Context-Aware Mobile Music Recommendation for Daily Activities

Xinxi Wang, David Rosenblum, Ye Wang
School of Computing, National University of Singapore
{wangxinxi,david,wangye}@comp.nus.edu.sg

ABSTRACT

Existing music recommendation systems rely on collaborative filtering or content-based technologies to satisfy users' long-term music playing needs. Given the popularity of mobile music devices with rich sensing and wireless communication capabilities, we present in this paper a novel approach to employ contextual information collected with mobile devices for satisfying users' short-term music playing needs. We present a probabilistic model to integrate contextual information with music content analysis to offer music recommendation for daily activities, and we present a prototype implementation of the model. Finally, we present evaluation results demonstrating good accuracy and usability of the model and prototype.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; H.5.5 [**Sound and Music Computing**]: [Modeling, Signal analysis, synthesis and processing]

General Terms

Algorithms, Design, Experimentation

Keywords

activity classification, context awareness, mobile computing, music recommendation, sensors

1. INTRODUCTION

Music recommendation systems help users find music from large music databases, and an effective system is one that consistently matches a user's preference. Most of the existing music recommendation systems that model users' long-term preferences provide an elegant solution to satisfying long-term music information needs [1]. However, according to some studies of the psychology and sociology of music, users' short-term needs are usually influenced by the users' *context*, such as their emotional states, activities, or external environment [2–4]. For instance, a user who is running generally will prefer loud, energizing music. Existing commercial music recommendation systems such as Last.fm and Pandora cannot satisfy these short-term needs very well. However, the advent of smart mobile phones with rich sensing capabilities makes real-time context information collection and exploitation a possibility [5–10]. Considerable attention has focused recently on *context-aware music recommender systems* (CAMRSs) in order to utilize contextual information and better satisfy users' short-term needs [11–28].

Existing CAMRSs have explored many kinds of context information, such as location [13, 19, 21, 24, 28], time [12, 19, 25, 26, 29, 30], emotional state [4, 12, 16, 18, 23, 27], physiological state [13, 17, 22, 25], running pace [11, 14, 17], weather [12, 25], and low-level activities [16]. To the best of our knowledge, none of the existing systems can recommend suitable music *explicitly* for *daily activities* such as working, sleeping, running, and studying. It is known that people prefer different music for different daily activities [2, 3]. But with current technology, people must create playlists manually for different activities and then switch to an appropriate playlist upon changing activities, which is time-consuming and inconvenient. A music system that can detect users' daily activities in real-time and play suitable music automatically thus could save time and effort.

Most existing collaborative filtering-based systems, content-based systems and CAMRSs require explicit user ratings or other manual annotations [1]. These systems cannot handle new users or new songs, because without annotations and ratings, these systems are not aware of anything about the particular user or song. This is the so-called *cold-start* problem [31]. However, as we demonstrate in this paper, with automated *music audio content analysis* (or, simply, music content analysis), it is possible to judge computationally whether or not a song is suitable for some daily activity. Moreover, with data from sensors on mobile phones such as acceleration, ambient noise, time of day, and so on, it is possible to infer automatically a user's current activity. Therefore, we expect that a system that *combines* activity inference with music content analysis can outperform existing systems when no rating or annotation exists, thus providing a solution to the cold-start problem.

Motivated by these observations, this paper presents a ubiquitous system built using off-the-shelf mobile phones that infers automatically a user's activity from low-level, real-time sensor data and then recommends songs matching

MM'12, October 29–November 2, 2012, Nara, Japan.

the inferred activity based on music content analysis. More specifically, we make the following contributions:

- *Automated activity classification:* We present the first system we are aware of that recommends songs explicitly for everyday user activities including *working*, *studying*, *running*, *sleeping*, *walking* and *shopping*. We present algorithms for classifying these contexts in real time from low-level data gathered from the sensors of users' mobile phones.
- *Automated music content analysis:* We present the results of a feasibility study demonstrating strong agreement among different people regarding songs that are suitable for particular daily activities. We then describe how we use music content analysis to train a statistical model for predicting the activities for which a song is suitable. This analysis can operate offline since the predictions they produce are independent of individual user activities or listening behaviors.
- *Solution to the cold-start problem:* We present an efficient probabilistic model for *Adaptive Context-Aware Content Filtering* (ACACF) that seamlessly unifies the activity classification and music content analysis results. This model can be updated *on-the-fly* for each user to adapt to their ongoing listening behavior.
- *Implementation and evaluation:* We present a prototype mobile application that implements all parts of the ACACF model except music content analysis entirely on a mobile phone, and we present evaluation results demonstrating its accuracy and usability.

This paper is organized as follows. Section 2 presents an overview of traditional music recommender systems. Section 3 formulates the probabilistic model used to do context-aware recommendation based on context inference and music content analysis. Section 4 describes the system design and implementation. Section 5 describes evaluations of our model and system. Section 6 discusses previous research that is most closely related to our own. Section 7 concludes the paper with a discussion of plans for future work.

2. BACKGROUND

Traditional music recommender systems can be classified according to three categories of recommendation methods: *collaborative filtering* (CF), *content-based* methods, and *hybrid* methods [1]. The main idea behind CF is that if user A and user B have similar music preferences, then songs liked by A but not yet considered by B will be recommended to B. CF-based systems suffer from the cold-start problem since they cannot recommend songs to new users whose preferences are unknown (the *new-user* problem) or recommend new songs to users (the *new-song* problem) [31]. In contrast to CF, content-based systems work as follows: If user A likes song S, then songs having content (i.e., musical features) similar to S will be recommended to A. Content-based systems help solve the new-song problem, but they still suffer from the new-user problem. Hybrid methods combine CF and content-based methods.

Traditional music recommender systems model only users' long-term preferences. CAMRSs, which take advantage of users' short-term context information, have been explored by some researchers. We next present our own CAMRS so that we may later discuss related work on CAMRSs.

3. UNIFIED PROBABILISTIC MODEL

In this section we present our Adaptive Context-Aware Content Filtering model, ACACF. The model uses a Bayesian framework to seamlessly integrate context-aware *activity classification* and *music content analysis*.

