International Journal of Multimedia and Ubiquitous Engineering
Vol.8, No.6 (2013), pp.87-96
http://dx.doi.org/10.14257/ijmue.2013.8.6.09

An Adaptive Algorithm to Recommend Favorable Digital Music

Taek Lee and Hoh Peter In*

Korea University
comtaek@korea.ac.kr, hohin@korea.ac.kr

Abstract

Many people enjoy digital music (e.g., MP3 songs), usually with random play mode, or their own favorable play list that they have composed. However, such play modes do not consider and support listener preferences of feeling or mood changing with time. Usually listeners have dynamic, not static, demands on music based on their arbitrary situation or mood (e.g., when studying, exercising, being sorrowful, being happy, etc.), so an adaptive algorithm to meet the momentary demand is required. This paper proposes an adaptive algorithm to recommend favorable songs successively, and enable people to seamlessly keep listening to favorable songs, without the action of skipping disliked ones. The algorithm monitors if a listener likes or dislikes a song currently being played. Once the algorithm detects that a listener likes the song, the algorithm recommends the next song that is most similar to the current song. Otherwise, the algorithm recommends quite a different style of a song as the next one, by recognizing that the listener now has a different demand. In our experiment, the proposed algorithm showed better performance, in terms of reducing the action of frequently skipping songs, than random play mode, with statistical significance.

Keywords: *MP3, digital audio, recommender system, adaptive algorithm, user preference*

1. Introduction

People enjoy listening to digital audio (*e.g.*, MP3 songs) under many situations, such as while exercising, studying, eating, relaxing, and so on. People tend to enjoy different music styles, depending not only on the physical situations, but on emotional situations, such as when being sorrowful, being happy, or being under stress. For different demands, people usually make their own favorable play lists in advance, and choose a proper list for playing music, or just simply use random play mode for the songs they possess.

Random play mode or a play list for preferred songs is a simple way to easily enjoy music. However, a problem is that making a preferred play list is effort-consuming if the number of songs is huge, and moreover, the preference is often not static, but dynamic, depending on the listener's feeling or mood at a moment, or depending on what the listener is then doing, as mentioned above. Random play does not react to the dynamic demand, or the listener's current emotion and mood, and a pre-composed play list also does not always meet listeners' preferences, which arbitrarily change from time to time. Therefore, an adaptive algorithm is required to meet the dynamics.

* Hoh Peter In is the corresponding author.

For example, supposed that a listener is now happy and satisfied with listening to a classic music currently being played. A loud song of rock genre, suggested and played as the next song by random play mode, suddenly interferes. The suggestion will make the listener really embarrassed and disrupted, so she will skip it right away. Our research hypothesis is that once a listener is into any style of song, she tends to keep listening to similar songs for a while.

Our proposed adaptive algorithm aims to recommend favorable songs successively, with minimizing the effort of skipping disliked ones, while listening to a given pool of songs. The algorithm monitors if a listener likes or dislikes a suggested song. Once the algorithm detects that the listener likes the suggested song, the algorithm adaptively searches the most similar style of a song from the pool, and then recommends the searched one as the next song to play. However, if the algorithm detects that the listener does not like the currently suggested song, it recommends the next song that has quite a different style (dissimilar), in order to search for any other favorable one. To compute and compare similarity of songs, we used four features digital songs inherently have in their metadata (*i.e.*, MP3 ID3 tag): genre, artist, tempo, and lyric. With the four kinds of features, the algorithm defines and distinguishes the style of songs. Genre is the most and representative feature to distinguish music, and the same genre of songs are differentiated even by artists. In addition, tempo (beat per minute) is another nice feature to depict slow or fast music. Lastly, lyrics can describe the semantic feeling of songs, even though the former three features of genre, artist, and tempo are hard to deliver such a semantic.

To verify the performance of our proposed algorithm, we conducted an experiment of a blind test with 29 subjects, by using our implemented MP3 player. They were asked to listen to given MP3 songs, but they do not know whether they are currently listening to the songs in the conventional mode of random play, or in the mode of our proposed adaptive algorithm; that is, there are two groups, the experimental group and control group. Our research question is to know which group of people keeps listening to songs successively and conveniently, without the effort of song skipping actions. To show outperformance of our proposed algorithm, we collected experimental data, tested our research hypothesis, and finally conclude it works effectively as expected, with statistical significance (Section 3.2).

