NEXTONE PLAYER: A MUSIC RECOMMENDATION SYSTEM BASED ON USER BEHAVIOR

Yajie Hu
Department of Computer Science
University of Miami
yajie.hu@umail.miami.edu

Mitsunori Ogihara
Department of Computer Science
University of Miami
ogihara@cs.miami.edu

ABSTRACT

We present a new approach to recommend suitable tracks from a collection of songs to the user. The goal of the system is to recommend songs that are favored by the user, are fresh to the user's ear, and fit the user's listening pattern. We use "Forgetting Curve" to assess freshness of a song and evaluate "favoredness" using user log. We analyze user's listening pattern to estimate the level of interest of the user in the next song. Also, we treat user behavior on the song being played as feedback to adjust the recommendation strategy for the next one. We develop an application to evaluate our approach in the real world. The user logs of trial volunteers show good performance of the proposed method.

1. INTRODUCTION

As users accumulate digital music in their digital devices, the problem arises for them to manage the large number of tracks in them. If a device contains thousands of tracks, it is difficult, painful, and even impractical for a user to pick suitable tracks to listen to without using pre-determined organization such as albums, playlists or computationally generated recommendation, which is the topic of this paper.

A good recommendation system should be able to minimize user's effort required to provide feedback and simultaneously to maximize the user's satisfaction by playing appropriate song at the right time. Reducing the amount of feedback is an important point in designing recommendation systems, since users are in general lazy. We thus evaluate user's attitude towards a song from partitioning of playing time. In particular, if a song is played from beginning to end, we infer that the user likes the song and it is a satisfying recommendation. On the other hand, if the song is skipped while just lasting a few seconds, we assume that the user dislikes the song at that time and the recommendation is less effective.

Using this idea we propose a method to automatically recommend music in a user's device as the next song to be played. In order to keep short the computation time for recommendation, the method is based on metadata and user behavior rather than on content analysis. Which song should be played next can be determined based on various factors. In this paper, we use five perspectives: genre, year, favor, freshness and time pattern.

The rest of this paper is organized as follows. In Section 2, we introduce recent related work. In Section 3 we describe our method for calculating recommendation. We will evaluate this method in Section 4. We conclude by discussing possible future work in Section 5.

2. RELATED WORK

Various song recommendation approaches have been developed so far. We can categorize these approaches in different views.

Automatical playlist generation focuses on recommending songs that are similar to chosen seeds to generate a new playlist. Ragno [1] provided an approach to recommend music that is similar to chosen seeds as a playlist. Similarly, Flexer [2] provided a sequence of songs to form a smooth transition from a start song till the end song. These approaches ignore user's feedback when the user listens to the songs in the playlist. They have an underlying problem that all seed-based approaches produce excessively uniform lists of songs if the dataset contains lots of music cliques. In iTunes, Genius employs similar methods to generate a playlist from a seed.

Dynamic music recommendation improves automatic playlist generation by considering the user's feedback. In the method proposed by Pampalk [3], playlist generation starts with an arbitrary song and adjusts the recommendation result based on user feedback. This type of method is similar to Pandora.

Collaborative-filter methods recommend pieces of music to a user based on rating of those pieces by other users with

similar taste [4]. However, collaborative methods require many users and many ratings and are unable to recommend songs that have no ratings. Hence, users have to be well represented in terms of their taste if they need effective recommendation. This principle has been used by various social websites, including Last.fm, myStrands.

Content-based methods computes similarity between songs, recommends songs similar to the favorite songs, and removes songs that are similar to the skipped songs. In an approach proposed by Cano [5], acoustic features of songs are extracted, such as timbre, tempo, meter and rhythm patterns. Furthermore, some work expresses similarity according to songs emotion. Cai [6] recommends music based only on emotion.

Hybrid approaches, which combine music content and other information, are receiving more attention lately. Donaldson [7] leverages both spectral graph properties of an item-base collaborative filtering as well as acoustic features of the music signal. Shao et al. [8] use both content features and user access pattern to recommend music.

