A Mobile-Based System for Context-Aware Music Recommendations

Börje F. Karlsson, Karla Okada, Tomaz Noleto

A Mobile-Based System for Context-Aware Music Recommendations

Börje F. Karlsson, Karla Okada, Tomaz Noleto

Nokia Institute of Technology (INdT), Av. Torquato Tapajós 7200,
Col. Terra Nova. 69093415. Manaus - AM, Brazil
{borje.karlsson, karla.gomes, tomaz.silva}@indt.org.br

Abstract. As mobile devices are always with users and music listening is a very personal and situational behaviour, contextual information could be used to greatly enhance music recommendations. However, making such use of context, while learning user profiles, is still a challenging problem. We present a system for collecting context and usage data from mobile devices, but targeted at recommending music according to learned user profiles and specific situations. The developed data flow system requires supporting both short enough response times and longer asynchronous reasoning on the collected data. Furthermore, the mobile phone acts not only as sensor, but is directly related to the effectiveness of the music service experience. Thus, this paper provides a description of our approach to the system and the initial results of a usability test of the mobile application and its backend system.

Keywords: Music, data flow, context-aware systems, recommender systems.

1 Introduction

Even though music listening is a highly personal and situational activity, and recommender systems for music are hardly a new idea, combining contextual data and user profiles in a music recommendation service is still an open problem. Depending on the application domain and the available data, at least certain contextual information can be useful for providing better recommendations [1, 11]. Quality of music recommendations is directly influenced by contextual data since people listen to diverse songs at different environments or when performing specific activities.

Context-aware services (CAS) are services enriched with information from their execution environment (in this case, users' mobile phones and situation). CAS are able to adapt to the current context to increase their usability and effectiveness [2]. Context-awareness and adaptation are key issues in mobile devices. Equipped with GPS receivers, RFID tags, microphones, cameras, etc., mobile devices could sense their changing environment and act intelligently based on their context.

In this paper, we present a system developed to explore the inclusion of contextual data into the music recommendation process, based on learning from collected data.

Recently, there has been much research on context-aware recommendation systems [4,5,12]. Rocha et al. [3] present the MoCA middleware, a context-aware middleware

for the development of collaborative applications. Carrillo et al. [8] propose context-aware user profiles, in which the profile definitions are associated with particular situations encountered by the users. Others have used case-based reasoning, collaborative filtering [4], or hybrid approaches [6,7]. Baltrunas et al. [12] present an interesting system for the kind of recommendations described here, but it is focused only in the "in car" context. Baltrunas and Amatriain [9] present a time-aware approach for music recommendation, creating time-based user models, which were then used to recommend music artists.

However, most of the work in the available literature assumes or proposes (beforehand) models for context representations or for user tastes (mostly based on ranking). Our approach tries to use implicit feedback and not to make assumptions about users or what constitutes proper *a priori* factor selection to represent contexts.

The described system uses user's behaviour and music metadata, i.e. besides usage data, the current system only processes music metadata for the recommendations; music structure is not taken into account nor is any kind of audio signal processing.

The remainder of this paper is organized as follow. The next section mentions the initial design goals of our music recommender system. In section 2 the system features are exposed (along with some reasoning on them): architecture, data gathering and processing, and user profile creation. Then in section 3, the anatomy of core components is explained, such as the contextual platform backend and the mobile recommender client including data provider, context-matching engine, and the actual contextual music application. Some preliminary tests and results are described in section 4; followed by concluding remarks in section 5.

1.1 Initial Design Goals

A usable interface is critical to any user-facing application. Increased demands on users' attention in dynamic environments can be addressed through interfaces that require less attention [10]. As such, one needs to understand how people interact with their surroundings. During the design and development effort described, a decision was made to start early with concept design/wireframes, not focusing on yet another music player, but on an application for understanding user context and recommending when and which songs to play; while, at the same time, getting the flow of data ready to handle reasoning on large amounts of data.