The rest of the paper consists of the following sections: Section 2 explains our proposed algorithm, Section 3 explains the experiment setup and the result analysis, Section 4 introduces some related work, and finally Section 5 concludes the paper, with remarks about the meaning of the paper, and the future work in brief.

2. The Proposed Algorithm

The proposed algorithm aims to recommend songs that might be favorable to listeners based on our hypothesis: once a listener likes any style of music, and is into that style at any moment, she wants to keep listening to a similar style of music for a while. Of course, the preferred style might change from time to time. Therefore, the overall recommendation process of preference monitoring and searching for similar songs must continuously work in real-time.

The overall process of how our proposed algorithm works is shown in Figure 1. The algorithm consists of two parts of steps: (a) one part of the prerequisite processing steps, and (b) the other part of the song recommendation steps.

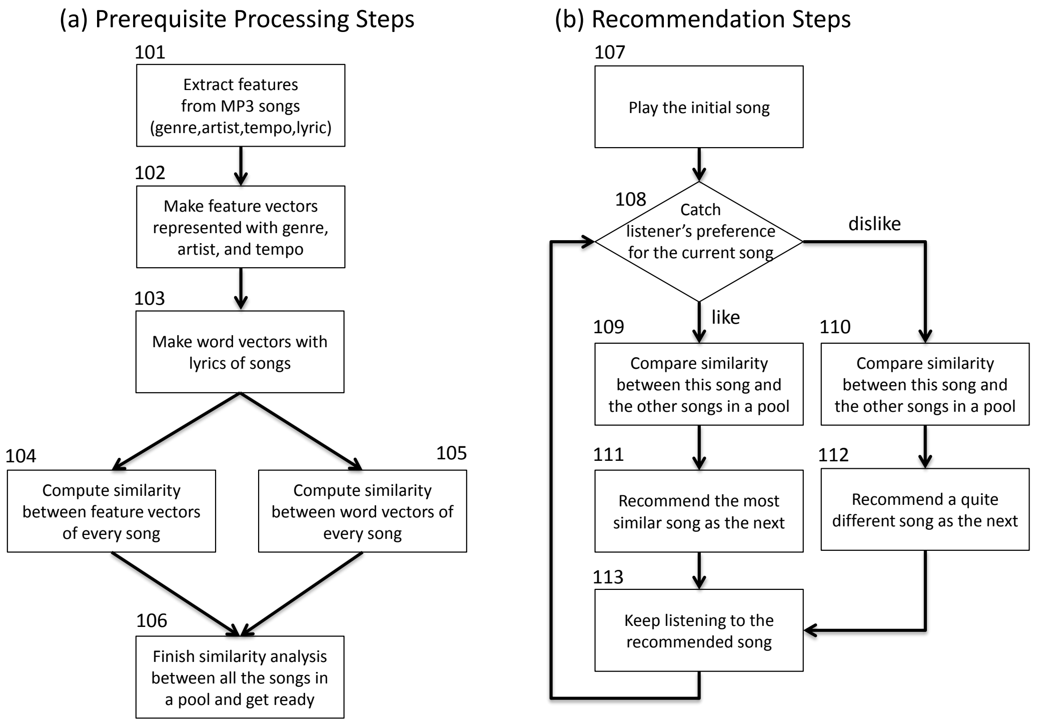

(a) Prerequisite Processing Steps
101
Extract features from MP3 songs (genre,artist,tempo,lyric)
102
Make feature vectors represented with genre, artist, and tempo
103
Make word vectors with lyrics of songs
104
Compute similarity between feature vectors of every song
105
Compute similarity between word vectors of every song
106
Finish similarity analysis between all the songs in a pool and get ready
(b) Recommendation Steps
107
Play the initial song
108
Catch listener's preference for the current song
dislike
like
109
Compare similarity between this song and the other songs in a pool
110
Compare similarity between this song and the other songs in a pool
111
Recommend the most similar song as the next
112
Recommend a quite different song as the next
113
Keep listening to the recommended song

Figure 1. An overview of our proposed algorithm

Firstly as shown in part (a) in Figure 1, some preprocessing steps are required to get ready for recommending songs. The steps aim at reducing heavy computation, which can be problematic in real-time service. Computing similarity between a large number of all the songs in real-time and every time is very inefficient and redundant, so it is better for similarity analysis to be done in advance, before the steps of similarity comparison (109 and 110 in Figure 1).