Context-based methods take context into consideration. Liu et al. [9] take the change in the interests of users over time into consideration and add time scheduling to the music playlist. Su et al. [10] improve collaborative filtering using user grouping by context information, such as location, motion, calendar, environment conditions and health conditions, while using content analysis assists system to select appropriate songs.

3. METHOD

We determine whether a song is to be recommended as the next one in the playlist from five perspectives: genre, year, favor, freshness and time pattern.

We use time series analysis of genre and year to predict these attributes of the next song rather than to select the song with similar genre and year to the current song. The reason is that some users like listening similar songs according to genre and year while others perhaps love mixing songs and the variance on genre and year. Hence, we cannot assume that a similar song to the current one can be reasonably seen as a good choice for recommendation. Prediction using time series analysis caters better to a user's taste.

Obviously, the system should recommend users' favorite songs to them. How many times a song has been actively played and how many times the song has been completely played can be used to infer the strength of favor to the song. We collected user's behavior to analyze the favor of songs.

In common sense, a few users dislike listening to a song many times in a short period of times, even though the song could be the user's favorite. On the other hand, some songs that the user favored many months ago may be now old and a little bit insipid. However, if the system recommend them at right time, the user may feel it is fresh and enjoy the experience. Consequently, we take freshness of songs into consideration.

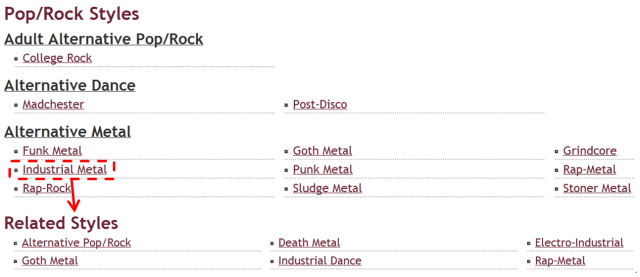

Figure 1. Genre taxonomy screenshot in AllMusic.com

Due to activities and biological clock, users have different tastes in choosing music. In a different period of a day or a week, users tend to select different styles of songs. For example, in the afternoon, a user may like a soothing kind of music for relaxation and may switch to energetic songs in the evening. This paper uses a Gaussian Mixture Model to represent the time pattern of listening and compute the probability of playing a song at that time.

3.1 Genre

The sequence of recent playing of a user represents the user's habit of listening so we analyze the playing sequence using a time series analysis method to predict the genre of the next song. The system records recent 16 songs that were played for at least a half of their length. Although most of the songs record their genres and years are available in ID3v1 or ID3v2 tags, a part of tags are notoriously noisy. Hence, we developed a web wrapper to collect genre information from AllMusic.com, a popular music information website, and use that information to retrieve songs' genres. The ID3v1 or ID3v2 tags will be used unless AllMusic.com has no information about the song.

Furthermore, AllMusic.com not only has a hierarchical taxonomy on genre but also provides subgenres with related genres. The hierarchical taxonomy and related genres are shown in Figure 1. For example, `Industrial Metal`, whose parent is `Alternative Metal`, is related to `Alternative Pop/Rock`.

We use the taxonomy to build an undirected distance graph, in which each node describes a genre and each edge's value is the distance between two genres. The values of the graph are initialized by a maximum value. The parent and related relationship are valued at a different distance, which varies by the depth in the taxonomy, that is, high level corresponds to larger distance while low level corresponds to smaller distance. Then, we assume the distance is transitive and update the distance graph as follows until there is no cell update.

$$E_{ij} = \min_k \left(E_{ij}, E_{ik} + E_{kj} \right), \tag{1}$$

where E_{ij} is the value of edge (i, j). Therefore, we obtain the similarity between any two kinds of genre and the maximum value in the matrix is 6.

Now, the system converts the series of genres of recent songs into a series of similarity between neighbor genres using the similarity matrix. The series of similarity will be seen as the input for time series analysis method and we can estimate the next similarity. Then, the current genre and the estimated similarity will give us genre candidates.