3.1 Problem Formulation

Let $\mathcal{S}$ be a set of songs and $\mathcal{C}$ a set of context categories.[1] For our model, the contexts are daily activities, with $\mathcal{C}$ ={*running*, *walking*, *sleeping*, *studying*, *working*, *shopping*}, but we can extend the model to other activities. A user is assumed to be in exactly one context category $c \in \mathcal{C}$ at any time. We also assume the user always carries his/her mobile phone, and that a sensor data stream can be recorded continuously from the phone. For our model, the sensor data includes time of day, accelerometer data, and audio from a microphone. The sensor data stream is divided into a sequence of frames, possibly with overlap between adjacent frames. For each frame, a vector $\mathbf{f}$ of *features* of the sensed data is extracted. The recommendation problem is then formulated as a two-step process: (1) infer the user's current context category $c \in \mathcal{C}$ from $\mathbf{f}$, and (2) find a song $s \in \mathcal{S}$ matching c the best. We call the first step *context inference* and the second step *music content analysis*.

3.2 Probability Models

Inferring a user's current context category c from the feature vector $\mathbf{f}$ is not an easy task. In our early experience we found it difficult sometimes to differentiate working and studying by a mobile phone; as sensed activities they appear to be similar, but they need to be differentiated because people have different music preferences when working versus studying. In order to capture such uncertainty, instead of obtaining exactly one context category from $\mathbf{f}$, we obtain a probability distribution $p(c_i|\mathbf{f})$ over all categories. For instance, if there is complete uncertainty about whether a user is working or studying, then we can assign the probability 0.5 to both working and studying. Using Bayes's rule, $p(c|\mathbf{f})$ can be as in Equation (1):[2]

$$p(c|\mathbf{f}) = \frac{p(\mathbf{f}|c)p(c)}{p(\mathbf{f})} \propto p(\mathbf{f}|c)p(c) \tag{1}$$

We call this part of our model the *sensor-context* model, and we elaborate it further in Section 3.4.

To model whether a song s is suitable for a context category c, we introduce a random variable $R \in \{0, 1\}$. $R = 1$ means s is suitable for c, and $R = 0$ otherwise. Then we use the probability $p(R = 1|c, s)$ to indicate the user satisfaction degree of song s when he/she is in context c. We call this part of our model the *music-context* model, and we elaborate it further in Section 3.3.

Combining $p(\mathbf{f}|c)p(c)$ with $p(R = 1|c, s)$, we obtain the joint probability shown in Equation (2):

$$p(c, \mathbf{f}, R, s) \propto p(\mathbf{f}|c)p(R|c, s)p(c) \tag{2}$$

[1]In the notation we present, bold letters represent vectors, calligraphic upper case letters represent sets, and random variables and their values are indicated by italicized upper-case and lower-case letters respectively.

[2]In this and subsequent formulas, we indicate proportional equivalents where normalizing constants can be omitted, thereby improving computation efficiency.

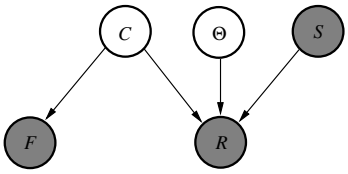


Figure 1: Graphical representation of the ACACF Model. Shaded and unshaded nodes represent observed and unobserved variables, respectively.

We assume that all songs share the same prior probability, so $p(s)$ can be omitted. The combined model can be represented by the graph depicted in Figure 1. The combined model is our ACACF model. Random variable Θ is a probability prior and will be explained in Section 3.3.1. The model is adaptive in that the component probabilities are updated dynamically as a result of evolving user behavior; the adaptive features of ACACF are presented in Sections 3.3.1 and 3.4.

With this model, the recommendation task is defined as follows: Given the feature vector $\mathbf{f}$ calculated from sensor data, find a song s that maximizes the user satisfaction $p(R = 1|s, \mathbf{f})$, which is calculated in Equation (3) as the sum of the joint probabilities for all possible context categories:

$$\begin{aligned} p(R = 1|s, \mathbf{f}) &\propto p(R = 1, s, \mathbf{f}) \\ &= \sum_{i=1}^{|\mathcal{C}|} p(R = 1, s, \mathbf{f}, c_i) \end{aligned} \tag{3}$$

To calculate the joint probabilities of Equation (2), we compute estimates for the probabilities of the music-context model and the sensor-context model, as explained in Sections 3.3 and 3.4, respectively.

3.3 Music-Context Model

3.3.1 Modeling and Adaptation

As described later in Section 5.2, we have found that users agree on the *features* of music they prefer when they are doing a particular activity. However, in general, different users like different songs. Thus, to provide a more personalized recommendation, ACACF incorporates implicit user feedback. For example, if a user listened to a song completely, the user probably likes the song; we call this *positive feedback*. If the user skipped a song after listening for only a few seconds, then the user probably dislikes the song; we call this *negative feedback*. Implicit feedback has been exploited by some researchers [32], and we integrate it seamlessly and efficiently in our own ACACF model.

To model implicit feedback, for each user we assign a *probability prior* (or simply a *prior*) $\Theta_{c,s} \sim beta(\theta_{c,s}; a_{c,s}, b_{c,s})$ to $p(R|c, s)$ for every pair (c, s). $beta(\theta, a, b)$ indicates the beta distribution with shape parameters a, b. Here a, b can be interpreted as the total number of occurrences of negative and positive feedback, respectively, when the user is in context c and is recommended song s. Therefore, the prior captures the personal history of preferences of the user. The probability $p(R = 1|c, s)$ can be expressed as in Equation (4):

$$p(R = 1|c, s) = \frac{b}{a + b} \tag{4}$$

User feedback can be described as a triple $\mathbf{x} = (\mathbf{f}, s, r)$, where $\mathbf{f}$ is a feature vector extracted from mobile phone sensors during the play of a recommended song, s is the recommended song, and r is the observed value of R, which is the user feedback. The value $r = 0$ indicates negative feedback, while $r = 1$ indicates positive feedback.