Secondly, as shown in part (b) in Figure 1, the recommendation steps are processed, while listeners are listening to songs in real-time. These steps are again grouped into two main processing sub parts: one is monitoring the listener's preference for a suggested song, and the other one is suggesting the next song, by searching for a (dis)similar song in the rest of the songs in a pool.

2.1. Part I: prerequisite processing steps

The overview is shown in (a) prerequisite processing steps of Figure 1. This part of the steps is required to run through just one time, and is not required again later. At the beginning, features are extracted from the metadata of MP3 song files **(101 in Figure 1)**, and then the features are represented in the form of vectors **(102 in Figure 1),** because similarity computation uses vector expression. Basically, genre, artist and lyric among the features have text format initially except tempo, so they need to be transformed to numeric values first.

We used the official list of genres in ID3 tags[1] to distinguish 149 different genres. By the way, the identifier number (0~148) has no meaning relating to its higher or lower number; the number is just a code to identify genres. From our application point of view, however, we need to convert the identifier numbers into a meaningful ordinal scale; it should mean slower

[1] List of genres: http://www.multimediasoft.com/amp3dj/help/amp3dj_00003e.htm#ss13.3

and calmer music like classic or blues as the number is closer to 0, in contrast it should mean faster and heavier music like rock or metal as the number is closer to 148. With that meaning, we rearranged the identifier numbers.

To convert artist names into an ordinal scale as genre, the text names of artists are sequentially encoded with numbers, but the numbers should have ordinal meaning. As the number is close to 1, it means the artist tends to present slower and calmer music. In contrast, as the number is further away from 1 (getting higher and closer to the maximum number), the artist tends to present a faster and heavier style of music. By analyzing every genre per artist for the given pool of songs, the encoding of artist names is done. In this way, all the artist names are converted into an ordinal scale, from 1 to the number of distinct artists observed in the pool of songs. Finally, the feature vector consisting of genre, artist, and tempo has a 3-dimensional format like (f_{genre}, f_{artist}, f_{tempo}), where tempo does not need to be converted, because it is inherently numeric **(102 in Figure 1)**.

Every song has a feature vector (f_{genre}, f_{artist}, f_{tempo}), so we use the Euclidean distance equation [1] to compute the similarity between two different songs with feature vectors **(104 in Figure 1)** as follows:

$$distance(song_1, song_2) = \sqrt{\left(f_{genre}^{song_1} - f_{genre}^{song_2}\right)^2 + \left(f_{artist}^{song_1} - f_{artist}^{song_2}\right)^2 + \left(f_{tempo}^{song_1} - f_{tempo}^{song_2}\right)^2}$$

$$similarity_{euc}(song_1, song_2) = 1 - \frac{distance(song_1, song_2)}{maximum\ distance} \qquad (1)$$

The Euclidean distance has no upper bound, so it is transformed into the normalized similarity measure of range from 0 to 1, *similarity*$_{euc}$(*song*$_1$, *song*$_2$).

In contrast to the feature vector (f_{genre}, f_{artist}, f_{tempo}), the lyric features of songs have different vector format in our algorithm, because lyrics consists of many words, not a single text attribute. Therefore, we convert the words consisting of a lyric document into the format of a word vector, instead of a feature vector. If lyrics of songs are considered as documents, a document consists of many words, so we simply used the Bag-of-Word model [2] to count and express the distribution of words consisting of lyric documents. For example, suppose that one song A has the lyric of "*I love you*", and the other song B has that of "*You know I love you so much*". In that case, the word vector of the song A becomes (I, love, you, know, so, much) = (1, 1, 1, 0, 0, 0), and the word vector of the song B becomes (I, love, you, know, so, much) = (1, 1, 2, 1, 1, 1). With the word vector expression of the Bag-of-Word model **(103 in Figure 1)**, the similarity between different lyrics of songs can be compared.