Autoregressive Integrated Moving Average (ARIMA) [11] model is a general class of models in time series analysis. An ARIMA(p, d, q) model can be expressed by following polynomial factorization.

$$\Phi\left(B\right)\left(1-B\right)^d y_t = \delta + \Theta\left(B\right)\varepsilon_t \tag{2}$$

$$\Phi\left(B\right) = 1 - \sum_{i=1}^{p} \phi_i B^i \tag{3}$$

$$\Theta\left(B\right) = 1 + \sum_{i=1}^{q} \theta_i B^i \tag{4}$$

,where y_t is the tth value in the time series of data $\mathbf{Y}$ and B is the lag operator; ϕ and θ are the parameters of the model, which are calculated in analysis; p and q are orders of autoregressive process and moving average process, respectively; And d is a unitary root of multiplicity.

The first step of building ARIMA model is model identification, namely, estimating p, d and q by analyzing observations in time series. Model identification is beneficial to fit the different pattern of time series. The second step is to estimate parameters of the model. Then, the model can be applied to forecast the value at $t + \tau$, for $\tau > 0$. As an illustration consider forecasting the ARIMA$(1, 1, 1)$ process

$$\left(1-\phi B\right)\left(1-B\right) y_{t+\tau} = \left(1-\theta B\right)\varepsilon_{t+\tau} \tag{5}$$

$$\hat{\varepsilon}_t = y_t - \left[\delta + \sum_{i=1}^{p+d} \phi_i y_{t-i} - \sum_{i=1}^{q} \theta_i \hat{\varepsilon}_{t-i}\right] \tag{6}$$

Our system uses ARIMA to fit the series of similarity and to predict the next similarity. The process is shown in Figure 2.

We use Gaussian distributions to evaluate each possible genre for the next track. We select the one with the biggest probability.

3.2 Recording year

The recording year is similar to genre so we use ARIMA to predict the next possible year and compute the probability of a recording year.

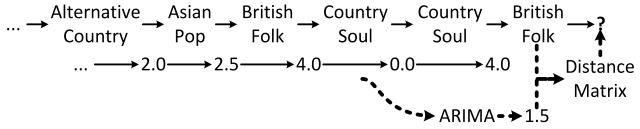

Figure 2. Predict the next genre

3.3 Freshness

As a new feature of this paper, we take into consideration freshness of a song to a user. Many recommendation systems [12] based on metadata of music and user behavior cannot avoid to recommend same music under same situations. As a result, a small set of songs will be recommended again and again. What's worse is that these songs will still be at the top of recommendation result since they have been recommended and played many times and then are seen as favorite songs. The iteration makes users fall into a "favorite trap" and feel bored. Therefore, an intelligent recommendation system should avoid to recommend same set of songs many times in a short period. On the other hand, the system is supposed to recommend some songs that have not been played for a long time because these songs are fresh to users even though they once listened to them multiple times.

Freshness can be considered as the strength of strangeness or the amount of experience forgotten. We apply Forgetting Curve [13] to evaluate the freshness of a song to a user. Forgetting Curve is shown as follows.

$$R = e^{-\frac{t}{S}}, \tag{7}$$

where R is memory retention, S is the relative strength of memory and t is time.

The lesser the amount of memory retention of a song in a user's mind, the fresher the song to the user. In our work, S is defined as the number of times the song has been played and t is the distance of present time to the last time the song was played. The reciprocal of memory retention is normalized to represent the freshness.

This metric contributes towards selecting fresh songs as recommendation results rather than recommending a small set of songs repetitively.

3.4 Favor

The strength of favor for a song plays an important role in recommendation. In playing songs, the system should give priority to user's favorite songs. User behavior can be implied to estimate how favored the user feels about the song based on a simple assumption: A user listens to a favorite song more often many an unfavorite song and on average listens to a larger fraction of the favorite song than the other.

We consider the favor of a song from four counts: active play times, passive play times, skip times and delete

times. Passive play time means the song is played as a recommendation result or as the next one in playlist. The favor is assessed by the weighted average of the four factors.