The user's true context category is unknown, and thus c is a *latent variable* during adaptation. In this situation, updating the beta prior by exact Bayesian learning is computation-intensive, and thus not suitable for mobile phones. Here approximate inference is used to reduce the computation. First the MAP estimation $\hat{c}$ of c is given by Equation (5):

$$\hat{c} = \arg\max_{c} p(c|\mathbf{f}) \tag{5}$$

Then the corresponding beta prior $\hat{\theta}$ for pair $(\hat{c}, s)$ is updated as in Equation (6):

$$p(\hat{\theta}|\mathbf{x}) \approx \begin{cases} beta(\hat{\theta}; a + 1, b) & \text{if } r = 0 \\ beta(\hat{\theta}; a, b + 1) & \text{if } r = 1 \end{cases} \tag{6}$$

Finally, the corresponding $p(R = 1|\hat{c}, s, \mathbf{x})$ representing the user's preference is updated as in Equation (7):

$$p(R = 1|\hat{c}, s, \mathbf{x}) \approx \begin{cases} \frac{b}{a+b+1} & \text{if } r = 0 \\ \frac{b+1}{a+b+1} & \text{if } r = 1 \end{cases} \tag{7}$$

Comparing Equation (7) with Equation (4), we can see that when a user skips song s in context $\hat{c}$, $r = 0$ and the probability of that song $p(R = 1|\hat{c}, s)$ decreases. Thus, s will have a smaller chance of being recommended next time. Otherwise, if the user listened completely ($r = 1$), then the probability $p(R = 1|\hat{c}, s)$ increases, and thus s will be more likely to be recommended next time.

The whole updating process is very efficient: We first obtain the MAP estimation $\hat{c}$, and then the counters a,b and $p(R = 1|\hat{c}, s)$ are updated. The adaptation can be done directly on a mobile phone without the use of backend servers.

We next describe the use of music content analysis results to initialize the beta priors.

3.3.2 Initialization

We model the relationship between music and context by examining the music audio content. There are many existing works on music content classification, such as genre classification, mood classification, and so on. Classification usually assumes that the classes are mutually exclusive. The problem here is different, since one song can be suitable for many context categories. Thus, our problem is similar to the tagging problem: Given a song, we want to know the probability that a tag is suitable for the song. Therefore, we use a state-of-the-art music tagging method called Autotagger [33].

Autotagger estimates the probability $\pi_{c,s}$ that a song s is suitable for context c for *all* users. We use $\pi_{c,s}$ in our model to initialize the prior $beta(\theta; a, b)$ described in Section 3.3.1. First, the ratio of a and b is determined by Equation (8):

$$p(R = 1|c, s) = \frac{b}{a + b} = \pi_{c,s} \tag{8}$$

To further determine a and b, the equivalent sample size β is needed:

$$a + b = \beta$$

	Prediction	Incremental training
AdaBoost	Fast	Slow and Non-trivial
C4.5	Fast	Slow and Non-trivial
LR	Fast	Not supported
NB	Fast	Fast
SVM	Fast	Slow and Non-trivial
KNN	Slow	Fast

Table 1: Comparison of Classifiers

β is a free parameter of the system, balancing user feedback against music content analysis results. A large β indicates a belief that music content analysis results are good enough to provide a good recommendation, and that the adaptation (Equation (7)) will change $p(R = 1|c, s)$ very slowly. On the other hand, a small β indicates that music content analysis is relatively inaccurate, requiring more reliance on user feedback to perform recommendation. From our subjects' experiences, $\beta = 5$ is a reasonable setting.

After initialization, $p(R = 1|c, s)$ is adapted dynamically to the particular user according to Equation (7).

3.4 Sensor-Context Model

There are many ways to infer context categories from sensor data. Choosing a proper model is very important and requires careful consideration. First, since much of the computation is to be done on a mobile phone, energy consumption is critically important. Second, the model needs be accurate. Third, in order to adapt the model to a user on the fly as he/she is using the system, the model should support efficient incremental training.

We considered six popular methods used in activity recognition—AdaBoost, C4.5 decision trees, logistic regression (LR), Naive Bayes (NB), support vector machine (SVM) and K-nearest neighbors (KNN). We compared them from three perspectives: prediction accuracy, overhead of prediction computation, and incremental training. Table 1 compares the methods qualitatively in terms of prediction overhead and incremental training, and we present results on prediction accuracy in Section 5.3.3. We chose Naive Bayes because it offers very good incremental training and prediction overhead with just a small relative loss in accuracy.

Feature vectors extracted from sensor data are usually real-valued vectors. Since Naive Bayes cannot handle real-valued feature attributes directly, we first discretize the attributes using the well known *equal frequency discretization* method. As a result, every feature vector $\mathbf{f}$ becomes a vector of integers: $(f_1, f_2, \ldots, f_v)$, and $1 \leq f_l \leq d_l$, where d_l is the number of bins of the l-th attribute. Using the Naive Bayes assumption (that the features are conditionally independent), the sensor-context model $p(\mathbf{f}|c)p(c)$ can be decomposed as follows:

$$p(\mathbf{f}|c)p(c) = \prod_{l=1}^{v} p(f_l|c)p(c) \tag{9}$$

To estimate the parameters $p(c)$ and $p(f_l|c)$ in Equation (9), training samples need to be collected, which are tuples of the form $(\mathbf{f}^k, c^k)$, where $\mathbf{f}^k$ is the k-th observed feature vector, and c^k is the corresponding context category. Then, based on Maximum Likelihood Estimation, parameters are learned using Equations (10) and (11), where $n(c)$ indicates the number of times that category c occurs in the training samples, and $n(F_l = f, c)$ indicates the number of times that the l-th attribute of $\mathbf{f}$ is f and the context category is c:

$$p(c) = \frac{n(c)}{\sum_{i=1}^{|\mathcal{C}|} n(c_i)} \tag{10}$$

$$p(f_l|c) = \frac{n(F_l = f_l, c)}{\sum_{f=1}^{d_l} n(F_l = f, c)} \tag{11}$$

An alternative to incremental training for adaptation is to store all old training data in the mobile phone, and then newly arriving training data is combined with the old data and a new model trained again on the combined dataset. We argue that this is not suitable for a mobile application. First, storing all the training data in the mobile phone is too expensive due to the limited storage space. Second, retraining the model on the complete data after each update would be too expensive computationally.

For these reasons we opt for incremental training. First, it trains a model on some training data and then discards that training data. When new data arrives, instead of training a completely new model, it uses the new training data to update the model *incrementally* and then again discards the new training data. In this way, no training data needs to be stored, and the model can be updated efficiently.

Incremental training in Naive Bayes is straightforward. According to Equation (10) and (11), the parameters are estimated using counters $n(c)$ and $n(F_l = f, c)$, which are the sufficient statistics of the sensor-context model. When new training data arrives, these counters are updated, and then parameters $p(c)$, $p(F_l = f|c)$ are updated via Equations (10) and (11). In this way, the sensor-context model can be efficiently updated to adapt to the user.

