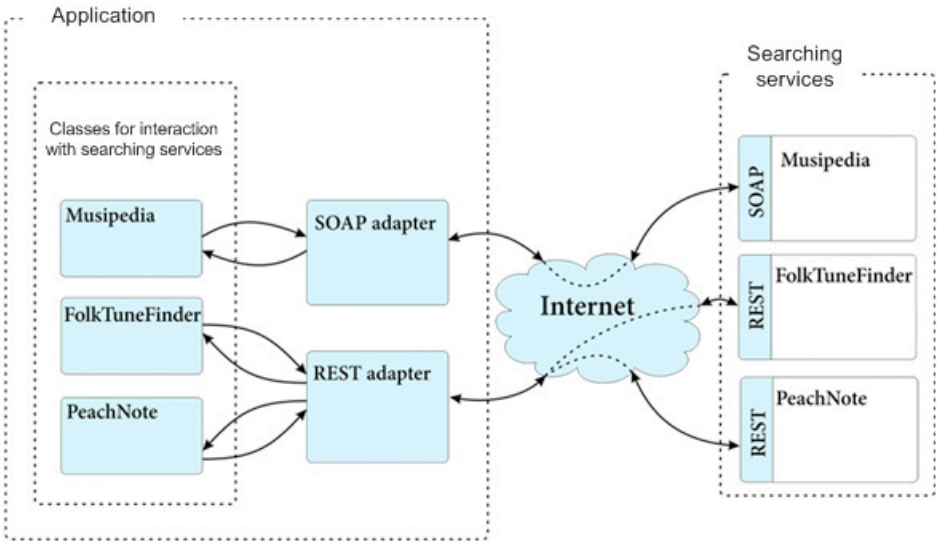

Fig. 4. Web protocol adapters

After this the respective information is included to the SOAP request which is subsequently sent to the *Musipedia* server. As a result, the searching system returns the list of retrieved compositions as shown in Figure 8 [4]. We see the confirmation of the known fact that this melody was used by Piotr Tchaikovsky in the 4th movement of his Symphony No.4 in F-moll (compare with the fragment of the symphony note score shown in Figure 9).

Using other searching engines may enhance searching results by taking into account other music genres. Let us take the *FolkTuneFinder* service which requires a sequence of MIDI pitches. Hence the user input is transformed into the sequence of MIDI pitches as follows:

76, 76, 76, 76, 74, 72, 72, 71, 69

[4]We selected Tchaikovsky's work, but as you can see, the similar theme may also be recognized in some other known compositions.

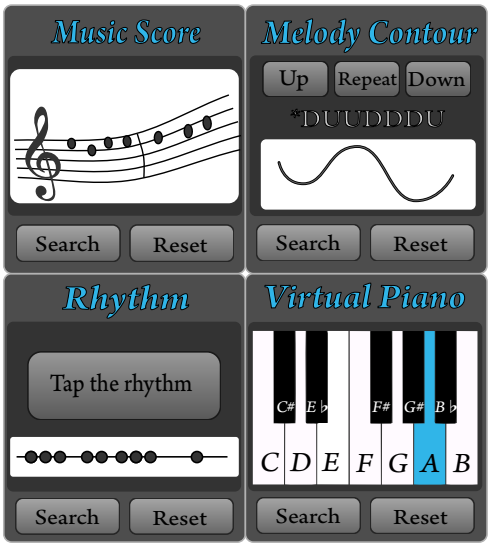

Fig. 5. Main activity: input style selection

Fig. 6. Virtual keyboard input

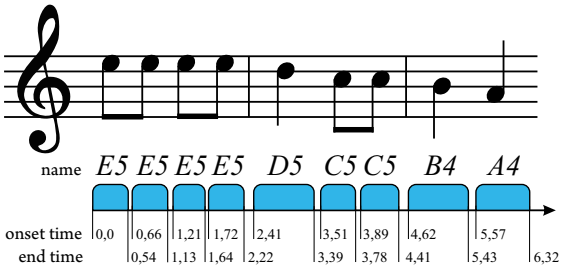

Fig. 7. Test melody: note score representation and timing chart

For the case of the melody contour defined with using Parsons code, the user input is the following *"RRRDDRDD"*.

Fig. 8. Result retrieved by *Musipedia* searcher: *Symphony No. 4*

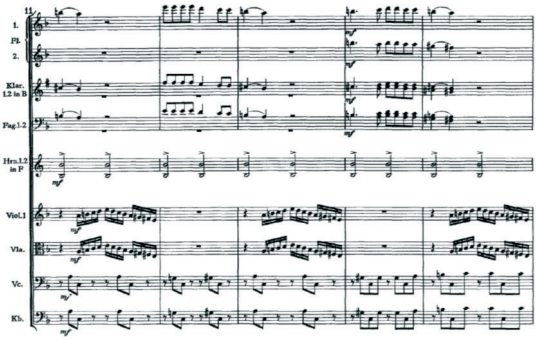

Fig. 9. Birch Tree song cited by Tchaikovsky in his 4th symphony

We implemented the interface component which allows constructing the URD-query by pushing buttons *Up*, *Down* and *Repeat* with synchronous demonstration of the respective graphical contour which is being generated automatically[5]. As you see in Figure 10 the resulting output also contains the "Birch Tree" among other compositions. Note that since the melody contour is a less exact input method (comparing to direct melody definition), it is normal that we don't have the desired melody in the very first lines.

