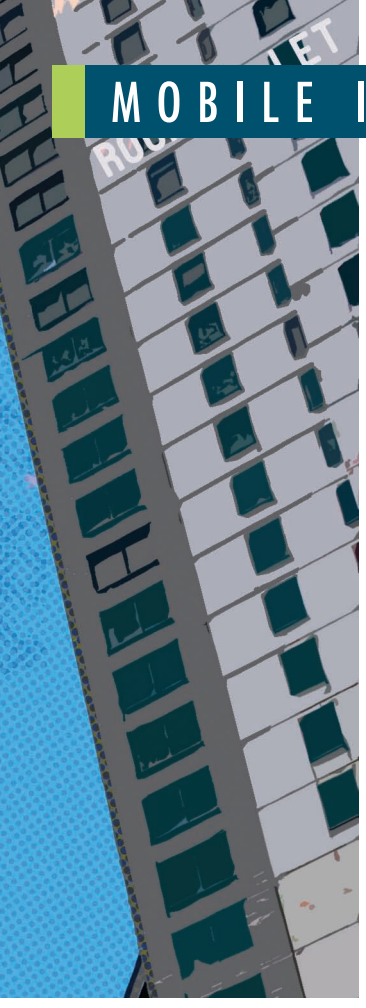

Music Recommendation Using Content and Context Information Mining

Ja-Hwung Su and Hsin-Ho Yeh, *National Cheng Kung University, Taiwan*

Philip S. Yu, *University of Illinois at Chicago*

Vincent S. Tseng, *National Cheng Kung University, Taiwan*

To offer music recommendations that suit the listener and the situation, uMender mines context information and musical content and then considers relevant user ratings.

Mobile devices such as smart phones are becoming popular, and real-time access to multimedia data in different environments is getting easier. With properly equipped communication services, users can easily obtain the widely distributed videos, music, and documents they want. Because of its usability and capacity requirements, music is more popular than other types of multimedia data. Documents and videos are difficult to view on mobile phones' small screens, and videos' large data size results in high overhead for retrieval. But advanced compression techniques for music reduce the required storage space significantly and make the circulation of music data easier. This means that users can capture their favorite music directly from the Web without going to music stores. Accordingly, helping users find music they like in a large archive has become an attractive but challenging issue over the past few years.

Traditional music recommenders have been based primarily on collaborative filtering (CF). (See "Related Work in Collaborative Filtering" for more information on this technique.) But their effectiveness has been limited by insufficient information, including sparse rating data and a lack of contextual information. Sparse rating data occurs frequently in real applications and can result in a distorted recommendation list. In addition, a user's preferences can vary in different contexts, such as location, time, movement state, and temperature. For example, someone jogging might prefer hip-hop to classical music. A survey showed that activity (a type of context information) significantly affects a listener's mood.[1] This finding delivers an important message that context information is an important element for a music recommender to consider in selecting music to suit the listener's mood. The next generation

1541-1672/10/$26.00 © 2010 IEEE
Published by the IEEE Computer Society

Related Work in Collaborative Filtering

In the music recommendation field, traditional approaches such as collaborative filtering (CF) are used to discover the relations between users and music items on the basis of the rating a user gives for each music item. (We use the term *music item* in this article to denote a musical piece such as a song.) There are several types of CF-like recommenders.

To establish a bridge between the user's interests and music items, *model-based recommenders* use machine-learning techniques to represent user preferences by a set of rating scores and to construct a special prediction model.[1] Motivated by model-based CF, several studies over the past few years have focused on the effectiveness of modeling behaviors.

In contrast to model-based recommenders, *memory-based recommenders* find an active user's nearest neighbors using a massive number of explicit user votes.[2-4] Then they aggregate the found neighbors' ratings to predict the active user's ratings of relevant items.

In addition to these CF-based methodologies, some past studies have developed *hybrid recommenders,* which combine different CF algorithms to produce a better recommendation list.[5] Jun Wang, Arjen de Vries, and Marcel Reinders aggregated user-based and item-based CF approaches and showed that hybrid methods outperform individual approaches.[4]

The major idea behind CF is that products like books, music, movies, and so on can be bridged to preferences through users with similar rating behaviors. Unfortunately, CF-like recommenders suffer from insufficient information, especially from a lack of contextual information. Although some contemporary context-aware recommenders attempt to enhance recommendations with more considerations of environmental metadata, they still have trouble learning the user's preferences under different contextual conditions.[6,7]