3.5 Time pattern

Since users have different habits or tastes in different periods of a day or a week, our recommendation system takes time pattern into consideration based on user log. The system records the time of the day and week that songs are played. It then employs Gaussian Mixture Model to estimate the probability of playing at a specific time. The playing times of a song in different periods trains the model using Expectation Maximization algorithm. When the system recommends songs, the model is used to estimate the probability of the song being played at that time.

3.6 Integrate into final score

A song is assessed whether it is a fit for recommendation as the next song from the five perspectives described in the above. In order to rank results and make a selection, the scores should be integrated into a final score. At first, the scores are normalized into the same scale. Since different users have different tastes, these five factors are assigned different weights at integration. Hence, we refer to Gradient Descent in order to match users' need. However, it is not user friendly to offer too many possible recommendation results and determine how to descent based on user's interaction. We use the recent recommendation results to adjust the weights, which is initialized by $(1.0, 1.0, 1.0, 1.0, 1.0)$. The algorithm is shown in Algorithm 1.

3.7 Cold start

Cold start is a difficult problem to tackle for recommendation system. When a recommendation system begins with no idea as to what kinds of songs users like or dislike, it hardly gives any valuable recommendation. As a result, in the cold start, the system randomly picks a song as the next song and records the user's interaction, which is similar to Pampalk's work [3]. After 16 songs has been played, the system uses the metadata of these songs and user behavior to recommend a song as the next one.

4. EXPERIMENT

The goal of the recommendation system is to cater to users' taste and recommend the next song at the right time and in the right order. Therefore, here, we focus on the user experience and compare users' satisfaction between our method and a baseline method, which randomly picks a song as the next one. We notice that most of the songs in a user's device are their favorite, but it doesn't mean that every song is fit to be played at anytime. The feedback to random selections represents the quality of songs in users' devices and the comparison result between our method and random selection shows the value of our method.

ALGORITHM 1: Adjust weights based on recent recommendation results

Input: Recent k recommendation results
$\Re_t\left(R_{t-k+1}, R_{t-k+2}, \ldots, R_{t-1}, R_t\right)$ at time t.
R_i contains user interaction of this recommendation χ_i, which is `like` or `dislike`, and the score of each factor of the recommendation i is $\mathbf{\Lambda}_i$.
Descent step δ, which is positive.
Current factor weights, $\mathbf{W}$.
Output: New factor weights, $\mathbf{W}'$.
Process:
if χ_t = `dislike` **then**
  Initialize an array $\mathbf{F}$ to record the contribution of each factor.
  for $i = R_{t-k+2}$ **to** R_t **do**
    $\Delta\mathbf{\Lambda}_i = \mathbf{\Lambda}_i - \mathbf{\Lambda}_{i-1}$
    $max = \arg\max_j \left(\Delta\lambda_j\right), 1 \le j \le 5$
    $min = \arg\min_j \left(\Delta\lambda_j\right)$
    if $\chi_i = Like$ **then**
      $\mathbf{F}_{max} = \mathbf{F}_{max} + 1$
    end
    else
      $\mathbf{F}_{max} = \mathbf{F}_{max} - 2$
      $\mathbf{F}_{min} = \mathbf{F}_{min} + 1$
    end
  end
  $inIndex = \arg\max_j \left(\mathbf{F}\right)$
  $w'_j = \begin{cases} w_j + \delta, j = inIndex \\ w_j - \delta/4, otherwise \end{cases}, j = 1, 2, 3, 4, 5$
  $deIndex = \arg\min_i \left(\mathbf{F}\right)$
  $w'_j = \begin{cases} w_j - \delta, j = deIndex \\ w_j + \delta/4, otherwise \end{cases}, j = 1, 2, 3, 4, 5$
end
else
  $\mathbf{W}' = \mathbf{W}$
end
return $\mathbf{W}'$

4.1 Data collection

An application system, named NextOne Player [1], is implemented to collect run-time data and user behavior for this experiment. It is developed in .NET Framework 4.0 using Windows Media Player Component 1.0. In addition to the functions of Windows Media Player, NextOne Player offers