However, one problem exists when adopting the Bag-of-Word model; the dimension size of the word vector explosively increases, and most of the attributes are zeros (no-counted words), as the number of lyric words from many songs increases. Even though documents have hundreds or thousands of words, each document is sparse, since it has relatively few non-zero words. Thus, similarity should not depend on the number of shared 0 values, since any two documents are likely to not contain many of the same words, and therefore, if 0-0 matches are counted, most documents will be highly similar to most other documents. Therefore, we use cosine similarity to address this problem, which is one of the most popular measures of document similarity [1] **(105 in Figure 1)**.

$$similarity_{cos}(song_1, song_2) = \cos(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \, \|v_2\|} \quad (2)$$

where, v_1 and v_2 are the word vectors of $song_1$ and $song_2$ respectively, and • indicates the vector dot product, $v_1 \cdot v_2 = \sum_{k=1}^{n} v_{1k} \cdot v_{2k}$ (k is the number of word attributes). $\|v\|$ is the length of vector v, $\|v\| = \sqrt{\sum_{k=1}^{n} v_k^2} = \sqrt{v \cdot v}$. Cosine similarity has a range number between 0 and 1; it means dissimilar as the measure is close to 0, and it means similar as the measure it close to 1.

The two types of similarity measures computed from the steps **104 and 105 in Figure 1** have the range from 0 to 1 respectively. Thus, we combine them, to get a finalized similarity measure as follows:

$$\begin{aligned} &similarity(song_1, song_2) \\ &\quad = \alpha \cdot similarity_{euc}(song_1, song_2) + \beta \cdot similarity_{cos}(song_1, song_2) \quad (3) \end{aligned}$$

where, a and β are weight coefficients so the total of a and β is one; we simply give 0.5 and 0.5 to a and β respectively in the paper. Of course, some classical or instrumental music has no lyrics, so the term $similarity_{cos}$ cannot be used in the similarity computation of the equation (3). In that case, the equation (3) is adopted, by just adjusting a as one and β as zero.

2.2. Part II: recommendation steps

This part of the steps refers to the similarity measures computed in Section 2.1 for the purpose of song recommendation. The algorithm begins with an arbitrary initial song **(107 in Figure 1)**. By understanding how long a time a listener has enjoyed the current song, the algorithm decides whether the listener likes the song, or not **(108 in Figure 1)**. In our experiment, we asked subjects to click the button ‘Next’ in the MP3 player if they do not like the current song, and want to skip it. By taking advantage of analyzing the click logs, we had figured out that most of the listener’s decision making for preference is done very quickly, at the beginning phase of songs; this fact is justified by Figure 3. Thus, our algorithm understands that a listener does not like a song, if the play time (the elapsed time before clicking the button ‘Next’) is less than 50% of the length of the song. In contrast, the algorithm understands that a listener likes a song, once she listens to the song during the play time of more than 50%.

Once the algorithm concludes that the listener likes the current song, the algorithm searches for the most similar song, by comparing the similarity measures of every song already computed in Section 2.1 **(109 and 111 in Figure 1)**. However, if the algorithm concludes that the listener does not like the current song, the algorithm firstly gathers half of the song candidates from the song pool, whose similarity gap is furthest away from the current song, and then picks up one song from the candidates, as the next song to suggest **(110 and 112 in Figure 1)**.

All the cycle described above is rotated, while listeners want to keep enjoying all the songs **(113 in Figure 1)**.

3. Experiment and Result

We conducted an experiment to verify our research question mentioned in Section 1. This section explains our experiment setup, and the result of experiment analysis.