4. SYSTEM IMPLEMENTATION

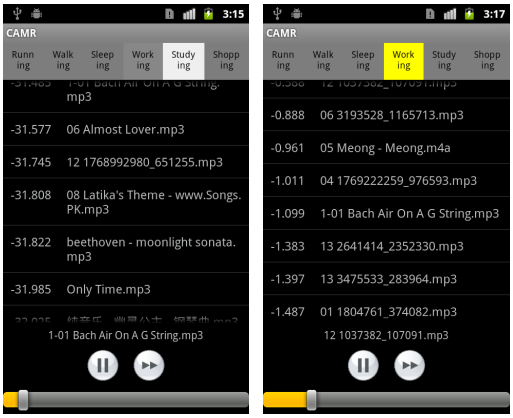
We have implemented the ACACF model in a prototype system, which comprises two components: (1) music audio content analysis on a remote server, and (2) a context-aware music recommender application on a mobile phone.

Music content analysis is done on a server since it is compute-intensive and needs to be performed just once per song. Doing it on a mobile phone would quickly drain the battery.

The mobile application is implemented on the Android SDK, and its interface is depicted in Figure 2. To stream music, the application connects to the server via a wireless connection (3G or WiFi). The application also can run without connecting to the server, but then songs and music content analysis results must be cached beforehand.

At the top of the user interface is a list of activities. Users can let the system infer his/her current activity automatically, which is called the *auto mode* and is shown as Figure 2a. The background intensity of the activity labels is adjusted according to the inferred probabilities. The whiter the background is, the higher the activity's probability is. Users also can select a single category manually, which is called *manual mode* and is shown as Figure 2b. When an activity is selected manually, its background becomes yellow. To switch back to auto mode from manual model, the user just needs to tap the yellow label once.

When the application is in manual mode, the selected activity and sensor data are used to update the sensor-context model described in Section 3.4, which makes context inference increasingly accurate. Ideally, manual mode will not



(a) auto mode (b) manual mode

Figure 2: Context-aware mobile music recommender.

be needed after several days since auto mode should be accurate enough by then.

The list in the middle of the user interface contains the recommended songs ranked by the probabilities described in Equation (3); logarithms of these probabilities are shown on the left side of the songs. At the bottom of the user interface are play/pause and skip buttons.

The adaptation described in Section 3.3.1 is performed whenever the user finishes listening to a song or skips a song. After adaptation, the probability of the song just listened to or skipped will be updated, and all songs will be re-ranked. This makes the list of recommended songs increasingly accurate, thereby adapting to the user's personal preferences.

5. EXPERIMENTS

In this section we describe results from our evaluation of the ACACF model and its prototype implementation. We have conducted extensive experimental evaluations of both model accuracy and system usability, and the results demonstrate significant promise from both perspectives.

5.1 Datasets

5.1.1 Playlists Crawled from the Web

To build and evaluate the music-context model, we require a large number of songs with context labels, which we use as ground truth for activity prediction. One dataset we considered is the publicly available CAL500 dataset, which incorporates some usage annotations such as *driving* and *sleeping* [34]. However, those annotations only partially cover our six categories. Furthermore, although the annotations were made by a large number of subjects (66 undergraduates), each subject annotated only a small portion of the dataset, and each song was annotated only by around three subjects, which is too few to obtain reliable results.

For these reasons, we constructed a new, larger dataset of 24224 songs crawled from Grooveshark[3] and YouTube[4]. Grooveshark has numerous playlists created by users , titled with context information such as *Studying*, *running songs*, etc. From Grooveshark we therefore collected playlists that match our context categories. The audio tracks for the songs were then crawled from YouTube through YouTube's open data API. Details of the dataset are presented in Table 2; the total number of 22108 distinct songs shown in the table is less than 24224, since latter number includes songs not associated with any of our six context categories.

[3] `http://grooveshark.com`
[4] `http://www.youtube.com`

Context	Playlists	Distinct Songs	Observations
Running	393	3430	7810
Walking	197	3601	4123
Sleeping	195	3941	5318
Working	194	4533	4988
Studying	195	3405	4363
Shopping	77	3786	3847
Total	1251	22108	30449

Table 2: Summary of the Grooveshark Dataset. *Distinct Songs* indicates the number of distinct songs from the playlists for the specified context, while *Observations* indicates the total number of songs including duplicates.

5.1.2 Context Annotation of 1200 Songs

From the 22108 distinct songs shown in Table 2, we selected 1200 for annotation. It was necessary to consider fewer songs for two reasons. First, data crawled from the Web is inevitably noisy, since some users may be careless in their creation of playlists. Second, in order to verify agreement between different users, we require songs labeled by multiple users. In the Grooveshark dataset, most songs exist in only a single playlist. For these reasons, it was necessary to carry out an additional phase of annotation in which *all* songs were annotated by *multiple* subjects to produce the ground truth classification for our study. 1200 songs provides a large sample size but not so large as to make the annotation effort unreasonable for our subjects. We randomly chose the 1200 songs so that there would be roughly an equal number of songs from each context category (as classified by the Grooveshark playlist titles).

We recruited 10 students to annotate all 1200 songs. There were equal numbers of males and females. All of them listen to music at least one hour a day, and exercise regularly (at least 3 hour-long sessions per week). They have different culture background and are from India, Malaysia, Singapore, Indonesia, China, and Vietnam. Every participant was rewarded with a small token payment for their time and effort. Participants were chosen with the requirement that they listen to music regularly for at least one hour per day. Annotation was performed through a Web site we set up that simply required clicking checkboxes. Because different parts of the same song can have very different styles, we required the subjects to listen to each song for at least 45 seconds. Subjects were allowed to advance or rewind the music playback. For each song, each subject selected one or more suitable context categories.

The resulting dataset thus contains 1200 songs, with each song annotated by all 10 subjects, and with each subject having selected one or more context categories per song.

5.1.3 Sensor Data Collection

To build and evaluate the sensor-context model, we had the same 10 subjects collect data from onboard sensors on

Activity	Kappa Agreement	Percent Agreement
Running	**0.27**	0.35
Working	0.03	0.02
Sleeping	**0.29**	0.28
Walking	0.03	0.03
Shopping	0.07	0.17
Studying	0.09	0.11

Table 3: Inter-Subject Agreement on Music Preferences for Different Activities

their mobile phones. The sensors we used were gyroscopes, accelerometers, GPS receivers, microphones and ambient light sensors.