A second goal was focused on how to perform context mapping as to allow learning context-specific music preferences for a given user. The third main design goal was to build a test bed architecture and system capable of supporting future research activities on this kind of data and problem combination.

2 System Features

This section provides a high level description of the architecture and an overview of the most important steps in its data flow.

2.1 Architecture

The proposed system combines different environments to tackle the problem. The mobile device acts as sensor, collecting contextual information, as well as usage data collector. The data is then sent to a backend platform that also handles music specific metadata and has access to external data sources. Storage of large amounts of data and capabilities to perform reasoning on this data for the music recommendation problem are also responsibilities of this backend.

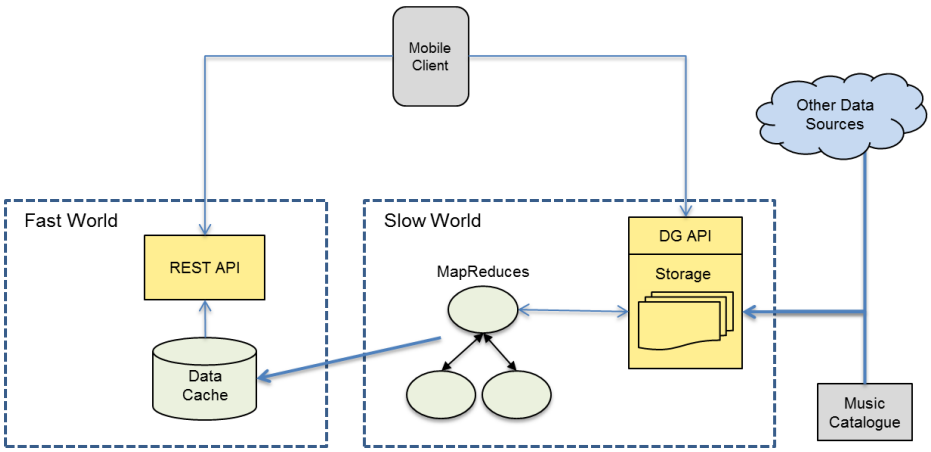

Fig. 1. System data flow. Mobile client sends contextual and usage data to the backend platform and receives recommendations from the REST API.

However, as the service focus is on providing adaptation to changes in user context, the system, as a whole, also needs to deal with constraints in how long request responses and recommendations can take to be generated. Also, even though most of the heavy lifting is done on the system backend, the mobile client needs to not only detect the current context, but to have enough data and flexibility to recommend the current songs (having some level of autonomy for cases when the backend is either not available or did not yet provide new data). The overview of the system data flow – with some of its constraints – is pictured in Fig. 1.

The backend is divided into two major sub-systems: a stateless layer (Fast World), focused on quick response times and exposing a REST API, which can be easily replicated to scale horizontally; and a MapReduce batch-like layer (Slow World), focused on processing of the gathered data and providing intermediary data structures – asynchronously – for the fast world cache. The mobile client application sub-modules are described in more detail in Section 3.2.

2.2 Data Flow

The device application collects three sets of data for processing: i) music metadata, from each track on the user's music library; ii) music listening habits of the user (which song was played, when, and for how long); and contextual data (i.e. data from sensors available on the device), that can later be reasoned over.

A set of miner sub-processes on the mobile device gather raw sensor data and - through the use of reasoning modules - try to infer more abstract context information (more details on section 3.2). The resulting aggregate data streams are then sent to the system backend through a data gathering API (designed to handle un-structured data).

The data coming from the device (as loosely-structured objects) is stored on the backend platform – a server cluster using a distributed file system environment. Scheduled MapReduce jobs are then responsible for processing chained recommendation flows. Periodically, the collected dataset, augmented by external data sources (e.g. a music catalogue from an international e-commerce music store, an artist influence graph, historic song purchase data, etc.), is processed to create/update user profiles, identify the most relevant recommendation contexts for each user, and generate new recommendations (playlist seeds, songs to buy, etc.).

The results are transformed into intermediary data-structures, which are exported to the fast world of the system backend and stored in a cache database; which is then available for queries by the services consumed by the mobile music application.