[1] Available at http://sourceforge.net/projects/nextoneplayer/

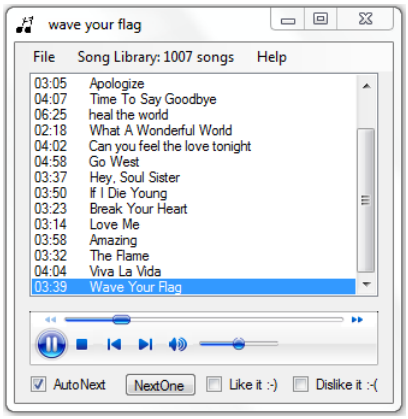

Figure 3. The appearance of NextOne Player

recommendation function using the approach described in Section 3 and also collects data for performance evaluation. The recommendation will work when the current song in the playlist ends or `NextOne` button is clicked. The appearance of the application is shown as Figure 3. The `Like it` and `Dislike it` buttons are used to collect user feedback. The proportion of a song played is recorded and viewed as the measure of satisfaction of a user for the song.

In order to compare our method with random selection, the player selects one of the two methods when it is loaded. The probability of running each method is set to 0.5. Everything is exactly the same except the recommendation method. In a contrasting experiment, users cannot realize which method is selected.

We have collected data from 11 volunteers. They consist of 9 graduate students and 2 professors and include 3 female students. They use the application in their devices which recommend songs from their own collections so the experiment is run on open datasets.

4.2 Results

First, we show the running time of recommendation function as it is known to have a major influence on the user experience. The running time results appear to be in an acceptable range. We run the recommendation system for different magnitudes of the song library and at each size the system recommends 32 times [2]. Figure 4 shows the variation in running time with the corresponding variations to the size of song library. We observe that the running time increases linearly with the increase in size of the song library. In order to provide a user-friendly experience, the recommendation results are processed near the end of the current song that is playing, and the result is generated when the next song begins.

[2] CPU: Intel i7, RAM: 4GB, OS: Windows 7

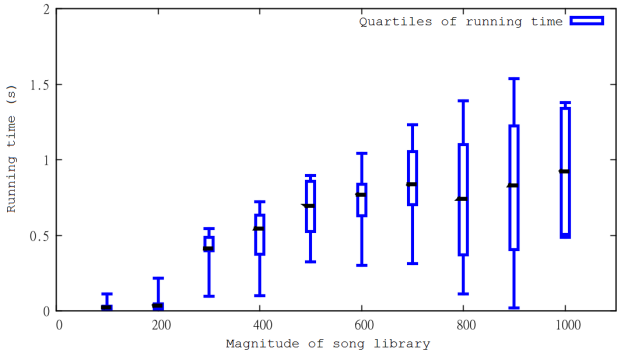

Figure 4. Running time of recommendation function

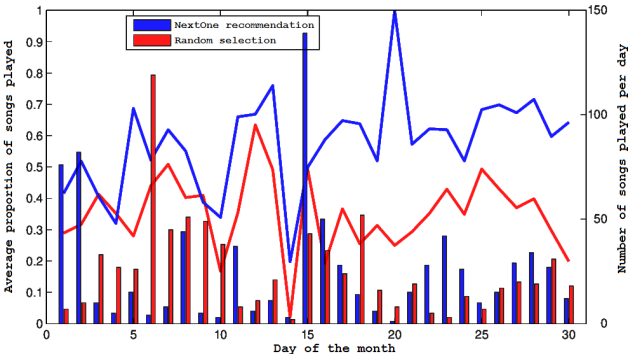

Figure 5. Representing the user logs to express favordness over a month

In order to evaluate the approach, the system records the playing behavior of the user. We collected the user logs from volunteers and calculated the average proportion of playing song length, which means how much partition of a song is played before it is skipped. Under the assumption that the partition implies the "favoredness" of the song for a user, we evaluate the recommendation approach by the partition as shown in Figure 5, where the histograms represent the number of songs that were played on a day. The curves in the graph represent the variation of the "playing proportion".

Moreover, continuous skips have a significant influence on the user experience, hence they can play an important role in evaluating the approach. A skip is defined as changing to the next track by the user before playing 5% of the length of the current track. The number of continuous skips can be used as a measure of user dissatisfaction. Figure 6 shows the distribution of continuous skips using our method and random selection.