References

1. G. Adomavicius and E. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 6, 2005, pp. 734–749.
2. R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, 2002, pp. 331–370.
3. B.M. Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l Conf. World Wide Web* (WWW 01), ACM Press, 2001, pp. 285–295.
4. J. Wang, A.P. de Vries, and M.J.T. Reinders, "Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion," *Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval* (SIGIR 06), ACM Press, 2006, pp. 501–508.
5. K. Yoshii et al., "Hybrid Collaborative and Content-Based Music Recommendation Using Probabilistic Model with Latent User Preferences," *Proc. 7th Int'l Conf. Music Information Retrieval* (ISMIR 06), Int'l Soc. Music Information Retrieval, 2006, pp. 296–301.
6. J. Hong et al., "Context-Aware System for Proactive Personalized Service Based on Context History," *Expert Systems with Applications*, vol. 36, no. 4, 2009, pp. 7448–7457.
7. G. Reynolds et al., "Interacting with Large Music Collections: Towards the Use of Environmental Metadata," *Proc. IEEE Int'l Conf. on Multimedia and Expo*, IEEE Press, 2008, pp. 989–992.

of mobile phones might provide such context information. Hence, ubiquitous music recommendation, shown in Figure 1, has recently received considerable attention.

In this article, we propose the Ubiquitous Music Recommender (uMender), which tackles the music recommendation problem by mining musical content and context information. For musical content, we propose a two-stage clustering approach that identifies each music item's *perceptual patterns*, which consist of acoustical and temporal features hidden in music. Considering such patterns lets us capture a user's listening interest in music items more precisely. For context information, we have designed a novel solution for capturing a user's preferences under various context conditions. Instead of computing the correlation coefficients of user ratings, as in traditional methods, we group users under similar context conditions to find implicit, more applicable perceptual patterns. Through integrated mining of both context information and musical content, uMender

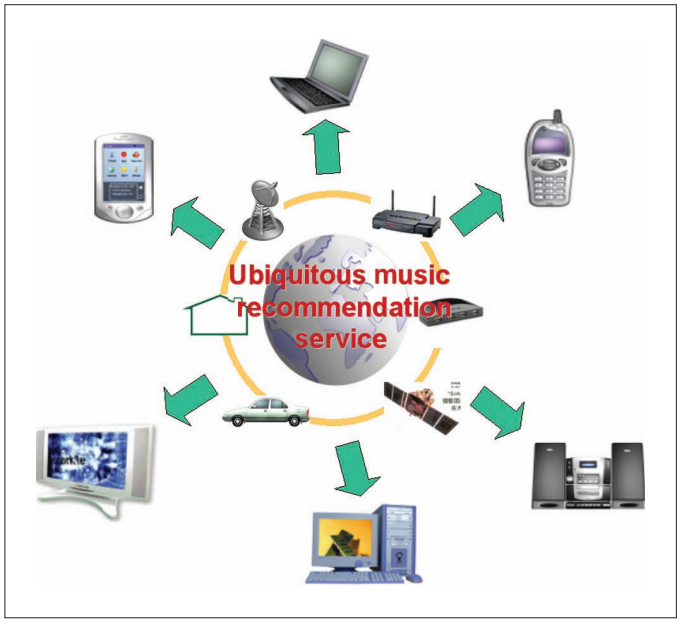

Figure 1. Scenario for ubiquitous music recommendation.

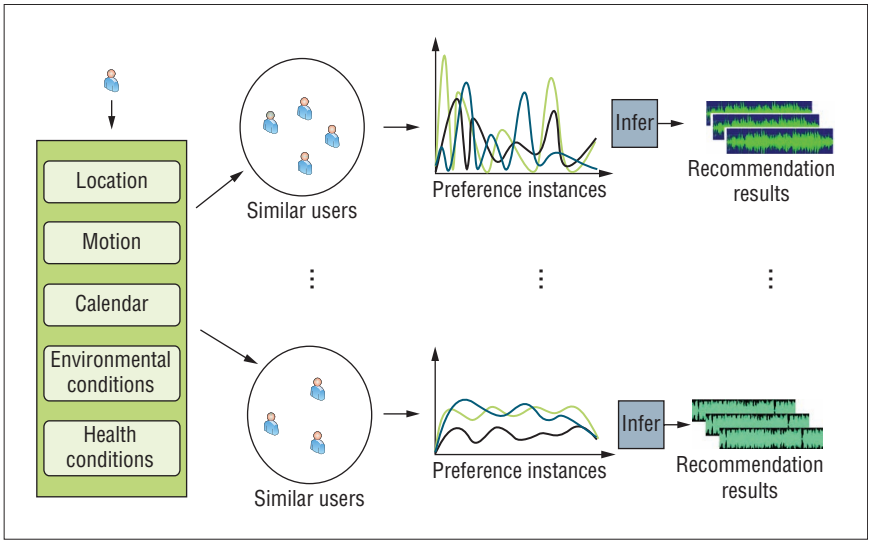

Figure 2. Main idea of the Ubiquitous Music Recommender. uMender predicts the user's preference for music by discovering the user's listening interests from musical content and context information.