2.3 Profile Creation

User profiles play an important role as they serve as an individualization filter in a wide range of possible context adaptation parameters [11]. Profiling can be realized by adding a temporal dimension and using it for automatically improving the user profile with respect to context (thus adding a form of "learning").

In the described system, the first step when processing the collected data is the creation of a User Music Profile (UMP), which is used as base for the posterior steps. Basically, the UMP is composed of three main parts. The first represents the user preferences, where it is possible to identify the user's favorite artists and music genres; and the second contains playlist seeds for each relevant context identified for the user based on his/her preferences. As the system also generates recommendations for song purchasing and attending nearby concerts, related to the user's favorite artists, these recommendations are embedded in the third part of profile.

3 Anatomy of Core Components

This section describes the core components of the system, both of the contextual platform backend and the mobile recommender client.

3.1 Contextual Platform Backend

As described in the system architecture overview, the contextual platform backend (CPB) receives data about both context and usage and generates context-specific music recommendations. The generation of user music profile and a set of recommenders are the main processes running on the backend platform. The slow world recommenders run as often as possible (provided there is new data) to update the data on the fast world cache, allowing the users to benefit as much as possible

from their historic data. The fast world is then responsible for answering device queries by manipulating the cached data appropriately.

The CPB system performs seven main tasks in its slow world layer, shown in Fig. 2: creation of the User Music Profile, identification of relevant usage contexts, kickoff of appropriate recommendation generation, identification of related artists, recommendation of events for the user to attend, recommendations of songs for online purchase, and recommendation of seeds for context-specific playlists. The blue circles denote the phases of the system, the green rectangles denote its primary inputs, and the yellow circles represent its output (recommendations).

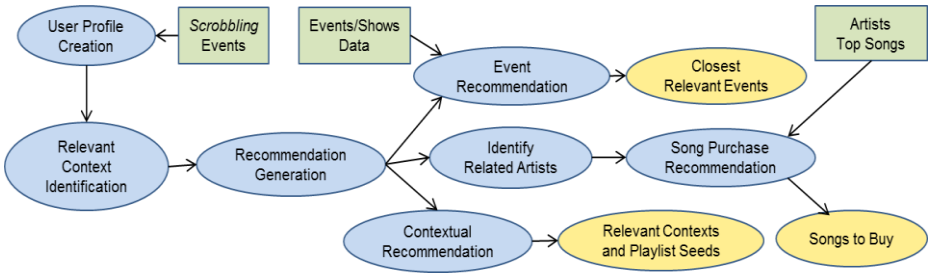

Fig. 2. Recommendation flow, running on CPB's slow world layer.

User Music Profile (UMP) - The main source of information for the creation of a UMP is the set of *scrobbling* events generated as users listen to music. S*crobbling* events contain metadata for the song being played and are generated when a user starts listening to a song. Subsequent event are also generated according to the same rules proposed and used by the Last.fm[1] service, wherein the track is considered to have been listened to when either 4 minutes or half of the song has been played.

Scrobbling events in our system are enriched because the player application also knows the context in which each song was played. Specifically, it tries to determine the action or activity and the environment the user is in based on the sensory data from the device (c.f. section 3.2). In addition, user location and event time are sent in each scrobble event. Thus, *scrobbling* events contain the following context attributes: a) music metadata (title, album name, artist, genre, and track length); b) user location; c) user activity (running, walking…); d) environment (restaurant, library…); e) time of the day. The set of attributes mentioned represents a context.

With this data the system is able to generate the profile for the user, a behaviour based profile, applying a statistical usage approach. The profile for the user tells what user has listened in each context. The basic profile consists of the total number of tracks listened and counts for each artist and genre of the listened tracks, both from tracks belonging to each context and for the full history of the user.

The initial version of the profiler is actually rather simple in determining the most common contexts for each user. The UMP process computes histograms for each context, which gives us a distribution of contexts based on the play count. Then from

[1] http://www.lastfm.com

this distribution contexts are taken from the top until the combined event count exceeds 30% of the total number of events.