From Figure 5 and 6, we can conclude that the recommendation approach surpasses the baseline and our recommendation is effective. Our approach is able to fit to a user's taste, and adjust the recommendation strategy quickly whenever user skips a song.

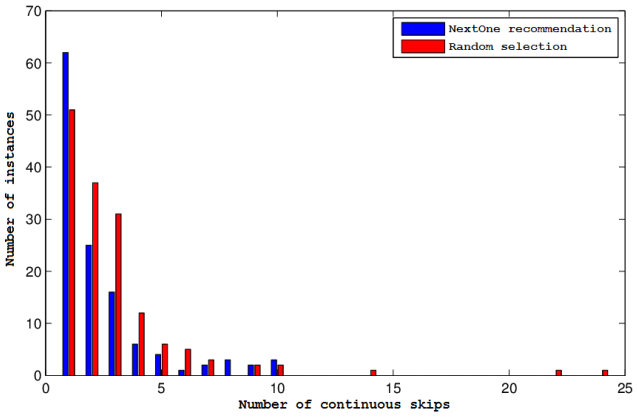

Figure 6. The distribution of continuous skips

5. CONCLUSION AND DISCUSSION

This paper presented a novel approach in recommending songs one by one based on user behavior. The approach considered genre, recording year, freshness, favor and time pattern as factors to recommend songs. The evaluation results demonstrate that the approach is effective.

In further research, we can apply this technique to a music database in a server. Also other users' behavior can be applied to recommend songs for a user. We can mix recommendation of music in a local device and an online server data to overcome the issue of cold start and hence obtain new favorite songs.

6. REFERENCES

[1] R. Ragno, C. Burges and C. Herley: "Inferring similarity between music objects with application to playlist generation," in *Proc. of 7th ACM Multimedia, Workshop on MIR*, pp. 73–80, 2005.

[2] A. Flexer, D. Schnitzer, M. Gasser and G. Widmer: "Playlist generation using start and end songs," in *Proc. of 9th ISMIR*, pp. 173–178, 2008.

[3] E. Pampalk, T. Pohle and G. Widmer: "Dynamic playlist generation based on skipping behavior" in *Proc. of 6th ISMIR*, pp. 634–637, 2005.

[4] W. W. Cohen and W. Fan: "Web-collaborative filtering: Recommending music by crawling the web," *Computer Network*, Vol. 33, pp. 685–698, 2000.

[5] P. Cano, M. Koppenberger and N. Wack: "An industrial-strength content-based music recommendation system," in *Proc. of 28th ACM SIGIR*, pp. 673, 2005.

[6] R. Cai, C. Zhang, C. Wang, L. Zhang and W. Ma: "MusicSense: Contextual music recommendation using emotional allocation modeling," in *Proc. of ACM Multimedia*, pp. 553–556, 2007.

[7] J. Donaldson: "A hybrid social-acoustic recommendation system for popular music," in *Proc. of the ACM Recommender Systems*, pp. 187–190, 2007.

[8] B. Shao, D. Wang, T. Li and M. Ogihara: "Music recommendation based on acoustic features and user access patterns," *IEEE Trans. on Audio, Speech And Language Processing*, Vol. 17, No. 8, pp. 1602–1611, 2009.

[9] N. Liu, S. Lai, C. Chen and S. Hsieh: "Adaptive music recommendation based on user behavior in time slot," *International Journal of Computer Science and Network Security*, Vol. 9, pp. 219–227, 2009.

[10] J. Su and H. Yeh: "Music recommendation using content and context information mining," *IEEE Intelligent Systems*, Vol. 25, pp. 16–26, 2010.

[11] G. E. P. Box, and D. A. Pierce: "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Jour. of the American Statistical Association*, Vol. 65, pp. 1509–1526, 1970.

[12] B. Logan: "Music recommendation from song sets," in *Proc. of 5th ISMIR*, pp. 425–428, 2004.

[13] H. Ebbinghaus: *Memory: A Contribution to Experimental Psychology*, Columbia University, New York, 1913.