3.1. Experiment setup

We implemented our proposed algorithm with a web-based prototype of the MP3 player, for easy access of subjects to our experimental setup. We aggregated an anonymous 29 subjects who are willing to join in our experiment enjoying the MP3 player. They are asked to click the button 'Next', depending on their preference. We used 562 genre-balanced songs in the experiment. The web server provides the MP3 player service, and at the same time records logs of the play time of songs; that is, how long a time subjects listened to songs, before clicking the Next button. We conducted a blind test with the subjects. To avoid any bias, they are randomly split into two groups: an experimental group (*i.e.* our proposed adaptive algorithm), and a control group (*i.e.* random play mode).

3.2. Result analysis

First of all, we surveyed the distribution of actions of clicking the Next button over time, by analyzing logs of the play time. Figure 2 shows the distribution; the x axis is the percentage of play time, and the y axis is the percentage of listeners. As shown in Figure 2, many listeners decide quickly (before 10% listening) whether they will skip the song or not, but once they decide to keep listening to the song, they almost always listen to the song until the end of song (more than 95% in length). This supports that the threshold of 50% play time used in our proposed algorithm (108 in Figure 1) is a reasonable decision making point, to decide whether a listener liked a song or not.

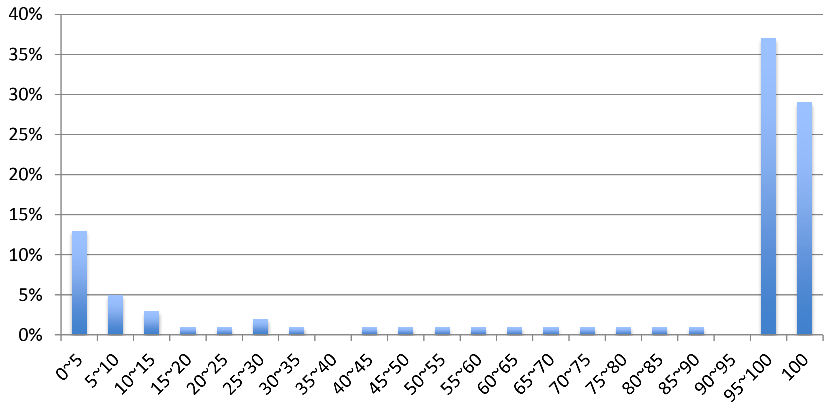
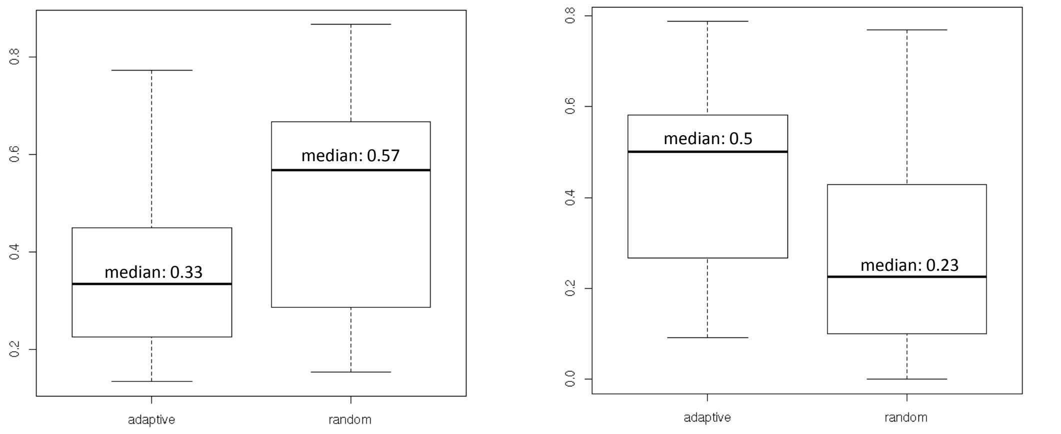


Figure 2. The distribution of clicking the next button over play time

From the blind test with subjects, we concluded that our algorithm works effectively as expected; with little effort of skipping songs, subjects enjoyed conveniently listening to their favorable songs. To measure the performance of our proposed algorithm, we prepared for two evaluation criteria: one criterion is a measure of how many songs listeners disliked (*i.e.*, clicked the Next button) among all their played songs (criterion 1 in Figure 3), and the other criterion is a measure of how many songs were successively played one after another, without any intermediate skipping action, among all their played songs (criterion2 in Figure 3).