Sensor data was collected by a mobile application we designed, which will be offered to other interested researchers in the future. All the mobile phones used are based on Android OS. To make the trained model robust to different phone models, we provided our subjects with five different phone models we purchased from Samsung and HTC. The quality of these phones is also different. Some are expensive and have all the sensors mentioned above, while some are cheaper models having only accelerometer, a GPS receiver and a microphone. We imposed no restrictions on how the subjects held or carried or used their phones. To record a data session, a subject first selected their current context category from the application interface and then recorded 30 minutes of data. Each subject was required to record one session for every context category. The recorded sensor data and selected context category were stored together in a SQLite database on the mobile phone's SD card. The resulting 30-hour dataset contains 6 context categories, and every category has 0.5 hour sensor data collected by every of the 10 subjects.

5.2 Music-Context Model Evaluation

Demonstrating agreement on suitable songs for an activity is a very important foundation for music content analysis, because if there is no agreement among people, then a trained music-context model will work only for the users in the training set but will not reliably generalize to other users. Therefore, we first studied inter-subject agreement. Fleiss's Kappa [35] and percent agreement were calculated among the 10 subjects for every context category, and the results are presented as Table 3. We observe that all Kappa values are significantly higher than 0 (p-value < 0.0001) and are and especially high for *running* and *sleeping*. The results therefore indicate that subjects have statistically significant agreement on context categories, indicating the feasibility of training generalizable statistical models.

Next, the music-context model was trained. The videos we crawled from YouTube were first converted by *ffmpeg*[5] into mono channel WAV files with a 16KHz sampling rate. Then feature vectors were extracted using a program we developed based on the MARSYAS library[6], in which a window size of 512 was used without overlapping. The features we used and their dimensionalities are ZeroCrossing (1), Centroid (1), Rolloff (1), Flux (1), MFCC (13), Chroma (14), SCF (24) and SFM (24). To reduce the training set size, we used the mean and standard deviation of feature vectors computed from every 30-second period. Finally, the we

[5]`http://ffmpeg.org`

[6]`http://marsyas.sourceforge.net`

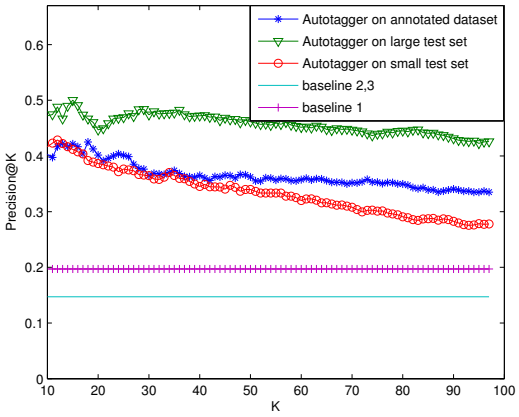


Figure 3: Retrieval performance of the music-context model.

added the 1-dimensional feature *tempo* to the summarized feature vectors. So the resulting combined feature vector is $79 \times 2 + 1 = 159$-dimensional.

We split the Grooveshark dataset into three disjoint subsets: a training set of 16281 songs, a large test set of 6943 songs, and our annotated dataset of 1200 songs, which we also used as a test set. We used the Autotagger method for the training: Using the training set, one binary AdaBoost classifier was trained for every context category. The classifier for context c_i estimates the probability that a song s_j is suitable for context c_i, which is $p(R = 1|c_i, s_j)$.

To measure the accuracy of these classifiers, we simulated the following retrieval process: Given context c as the query, we use its corresponding classifier to compute the probability $p(R = 1|c, s_j)$ for every song s_j and then rank all songs in descending order according to the estimated probabilities. Then the top-K songs are returned. Suppose there are only L songs of the top-K are labeled with context c in our dataset. Then L/K is the Precision@K for context c. The final Precision@K is the average of all Precision@K for the six categories.

We tested the classifiers on the three datasets. For our dataset of 1200 annotated songs, we used majority voting to determine the context for every song. For instance, if at least six of the 10 subjects annotated a song as being suitable for context c, then the song was labeled with c. The retrieval performance measured by Precision@K depends on the test set size, because the more songs we have, the more likely that we can find good songs for a context category, and thus the Precision@K will be higher. Therefore, in order to produce results that are comparable between our annotated song set and the large test set of 6943 songs, a set of 1200 songs was randomly sampled from the large test set; we refer to this as the *small test set* below.

We used a random estimator as our baseline, which ranks all songs randomly. The random estimator was also tested on the annotated dataset (baseline 1), the large test set (baseline 2) and the small test set (baseline 3). All results are presented in Figure 3. We observe that our trained models significantly outperformed the random estimator. Therefore, the models are able to associate a song accurately with daily activities by examining the song's audio content.

	AdaBoost	C4.5	LR	NB	SVM	KNN
Running	0.974	0.976	0.975	0.841	0.974	0.97
Working	0.933	0.932	0.921	0.876	0.929	0.922
Sleeping	0.999	0.999	0.999	0.994	0.999	0.993
Walking	0.961	0.960	0.955	0.909	0.960	0.953
Shopping	0.972	0.972	0.948	0.953	0.965	0.955
Studying	0.854	0.867	0.835	0.694	0.860	0.855
overall	0.951	0.952	0.941	0.893	0.950	0.943

Table 4: Activity Classification Accuracy

5.3 Sensor-Context Model Evaluation

5.3.1 Sensor Selection

Time data were used with data from accelerometers and microphones in our sensor-context model. Although GPS data are used by much of the previous work in human activity recognition, we did not use it for our own work because our activity set is different from other work, plus GPS appears not to increase classification accuracy even though it consumes a great deal of power. Additionally, we did not use the ambient light sensors and gyroscopes, for two reasons: First, gyroscopes do not improve accuracy very much since accelerometers already provide good motion data. Second, both kinds of sensors reside only in a small number of relatively expensive phones, while our aim is to build a model suitable for most available Android phones.

5.3.2 Feature Extraction from Sensor Data

Performing feature extraction from sensor data involves computing feature vectors from the sensor data stream.

Human daily activities have very strong time regularity. Most of us sleep at night and work during the day. Therefore, time is a very important feature for daily activity recognition, and we use the hour of the day in our feature set.

Window size in feature extraction is important. Generally, a larger window size can make inference more accurate because it captures more information. However, a larger window size also reduces system responsiveness, thus degrading the user experience. From our experience, a window size of five seconds appears to be a reasonable setting.

Each accelerometer data sample has three axes, x, y and z. From this data we use the magnitude $m = \sqrt{x^2 + y^2 + z^2}$, which is robust to the direction of the phone. Then the mean, standard deviation, minimum and maximum of all five-second samples of m are used in the final feature vectors.