Identification of Relevant Contexts - Some issues are detected at this point. First, time and location need to be discretized in order to be useful. When using time, we simply take the day of the week and time of the day as the relevant quantities. For week day we can use all days in the week or aggregate to weekdays and the weekend, giving nine discrete values. In the case of time the day is divided into four quarters.

The device is responsible for providing location information latitude, longitude pairs and simply binning them will not provide good results. Therefore the location data for each user is clustered to find out their key locations. The locations are assumed to be found if there are few (one to three) dense clusters with significant number of events. An algorithm that uses the k-means iteratively, while stripping away "unnecessary" points, is utilized to reduce the dataset.

At first, k-means is run for the whole dataset for the user. From the resulting clusters the points furthest away from the cluster centroid are removed. The same procedure is repeated until clusters are found such that points in each cluster are within 1 Km of the centroid and the points in all of the clusters comprise at least half of the full location data for that user.

After this all of the data to determine the relevant contexts for each user has discrete values and we can simply count the most common ones. However, it might not be feasible to use the full five dimensional (activity, environment, day, time, location) data to find the clusters as the data simply might be spread too thin. Thus, if not enough data is available we find the key contexts for the following cases while aggregating over the others: a) all five values with all days of week; b) all five values with day as weekday/weekend; c) only activity and environment; d) only day of week and time. From this we must take away possible duplicates; in these cases the more generic one is considered the more relevant one.

Recommenders - On the backend platform, several algorithms fed with the UMPs form the set of recommenders. For each relevant recommendation context identified in the previous step, a set of playlist seeds (recommended song IDs) is generated by using user preferences and historic data on the system.

From the preferences identified on the UMP, other recommendations are also generated: i) events recommendations; and ii) music purchase recommendations. Details of these recommendations processes are outside the scope of this paper.

3.2 Mobile Recommender Client

The mobile client includes primarily two processes: a daemon process, responsible for collecting context-related data, as well as music-related usage data; and the music application itself, the user-facing side of the system, responsible for interfacing with the user and playing the appropriate songs for each context. Fig. 3 shows the internal modules of these processes.

Data Provider - Contextual data retrieval is performed by the data provider daemon, which is composed of: i) a set of miners, each one responsible for gathering music data or data from a specific context attribute sub-module; ii) a serializer layer,

responsible for translating the mined data into intermediary objects, assembling the *scrobbling* events structure, and serializing the events; and iii) the distribution engine, responsible for interacting with the Data Gathering API on the backend system.

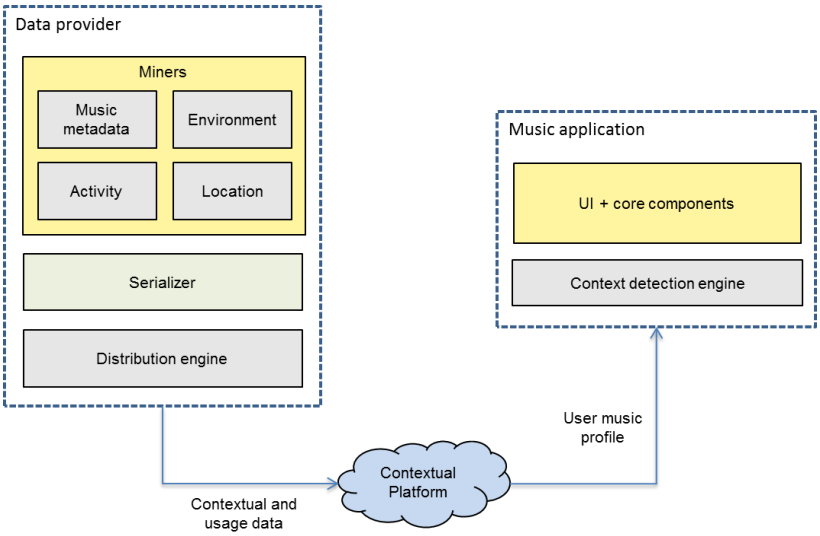

Fig. 3. Device-side processes.