$$\text{(a) criterion1} = \frac{\text{\# of skipped songs}}{\text{\# of played songs}}$$
(Wilcox.test, p=0.04591)

$$\text{(b) criterion2} = \frac{\text{\# of songs succesively listened}}{\text{\# of played songs}}$$
(Wilcox.test, p=0.04583)

Figure 3. A performance comparison of random play and the proposed adaptive algorithm

The boxplots of Figure 3 show the evaluation results from subjects. For the criterion 1 ((a) left-hand side of boxplots), Figure 3 shows that subjects skipped 57% of songs among their played songs in the random play mode. In contrast, however, subjects skipped only 33% of songs among their played songs in our proposed adaptive algorithm. In the boxplot of the distribution, the variation of skipping actions was also narrower in the case of our algorithm, than that of the random play mode. To test the statistical significance of the result analysis, we conducted a Wilcoxon Rank-Sum test [3]. Since we cannot assume the exact distribution of samples, and we have a relatively small number of samples, the Wilcoxon Rank-Sum test was applied to our hypothesis test. With 95% of confidence level, we could confirm ($p=0.04591<0.05$) that the null hypothesis is rejected (N_0: there is no difference in medians between ‘adaptive’ and ‘random’ experiments), and instead the alternative hypothesis is accepted (the approach of ‘adaptive’ has a lower median).

The evaluation measure of criterion2 is more important than criterion1, because criterion 2 can verify our main research hypothesis (assumption) -- once a listener likes any style of a song, she will want to listen to a similar style of songs for a while. As shown in criterion 2 of Figure 3 ((b) right-hand side of boxplots), subjects liked to listen to similar songs successively, as suggested by our proposed adaptive algorithm. If they had not liked the similar style of suggested songs, they would have frequently clicked the Next button. The number of clicking actions in our proposed algorithm was surely less than that of the random play mode, so that in other words the indicators of enjoying successive songs without skipping is higher (‘adaptive’ = median 0.5 > ‘random’ = median 0.23). Besides, the performance gap was more than 200% (0.5 vs 0.23). Lastly, we also applied the Wilcoxon Rank-Sum test to the criterion2 evaluation, as done for criterion1, and could conclude that our proposed algorithm still outperformed the random play mode in terms of criterion2, with statistical significance ($p=0.04583<0.05$, confidence level = 95%).

4. Related Work

Music recommendation technology is referred to with the term of music information retrieval (MIR)[2] in the literature. For recommendation purpose, MIR technologies mainly rely on two sources of information: one is content-based information coming from the music file itself, by signal processing, such as loudness, tempo, rhythm, and melody. The other is metadata-based information, coming from the metadata of music described by human, such as title, artist, and user's social review or tags [4].

Each information source has its own pros and cons. Metadata-based information is nowadays not just only dependent on the metadata recorded in the music files, but also on social preference data. This is called collaborative filtering (CF) [5]. To discover a user's preference, CF uses metadata collaboratively aggregated from music consumers, such as users' metadata editing, social tags, and statistics about music consumption patterns, such as play counts, music chart of artists, and sales revenue. The working mechanism of CF is simple. Supposed that a user X likes a song A, and there is information that many other people who enjoyed the song A simultaneously liked a song B in their play list. If so, the recommender system suggests the user X listen to the song B, as well. For example, some representative commercial services taking advantage of the CF technology are *Last.fm* and *iTunes Genius*.

However, a problem in using CF is that it relies on large data from many people, and tends to only recommend publicly popular music. Major songs frequently listened to by many people have abundant information about surveyed preference, but the minor songs not frequently listened to by people usually do not have sufficient information of preference. As a result, it is easy for a minor song to be missed in recommendation, even if the song is very wonderful and favorable song to a listener. This problematic situation happens,[3] since CF relies on the "wisdom of crowds".

For that reason, content-based information is sometimes a better source of music recommendation, since it does not at least fall to the defect of the "wisdom of crowds". However, content-based information is not enough in itself for an effective recommendation service, but is good for music classification, if users want to classify music in various styles, and pick up one at their demand [7, 8, 9].