For audio data from a microphone, we calculate the average amplitude of all samples as a measure of how noisy the environment of the phone is is. The final feature vector therefore has $1 + 4 + 1 = 6$ dimensions.

5.3.3 Context Classification Accuracy

We evaluated AdaBoost, C4.5, LR, NB, SVM and KNN for context classification, and we used 10-fold cross-validation to compare their accuracy. The results are presented in Table 4, with a value of 1.0 representing perfect accuracy. We observe that while NB is not as accurate as other methods, it still produces very good results. The categories *studying* and *working* are not distinguished well by any of the methods, because the context information sensed during those two activities is very similar. In fact, sometimes even human beings cannot distinguish the two.

	Questions	Mean	Stdev
Q1	I prefer different music (different genre, tempo, pitch, dynamics etc.) when I'm in different context (In different contexts means doing different things e.g. running, sleeping, or at different places e.g. school, home).	3.7	0.95
Q2	I usually listen to different sets of music when I'm in different context .	3.5	1.18
Q3	It is time consuming to create different lists of songs for different contexts with existing technologies.	4.4	0.97
Q4	It is not convenient to change music when I'm doing other things with existing technologies.	4.0	0.94
Q5	I want to have a mobile application that can accurately play suitable music to me according to my context automatically.	4.4	0.52

Table 5: Questionnaire 1

5.4 User Study

5.4.1 User Needs Study

To understand user needs for music recommendation, we conducted a survey (Questionnaire 1) with our 10 subjects. The questionnaire and survey results are presented in Table 5. All questions were answered on a 5-point Likert scale from "strongly disagree" (1) to "strongly agree" (5). $Q1$ helps in understanding user needs for a context-aware music experience; the results demonstrate that subjects generally prefer different music in different contexts. The results for $Q2$, $Q3$ and $Q4$ demonstrate that their requirements cannot be satisfied very well with the existing technologies. Finally, the results for $Q5$ demonstrate that a context-aware mobile music recommender potentially can satisfy their needs better.

5.4.2 Evaluation of Recommendation Quality

Most existing music recommender systems, including context-aware ones, require user ratings or annotations. During the cold-start stage, these systems are able only to recommend songs randomly. Comparison with existing CAMRSs is impossible, for three reasons: First, we focus on daily activities, and none of the reported literature has used these before. Second, most existing CAMRSs do not infer context categories from mobile phone sensor data. Third, most existing CAMRSs do not use music content analysis. Therefore, for our evaluation we undertook a comparison between the following three kinds of recommendations:

(R1) recommending songs completely randomly. This simulates traditional recommender systems during the cold-start stage.

(R2) recommending songs with context category inferred by the system automatically. This is the auto mode of our application.

(R3) recommending songs with context category selected by subjects manually. This is the manual mode of our application.

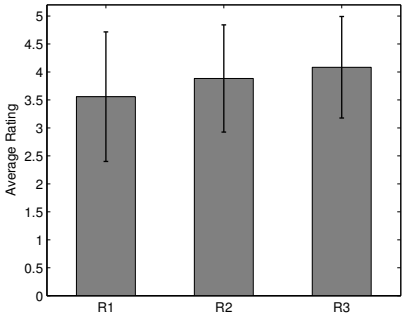


Figure 4: Average recommendation ratings. The error bars show the standard deviation of the ratings.

The same 10 subjects participated in this evaluation. The subjects were divided into an experimental group and a control group of five subjects each. The experimental group tested R2 and R3, and the control group tested R1. The subjects did not know which group they were in. All phones were supplied with the music content analysis results and with an identical set of 800 songs chosen randomly from the large test set of 6943 songs described in Section 5.2. During evaluation, each subject did each of the six activities for about 20 minutes while listening to the activity's top recommended songs. Each song was played for about a minute and then rated with the above 5-point Likert scale. Thus, the higher the rating for a song, the more the subject liked the song. Adaptation of both the music-context model and sensor-context model was turned off during this evaluation.

The average and standard deviation of the resulting ratings are presented in Figure 4. We observe that R2 performs significantly better than R1 (p-value = 0.0478), and R3 is much better than R1 (p-value = 0.0001). These results indicate that context-awareness combined with music content analysis can produce significantly better results than random recommendation. Therefore, our system provides a promising solution to the cold-start problem. R3 is better than R2 but not significantly better (p-value=0.1374), demonstrating that auto mode is almost as good as manual mode, and further demonstrating the accuracy of automated context inference.

5.4.3 Adaptation Evaluation

Two of our 10 subjects participated in a one-week adaptation evaluation. The subjects used the application continuously every day for a week. Most of the time the application was used in auto mode. If a subject found that the recommended songs did not match his/her activity, he/she could switch the application to manual mode or skip the recommended song. The whole system was updated continuously and became more and more accurate over the one-week period with respect to the subject's preferences. We compared the accuracy of both context inference and recommendation quality, both before the one-week adaptation period and after the one-week adaptation period.

Context inference: The trained Naive Bayes model described in Section 5.3.3 was used as the initial sensor-context model. Before and after adaptation, each subject's sensor-context model was evaluated on sensor data collected by that subject. The average classification accuracy for the two subjects is presented in Table 6.

	Before Adaptation	After Adaptation
Context Inference	0.87	0.96
Recommendation	0.68	0.93

Table 6: Context Inference and Recommendation Accuracy Before and After Adaptation

	Questions	Mean	Stdev
Q6	I can fully understand the functionalities of the mobile application and it's easy to use.	4.6	0.51
Q7	I'm willing to use the application if I have a copy.	4.4	0.84

Table 7: Questionnaire 2

Recommendation: Each subject rated the top-20 recommended songs with "like", and "dislike" for every context category, both before and after adaptation. Recommendation accuracy is defined as the proportion of liked songs. The results are presented as Table 6.

We observe that the accuracy of both context inference and recommendation increased significantly after one week of adaptation. Therefore, our model is able to adapt accurately to individual user preferences.

5.4.4 User Experience

All 10 subjects completed a second survey at the end of the study (Questionnaire 2). Two of the questions and the survey results are presented in Table 7. All questions were answered with a 5-point Likert scale as before. We observe that most of the subjects agree that the application is easy to use and are willing to use it. One subject commented, "I really like the idea, hope you can improve on it and sell it". However, some of the subjects thought that more context categories should be added, such as: *entertaining* (playing games, surfing the Web), *cooking*, *traveling* (bicycle/bus/subway), *partying* and *just relaxing and enjoying music* (no work, no study). Two of the subjects thought the interface could be made more appealing.