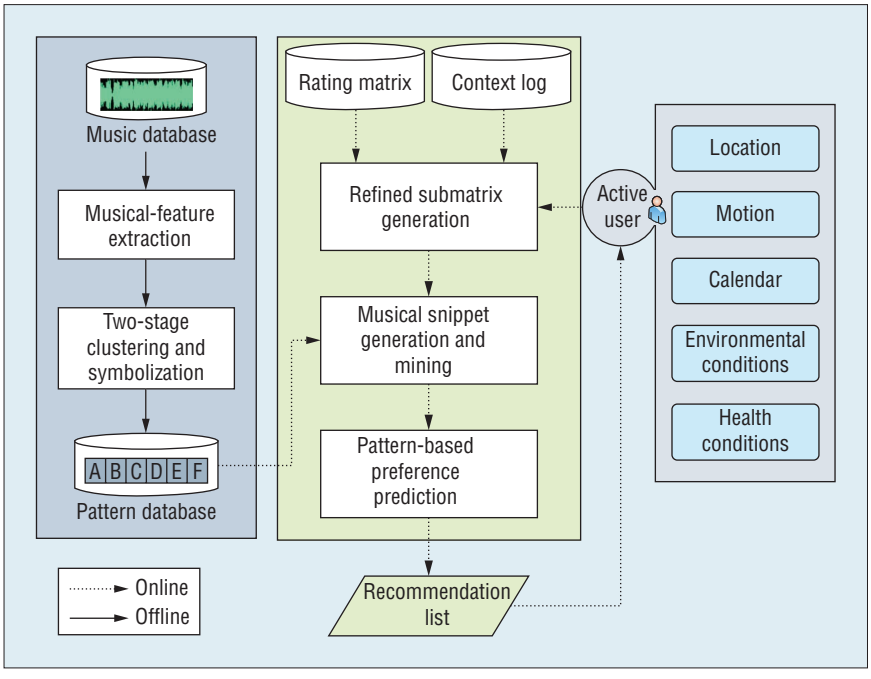

Figure 3. Workflow of uMender. The system mines the perceptual patterns for the user's interests in the offline preprocessing stage to facilitate online preference prediction on the basis of context information.

provides appropriate, ubiquitous music recommendations.

Conceptual Basis

The main challenge of ubiquitous music recommendation is how to use contextual information and musical content to effectively discover a user's implicit interests. To deal with this challenge, we explored mining music's perceptual patterns on the basis of available contextual information.

Figure 2 shows the main idea of our proposed recommender. Basically, our system represents a music item as a sequence of transformed symbols that indicate the music item's signatures. On the basis of the CF principle, uMender groups users with similar contexts, implying similar interests in music. Then, uMender identifies relevant music items that match both the user's listening interests and the current context. uMender is an innovative recommender that integrates musical content mining and context information filtering to achieve high-quality ubiquitous music recommendations.

Framework

As shown in Figure 3, the uMender framework consists of two phases: offline preprocessing and online prediction. During offline preprocessing, the system mines the perceptual patterns for the user's listening interests, which facilitates online preference prediction on the basis of context information.

Offline Preprocessing

The emphasis of this phase is music preprocessing, which includes feature extraction, two-stage clustering, and music symbolization. Because the prediction we propose is based on musical features, this phase is important to online prediction. Our system first extracts the musical features from the music in the database. Next, through two-stage clustering, it converts each music item into a consecutive symbolic string (also called a perceptual-pattern string) composed of a set of perceptual patterns.

Online Prediction

When an active user u_i arrives, the system searches the rating matrix and context log to produce a refined submatrix that represents the most-relevant users and music items with

similar context information. Then, it predicts the rating of each music item, one by one. For each music item, the system encodes a set of consecutive patterns as a snippet in a window-by-window manner. According to the matching snippets mined from the positive and negative sets of the refined submatrix, the system can use pattern-based preference prediction to determine preferred music items. Finally, the system generates the recommendation list.