Our proposed algorithm uses metadata-based information for music recommendation, but does not adopt the popular CF approach. Rather, the proposed algorithm focuses on personal preference, because we believe that the preference dynamically changes, and the demand of a user is not always same as that of the public. In addition, some users probably do not want to share their preference data with the public, because of privacy reasons.

5. Conclusion

In this paper, we proposed an adaptive algorithm to search and recommend a digital music that listeners will be happy to enjoy, whose demand is however not well met by the conventional random play mode, or by means of using pre-composed play lists. To

[2] MIR: http://en.wikipedia.org/wiki/Music_information_retrieval
[3] The problem is called the long-tail and cold-start problems in the literature [6]

recommend a (dis)similar song, we used the information of four distinct features, which any digital music files universally have, namely genre, tempo (beat per minute), artist, and lyrics. The source of features is from standard metadata (*e.g.*, MP3 ID3 tags) embedded in digital music files, so our approach is easily applicable to any usual digital songs. Lack of the metadata does not matter, because there are many available helper tools automatically feeding the required metadata to digital music files. In addition, the proposed algorithm does not require any extra data about listeners' song preference for model training purpose, as machine learning algorithms usually require. Nevertheless, without relying on such model training data, our simple algorithm substantially searches and recommends favorable songs in an adaptive way, in real-time. We believe our algorithm will be helpful, especially when quickly searching many favorable songs from a large size pool of songs, which have never even been listened to before. In the future, we will improve our algorithm towards location-aware, time-aware, and user behavior-aware recommender system.

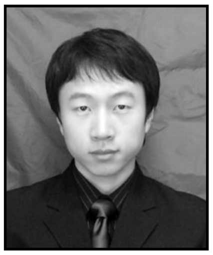
Acknowledgements

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012M3C4A7033345) and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2009021).

References

[1] P. Tan, M. Steinbach and V. Kumar, "Introduction to Data Mining", Addison Wesley Publishers, **(2006)**.
[2] S. Scott and S. Matwin, Proc. 16th Int'l Conf. Machine Learning, **(1999)**, pp. 379-388.
[3] R. E. Walpole, R. Mers and S. L. Myers, "Probability & Statistics for Engineers & Scientists", Pearson Prentice Hall, **(2006)**.
[4] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang and X. He, "ACM Transactions on Multimedia Computing", Communications, and Applications (TOMCCAP), vol. 7S, Issue 1, **(2011)** October.
[5] S. Bourke, M. P. O'Mahany, R. Rafter, K. McCarthy and B. Smyth, "Research and Development in Intelligent Systems XXIX", **(2012)**.
[6] O. Celma, PhD thesis, UPF, Barcelona, Spain, **(2008).**
[7] D. Bogdanov, J. Serr`a, N. Wack, P. Herrera and X. Serra, IEEE Transactions on Multimedia, vol. 13, no. 4, **(2011)**, pp. 687–701.
[8] T. Magno and C. Sable, International Conference on Music Information Retrieval (ISMIR'08), **(2008)**, pp. 161–166.
[9] T. Pohle, D. Schnitzer, M. Schedl, P. Knees and G. Widmer, International Society for Music Information Retrieval Conf. (ISMIR'09), **(2009)**, pp. 525–530.

Authors



Taek Lee

He is currently a Ph. D. candidate in Computer Science and Engineering at Korea University in Seoul, Korea. He received his MSc in Computer Science and Engineering at Korea University in 2006. His research interests include man-machine interaction, user behavior modeling in software systems, software defect prediction, information security, and information risk analysis.

Hoh Peter In

He received his Ph.D. degree in Computer Science from the University of Southern California (USC). He was an Assistant Professor at Texas A&M University. At present, he is a professor in Department of Computer Science and Engineering at Korea University in Seoul, Korea. He is an editor of the EMSE and TIIS journals. His primary research interests are software engineering, social media platform and services, and software security management. He earned the most influential paper award for 10 years in ICRE 2006. He has published over 100 research papers.