The example we selected for the illustration shows well one important aspect of music searching process, although in a slightly simplified manner. When we discover the composition corresponding to the given request, we may expect obtaining even more information than simply a desired piece of music. Fast every Russian knows the "Birch Tree" song since the early childhood years. But only those who listen to the classical music discover this theme in one motive of Tchaikovsky's symphony. In contrast to this, western music lovers may listen this motive first just in the Tchaikovsky's work, and after a while recognize it as a citation of the Russian folk song. Isn't it a kind of process similar to a music perception in terms of musicology?

[5]We consider to investigate possibility to support a melody contour drawing interface in future implementations.

Fig. 10. Results retrieved by *FolkTuneFinder* searcher

V. Conclusion and Future Work

In the domain of human-centric computing much attention is paid to the facilitating user interface features in relation with a kind of data being processed. As a special type of information retrieval systems, music retrieval systems demand special ways to interact with users. They include not only traditional text or media based queries but specific forms of user input facilities such as note score representations, virtual or MIDI-compatible instruments, as well as composing queries based on melody humming or rhythm tapping which may contain errors of human interpretation. Such approaches may help to overcome limitations of fingerprinting techniques which require exact or nearly exact audio fragments to proceed with searching in the databases of stored music compositions. In our work we investigated styles of user inputs used in various music searching services and applications. We applied transformation rules of query conversion from one input style to another to a software tool communicating with programmatically accessible music searching services from mobile devices running on the Android operating system.

In the current implementation we supported only those music queries which are representable in symbolic form (e.g. note score, pitch notation, note sequences, or contour symbolic description). User interface facilities may be improved if we consider other ways to interact with the user having a touch screen device. It may include, for example, melody contour or rhythm drawing facilities. Even for the searching services that we used currently, there are input styles which are still not incorporated into the existing software prototype. We investigate possibilities to support interfaces for melody singing or humming. Actually we faced the problem to pass the audio query to the searching engines via existing data transfer protocols that we are allowed to use. Ways to extend the interface may also include a support for connected MIDI-compatible devices and text-based searching facilities aimed to explore music metadata information. The another interesting improvement which could fit well especially mobile equipment interfaces is to support music tagging as described for example in [20]. Hence the key idea is to connect different kinds of searching services with rich user input facilities so as to follow better the usage style of modern mobile devices.

Acknowledgment

The authors would like to thank Joe, creator of the *FolkTuneFinder* service, for the opportunity to access his software from our application and for his kind help in organizing software interface with the *FolkTuneFinder*.

References

[1] E. Pyshkin and A. Kuznetsov, "Approaches for web search user interfaces," *Journal of Convergence*, vol. 1, no. 1, 2010.
[2] Z. Mazur and K. Wiklak, "Music information retrieval on the internet," in *Advances in Multimedia and Network Information System Technologies*. Springer, 2010, pp. 229–243.
[3] A. Kuznetsov and E. Pyshkin, "Searching for music: from melodies in mind to the resources on the web," in *Proceedings of the 13th international conference on humans and computers*. University of Aizu Press, 2010, pp. 152–158.
[4] B. Snyder, *Music and Memory: An Introduction*. Cambridge, Mass. [u.a.]: MIT Press, 2000.
[5] D. Deutch, "Music perception," *Frontiers in Bioscience*, 2007.
[6] A. Kuznetsov and E. Pyshkin, "Function-based and circuit-based symbolic music representation, or back to Beethoven," in *Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments*. ACM, 2012, pp. 171–177.
[7] A. L. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," in *Proceedings of the sixth ACM international conference on Multimedia*, Bristol, United Kngdm, September 1998.
[8] C. Isikhan and G. Ozcan, "A survey of melody extraction techniques for music information retrieval," in *Proceedings of 4th Conference on Interdisciplinary Musicology (SIM'08)*, Thessaloniki, Greece, July 2008.
[9] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.
[10] W. Hatch, "A quick review of audio fingerprinting," McGill University, Tech. Rep., March 2003. [Online]. Available: http://www.music.mcgill.ca/~wes/docs/finger2.pdf
[11] P. Cano, T. Kalker, E. Batlle, and J. Haitsma, "A review of algorithms for audio fingerprinting," *The Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 271–284, 2005.
[12] A. Nanopoulos, D. Rafailidis, M. M. Ruxanda, and Y. Manolopoulos, "Music search engines: Specifications and challenges," *Information Processing & Management*, vol. 45, no. 3, pp. 392–396, 2009.
[13] "Musipedia SOAP interface." [Online]. Available: http://www.musipedia.org/soap_interface.html
[14] "Music ngram viewer API." [Online]. Available: http://www.peachnote.com/api.html
[15] D. Shasha and Y. Zhu, *High Performance Discovery in Time Series: Techniques and Case Studies*. Springer Verlag, New York, 2004.
[16] M. Wang, W. Mao, and H.-K. Goh, "Music search engine with virtual musical instruments playing interface," in *Advances in Multimedia Modeling*. Springer, 2013, pp. 502–504.
[17] V. Klyuev and Y. Haralambous, "A query expansion technique using the ewc semantic relatedness measure," *Informatica: An International Journal of Computing and Informatics*, vol. 35, no. 4, pp. 401–406, 2011.
[18] "Package org.ksoap2." [Online]. Available: http://ksoap2.sourceforge.net/doc/api/org/ksoap2/package-summary.html
[19] I. Singh, J. Leitch, and J. Wilson, "Gson user guide." [Online]. Available: https://sites.google.com/site/gson/gson-user-guide
[20] K. Bischoff, C. S. Firan, and R. Paiu, "Deriving music theme annotations from user tags," in *Proceedings of the WWW 2009*, 2009.