Context attribute miners here can be seen, according to the semantic context interpretation and abstraction layers presented in [13], as belonging to the High-Level Context layer, where the lower level sensory information is semantically interpreted.

The activity miner classifies what the user is doing based on readings from the phone accelerometer, and its output classifications vary among walking, running, bicycling, etc. The environment miner tries to infer where the user is at a given moment by getting audio input samples and comparing them to a trained database of classified samples. Possible values include meeting lecture, office, bus, and others. The location miner provides (lat, long) coordinates, which can be more or less precise depending on the active source (GPS, Cell-ID), or other location-related items like Wi-Fi MAC addresses, or Bluetooth IDs, associated to the last know position.

Finally, the music metadata miner observes the user's local music catalogue – keeping the backend aware of eventual addition or removal of songs – and the mobile platform's underlying media framework for music playing events to be translated into *scrobbling* events. Metadata, context and *scrobbling* events are then serialized and sent via the proper API to the backend.

Context Matching Engine - As mentioned when describing the UMP, whenever the backend has new relevant profile data, the music recommender application fetches this data and caches it locally to use it as input for feeding context-specific structures.

In order to decide when and what types of song to play, the application needs to figure out what the user wants to listen in a given moment. This context detection engine uses the data in the user profile and from the sensor readings in the data provider, trying to find the best possible matches, therefore assembling a dynamic

playlist containing only songs that belong to that specific context. If the user context changes, the engine decides whether the current playlist is no longer suitable for the user and repopulates it with the new context's songs.

The context identification involves aggregating the current context attributes and comparing the resulting set with the different sets in the UMP. Currently the set with the most matches is the one selected as current appropriate context.

Contextual Music Application - The concept behind the application is that the UI shall allow the user to explore its local (to the phone) catalogue of songs and, from usage data, start to suggest context-appropriate songs for playing or purchasing (as well as other content-related data, e.g. nearby events from related artists/bands). Fig. 4 shows the two most important screens according to this concept. The **"What is new?"** screen is the main recommendation panel on the application and surrounds the currently playing song with recommendations for the current context: to play (songs from the user's local catalogue), to purchase (songs from artist similar to the ones the user listens in the detected contexts), and to augment (e.g. event suggestions).

The second screen shows one of the available panels when playing the song. It shows more purchase suggestions (on the list at the top), and the next recommendation on the created playlist for the current context (at the bottom).

As 3G services provide network access capabilities on the device, other side-scrollable panels provide more related data (as for example discography or band information about the artist being played), as well as services that are independent from our main system (e.g. what other close by users are listening to).

Fig. 4. Main screen and purchase recommendations.

4 Preliminary Tests and Results

While a complete user evaluation is beyond the scope of this paper, ongoing experiments suggest the described system produces interesting results, provided users have specific behaviours. A two-step trial was executed to evaluate both the usability of the concept and the actual contextual recommendations. At a first stage a small test lasting one week was performed with 10 users (5 male, 5 female, from age 18 to 32) that are active music listeners and listen to songs in a range of activities and places.

As this test focused on the comprehension of the recommender concept, a set of tasks had to be completed by the participants; which later were interviewed to gather their preferences, needs and opinions on possible improvements. This first experiment generated 3,704 *scrobbling* events and allowed us to track how users' initial to final perceptions improved. Some key points from the first trial were:

- All users seemed very interested in the recommendation of events;
- Six of the ten users were initially confused by the lack of direct control over which songs were played, but towards the end most were satisfied with the recommendation to play or purchase (7 out of 10; 4 mentioning that the recommendations exceeded their expectations in certain scenarios);
- Three users declared to be always curious to see what the system would suggest;

While nine out of the ten participants were not used to buying songs online, all considered the experience of discovering a new song interesting and attractive; and commented that they felt encouraged, by the recommendations, to buy new songs.