6. RELATED WORK

6.1 Existing CAMRSs

XPod is a mobile music player that selects songs matching a users' emotional and activity states [16]. The player uses an external physiological data collection device called BodyMedia SensorWear. Compared to the activities we consider, the activities considered in XPod are very coarse-grained, namely *resting*, *passive* and *active*. User ratings and metadata are used to associate songs with these activities, but recommendation quality is not evaluated.

In addition to XPod, many other CAMRs exploit user emotional states as a context factor. Park *et al.* were probably the first to propose the concept of context-aware music recommendation [12]. They used a fuzzy Bayesian network to infer a user's mood (*depressed*, *content*, *exuberant*, or *anxious/frantic*) from context information including weather, noise, time, gender and age. Music is then recommended to match the inferred mood using mood labels manually annotated on each available song. In the work of Cunningham *et al.*, user emotional states are deduced from user movements,

temperature, weather and lighting of the surroundings based on reasoning with a manually built knowledge base [18]. Rho *et al.* built an ontology to infer a user's mood from context information including time, location, event, and demographic information, and then the inferred mood is matched with the mood of songs predicted from music content [23,27]. However, we believe that with current mobile phones, it is too difficult to infer a user's mood automatically.

Some work tries to incorporate context information into CF. Lee *et al.* proposed a context-aware music recommendation system based on case-based reasoning [21]. In this work, in order to recommend music to a target user, other users who have similar context as the target user are selected, and then CF is performed among the selected users and the target user. This work considers time, region and weather as the context information. Similarly, Su *et al.* use context information that includes physiological signal, weather, noise, light condition, motion, time and location to pre-filter users and items [25]. Both context and content information are used to calculate similarities and are incorporated into CF.

SuperMusic [19], developed at Nokia, is a streaming context-aware mobile service. Location (GPS and cell ID) and time are considered as the context information. Since that application cannot show a user's current context categories explicitly, many users get confused about the application's "situation" concept: "If I'm being honest, I didn't understand the concept of situation. ... I don't know if there is music for my situation at all?" [19]. This inspired us to design our system recommending music explicitly for understandable context categories such as *working*, *sleeping*, etc.

Resa *et al.* studied the relationship between temporal information (e.g., day of week, hour of day) and music listening preferences such as genre and artist [26]. Baltrunas *et al.* described similar work that aims to predict a user's music preference based on the current time [30]. Several other works exploit physiological signals to generate music playlists automatically for exercising [11, 14, 17]. Only the tempo attribute of music was considered in those works, whereas in our work, several music audio features including timbre, pitch and tempo are considered. Kaminskas *et al.* recommend music for places of interest by matching tags of places with tags on songs [28]. In other work by Baltrunas *et al.*, a song's most suitable context is predicted from user ratings [36], and a CAMRS was built specifically for the context of riding in a car [37].

Reddy *et al.* proposed a context-aware mobile music player but did not describe how they infer context categories from sensor data or how they combine context information with music recommendation [15]. In addition, they provide no evaluation of their system. Seppänen *et al.* argue that mobile music experiences in the future should be both personalized and situationalized (i.e., context-aware) [20]. In this work, no CAMRS is implemented. Bostrom *et al.* also tried to build a context-aware mobile music recommender system, but the recommendation part was not implemented, and no evaluation is presented [38].

6.2 Content Analysis in CAMRSs

Only a relatively small number of CAMRSs described in the literature use music content information. To associate songs with context categories such as emotional states, most of them use manually supplied metadata or annotation labels or ratings [12,16–19,21,28,36], or implicit feedback [16]. In other cases, content is not directly associated with context, but is used instead to measure the similarity between two songs in order to support content-based recommendation [19, 25].

The method of Rho, Han *et al.* is similar to ours [23, 27]: Emotion classifiers are first trained, and then every song is classified into a single emotional state. But there are also differences: First, we associate music content with daily activities instead of mood. Second, in our models, a song can belong to multiple categories simultaneously. The method that we use to associate music content with daily activities is called Autotagger [33]. Similar methods have been proposed by others [34], but Autotagger is the only one evaluated on a large dataset. All these methods were used originally to annotate songs with multiple semantic tags, including genre, mood and usage. Although their tags include some of our context categories such as *sleeping*, the training dataset used in these studies (the CAL500 dataset discussed in Section 5.1.1) is too small (500 songs with around 3 annotations per song), and evaluations were done together with other tags. From their reported results, it is difficult to know whether or not the trained models capture the relationship between daily activities and music content.

6.3 Context Inference in CAMRSs

None of the existing CAMRSs tries to infer user activities using a mobile phone. XPod uses an external device for classification—the classified activities are very low-level, and classification is performed on a laptop [16]. While activity recognition using mobile phones is itself not a new idea, none of the systems that have been studied can be updated incrementally to adapt to a particular user [5–10]. In one remarkable work, Berchtold *et al.* proposed an activity recognition service supporting online personalized optimization [7]. However, their model needs to search in a large space using a genetic algorithm, which requires significant computation. And to update the model to adapt to a particular user, all of that user's sensor data history is needed, thus requiring significant storage.

7. CONCLUSION

In this paper we have described a novel probabilistic model for music recommendation that combines automated activity classification with automated music content analysis, with support for a rich set of activities and music content features. We collected three datasets—a set of playlists from the Web, a set of 1200 cleanly annotated songs, and a set of sensor data recorded from daily activities. These datasets will be offered eventually to researchers who want to carry out related research on context-aware music recommendation. Based on the set of 1200 annotated songs, we found that although context annotation can be subjective, people nevertheless often do agree on their annotations. Using the datasets, both the sensor-context model and the music-context model were evaluated, and the results are very promising. Based on the probabilistic model, we implemented a CAMRS for off-the-shelf mobile phones. The results from our user study demonstrate that the system is easy to use and can provide good recommendations even in the absence of pre-existing user ratings or annotations, a situation in which traditional systems only can recommend songs randomly. Therefore, our system satisfies users'

short-term needs better because of context-awareness, and also provides a solution to the cold-start problem. Evaluation results demonstrate that our system can update itself in real-time to adapt to a particular user.

We are currently improving the user experience of the system and will eventually publish it in the Google Android Market. We also are exploring the integration of context-awareness with collaborative filtering to provide even more accurate recommendations.

8. ACKNOWLEDGMENTS

This work is supported by grant R-252-000-473-133 and R-252-000-428-290 from the School of Computing at the National University of Singapore.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the State-of-the-Art and possible extensions," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, pp. 734–749, June 2005.

[2] A. C. North, D. J. Hargreaves, and J. J. Hargreaves, "Uses of Music in Everyday Life," *Music Perception: An Interdisciplinary Journal*, vol. 22, no. 1, 2004.

[3] D. J. Levitin and J. McGill, "Life Soundtracks: The uses of music in everyday life." 2007.

[4] G. Reynolds, D. Barry, T. Burke, and E. Coyle, "Interacting with large music collections: Towards the use of environmental metadata," in *ICME*, June 2008.

[5] T. S. Saponas, J. Lester, J. Froehlich, J. Fogarty, and J. Landay, "iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones," in *UW CSE Tech Report*, 2008.

[6] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity Recognition from Accelerometer Data on a Mobile Phone," in *IWANN*, 2009.

[7] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke, and M. Beigl, "ActiServ: Activity Recognition Service for mobile phones," in *ISWC*, pp. 1–8, Oct. 2010.

[8] M. Khan, S. I. Ahamed, M. Rahman, and R. O. Smith, "A Feature Extraction Method for Real time Human Activity Recognition on Cell Phones," in *isQoLT*, 2011.

[9] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *SIGKDD Explor. Newsl.*, vol. 12, pp. 74–82, Mar. 2011.

[10] Y. S. Lee and S. B. Cho, "Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer," in *HAIS*, pp. 460–467, 2011.

[11] G. Wijnalda, S. Pauws, F. Vignoli, and H. Stuckenschmidt, "A Personalized Music System for Motivation in Sport Performance," *IEEE Pervasive Computing*, vol. 4, pp. 26–32, July 2005.

[12] H.-S. Park, J.-O. Yoo, and S.-B. Cho, "A Context-Aware Music Recommendation System Using Fuzzy Bayesian Networks with Utility Theory," in *FSKD*, 2006.

[13] J.-H. Kim, C.-W. Song, K.-W. Lim, and J.-H. Lee, "Design of Music Recommendation System Using Context Information," in *LNCS*, vol. 4088, ch. 83, pp. 708–713, 2006.

[14] G. T. Elliott and B. Tomlinson, "PersonalSoundtrack: context-aware playlists that adapt to user pace," in *SIGCHI*, 2006.

[15] S. Reddy and J. Mascia, "Lifetrak: music in tune with your life," in *HCM*, 2006.

[16] S. Dornbush, A. Joshi, Z. Segall, and T. Oates, "A Human Activity Aware Learning Mobile Music Player," in *Proc. of the 2007 conference on Advances in Ambient Intelligence*, 2007.

[17] R. D. Oliveira and N. Oliver, "TripleBeat: enhancing exercise performance with persuasion," in *MobileHCI*, 2008.

[18] S. Cunningham, S. Caulder, and V. Grout, "Saturday Night or Fever? Context-Aware Music Playlists," in *AM '08*, 2008.

[19] A. Lehtiniemi, "Evaluating SuperMusic: streaming context-aware mobile music service," in *ACE*, 2008.

[20] J. Seppänen and J. Huopaniemi, "Interactive and context-aware mobile music experiences," in *DAFx-08*, Sept. 2008.

[21] J. Lee and J. Lee, "Context Awareness by Case-Based Reasoning in a Music Recommendation System," *Ubiquitous Computing Systems*, pp. 45–58, 2008.

[22] H. Liu, J. Hu, and M. Rauterberg, "Music Playlist Recommendation Based on User Heartbeat and Music Preference," in *ICCTD*, 2009.

[23] S. Rho, B. J. Han, and E. Hwang, "SVR-based music mood classification and context-based music recommendation," in *ACM MM*, 2009.

[24] A. Camurri, G. Volpe, H. Vinet, R. Bresin, M. Fabiani, G. Dubus, E. Maestre, J. Llop, J. Kleimola, S. Oksanen, V. Välimäki, and J. Seppanen, "User-Centric Context-Aware Mobile Applications for Embodied Music Listening User Centric Media," in *LNICST*, pp. 21–30, 2010.

[25] J.-H. Su, H.-H. Yeh, P. S. Yu, and V. S. Tseng, "Music Recommendation Using Content and Context Information Mining," *Intelligent Systems, IEEE*, vol. 25, pp. 16–26, Jan. 2010.

[26] Z. Resa, "Towards Time-aware Contextual Music Recommendation: An Exploration of Temporal Patterns of Music Listening Using Circular Statistics," Master's thesis, 2010.

[27] B. J. Han, S. Rho, S. Jun, and E. Hwang, "Music emotion classification and context-based music recommendation," *Multimedia Tools Appl.*, vol. 47, pp. 433–460, May 2010.

[28] M. Kaminskas and F. Ricci, "Location-Adapted Music Recommendation Using Tags.," in *UMAP*, 2011.

[29] D. Leake, A. Maguitman, and T. Reichherzer, "Cases, Context, and Comfort: Opportunities for Case-Based Reasoning in Smart Homes," in *Designing Smart Homes*, LNCS, 2006.

[30] L. Baltrunas and X. Amatriain, "Towards Time-Dependant Recommendation based on Implicit Feedback," in *CARS*, 2009.

[31] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *SIGIR*, 2002.

[32] Y. Hu and M. Ogihara, "Nextone player: A music recommendation system based on user behavior," in *ISMIR*, 2011.

[33] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere, "Autotagger: A Model for Predicting Social Tags from Acoustic Features on Large Music Databases," *JNMR*, vol. 37, pp. 115–135, June 2008.

[34] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Towards musical query-by-semantic-description using the CAL500 data set," in *SIGIR*, 2007.

[35] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data.," *Biometrics*, vol. 33, pp. 159–174, Mar. 1977.

[36] L. Baltrunas, M. Kaminskas, F. Ricci, L. Rokach, B. Shapira, and K. H. Luke, "Best Usage Context Prediction for Music Tracks," in *CARS*, Sept. 2010.

[37] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, "InCarMusic: Context-Aware Music Recommendations in a Car E-Commerce and Web Technologies," in *LNBIP*, 2011.

[38] F. Boström, "AndroMedia - Towards a Context-aware Mobile Music Recommender," Master's thesis, 2008.