Based on this feedback, refinements were made to the UI and some minor changes were applied to the system data flow. A more extensive trial was initiated with 59 users in 4 different countries, which generated additional 22,467 *scrobbling* events.

As quality for activity/environment specific recommendations is subjective, the analysis of the collected data and participant opinions has not been concluded yet. So far only around 20% of the users were debriefed and, of these, most were neutral regarding the application user interface and more than half mentioned enjoying the recommendations. A full analysis of the trial results will be published appropriately.

5 Concluding Remarks

This paper described a CAS for music recommendation whose goals were to explore the problem space and to allow a good degree of quality to the user experience in receiving music recommendations. The system employs a number of techniques and approaches to both deal with the necessary data flow (in quantity and different data types) and to generate quality recommendations. The implementation of the described architecture for collecting contextual data, reasoning on it and generating situational recommendations provided many insights and opportunities for improvement.

The initial results suggest that the system can properly identify and recommend song per context. A more in depth analysis of the trial results we already have is our first objective. Some other immediate goals are: improvements to efficiency, optimization of battery usage on device, and improvements to the offline case.

However, context specific music recommenders are complex systems and much remains to be done and other features are just starting to be explored, such as relevance feedback and clustering. Combination of content-based and collaborative filtering to predict interests (similar to Yeung et al. [6] approach in the news domain) and other forms of social data (e.g. ongoing research concerning the extension of context profiling to groups of users [11]) is also on the agenda.

6 Acknowledgments

We'd like to thank Francimar Maciel and Maysa Martins for all the help in setting and running the trials. We would also like to thank Ingemar Larsson for being the tireless champion of this idea, and for making sure everything ran as smoothly as possible.

References

[1] Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowledge and Data Engineering, vol. 17 (6), pp. 734--749 (2005)

[2] Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing vol. 2 (4), pp. 263--277 (2007)

[3] da Rocha, R., Endler, M.: Context Management in Heterogeneous, Evolving Ubiquitous Environments. IEEE Distributed Systems Online, vol. 7 (4), IEEE Comp. Society (2006)

[4] Chen, A.: Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment. In: Strang, T., Linnhoff-Popien, C. (eds.) Location and Context Awareness 2005. LNCS, vol. 3479, pp. 75--81. Springer (2005)

[5] Woerndl, W., Brocco, M., Eigner, R.: Context-Aware Recommender Systems in Mobile Scenarios. Intl. Journal of Information Technology and Web Engineering, vol. 4 (2009)

[6] Yeung, K.F., Yang, Y., Ndzi, D.: Context-Aware News Recommender in Mobile Hybrid P2P Network. In: 2nd International Conference on Computational Intelligence, Communication Systems and Networks - CICSYN (2010)

[7] Said, A.: Identifying and Utilizing Contextual Data in Hybrid Recommender Systems. In: 4th ACM Conference on Recommender Systems - RecSys (2010)

[8] Carrillo-Ramos, A., Villanova-Oliver, M., Gensel, J., Martin, H.: Contextual User Profile for Adapting Information in Nomadic Environments. In: Personalized Access to Web Information Workshop (PAWI), LNCS, vol. 4832, Springer-Verlag, pp. 337--349 (2007)

[9] Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: Workshop on Context-Aware Recommender System - CARS (2009)

[10] Tarasewich, P.: Designing mobile commerce applications. Communications of the ACM - Mobile computing opportunities and challenges, vol. 46 (12), Dec. 2003, USA (2003)

[11] Do, T. M. T., Blom, J., Gatica-Perez, D.: Smartphone usage in the wild: a large-scale analysis of applications and context. In: 13th International Conference on Multimodal Interfaces - ICMI (2011)

[12] Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Luke, K.H., Schwaiger, R.: InCarMusic: Context-Aware Music Recommendations in a Car. In: 12th International Conf. on Electronic Commerce and Web Technologies - EC-Web (2011)

[13] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. In: Pervasive and Mobile Computing, vol. 6 (2), pp. 161-180 (2010)