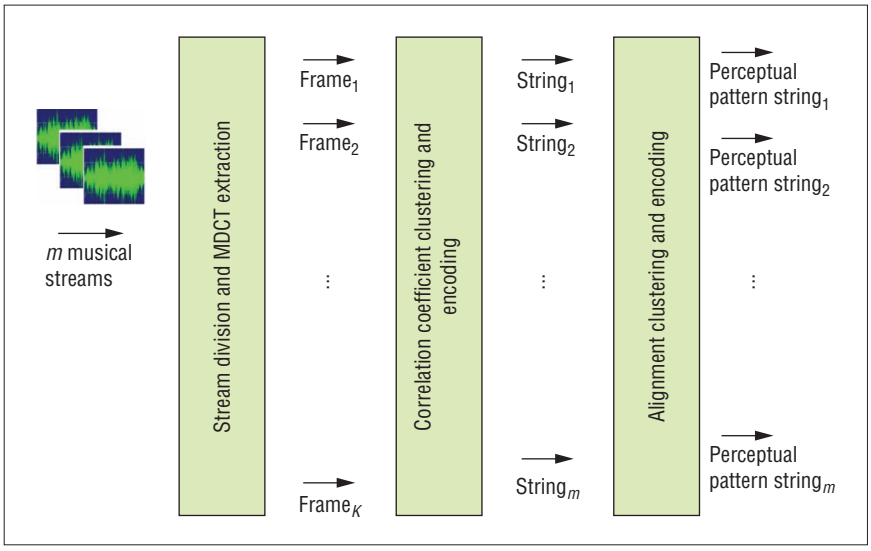

Figure 4. Offline preprocessing. A two-stage clustering approach transforms each music item into a set of consecutive perceptual patterns. MDCT: modified discrete cosine transformation.

Preprocessing Phase

Up to this point, the effectiveness of music recommendation based on musical features has been very limited.[2,3] This is because users' listening interest patterns can be lost during the process of transforming traditional low-level features into semantic features such as tempo, pitch, rhythm, timbre, and so on. To capture the user's listening interests, we propose a two-stage clustering approach that transforms the music into a set of consecutive perceptual patterns (Figure 4). Perceptual patterns are like a music item's genes, representing the music's signature through acoustical and temporal features.

Experimental evaluation results have shown the robustness of perceptual patterns. Now, we show how to derive the perceptual patterns from low-level musical features. Before clustering, we extract low-level features from the music. For music in MP3 format, we extract 38 frames within one second and represent each frame by 576 modified discrete cosine transform (MDCT) coefficients within 26 ms. In our approach, we select 36 important coefficients out of the 576 to reduce the computation cost. The first clustering stage, *frame-based clustering*, groups similar frames into a cluster by calculating the Pearson correlation coefficient. That is, it groups frames with similar spectrums to represent the acoustical features. Basically, frame-based clustering is a hierarchical splitting strategy. Our approach uses two criteria—*proportion* and *density*—as thresholds for the splitting of each leaf node. Proportion represents the total number of frames in a cluster; density represents the number of frames in a cluster's confident radius (The confident radius specifies the qualificatory area around the cluster centroid to verify the frame distribution for density.) Assuming a cluster C_j consists of a set of frames, and c is the centroid of C_j,

$$\text{confident radius} = \bar{d} + \frac{1.95\sigma}{\sqrt{|C_j|}} \quad (1)$$

where

$$\bar{d} = \frac{\sum_{f \in C_j} dist(f,c)}{|C_j|},$$

and

$$\sigma = \sqrt{\frac{1}{|C_j|}\sum_{f \in C_j} dist(f,c)^2}\ .$$

The term $dist(f, c)$ denotes the distance between frame f and centroid c, as follows:

$$dist(f,c) = \frac{\sum_{1 \le i \le |\text{MDCT}|}(mf_f^i - \overline{mf_f})(mf_c^i - \overline{mf_c})}{\sqrt{\sum_{1 \le i \le |\text{MDCT}|}(mf_f^i - \overline{mf_f})^2}\sqrt{\sum_{1 \le i \le |\text{MDCT}|}(mf_c^i - \overline{mf_c})^2}}$$

where |MDCT| = 36, mf_f^i and mf_c^i are the ith MDCT coefficient features of frame f and centroid c; and $\overline{mf_f}$ and $\overline{mf_c}$ are the mean MDCT values of frame f and centroid c. For each leaf node, if the proportion is lower than the presetting threshold, or the density is higher than the presetting threshold, the node isn't split. Finally, the splitting procedure stops when every node has stopped splitting. At the end of frame-based clustering, the music's MDCT values are transformed into a set of symbols. In fact, the acoustical characteristic of music can be represented by the correlation-coefficient-based symbols.

After frame-based clustering, a music stream is represented by a set of sequential symbols. According to the sequential symbols, we enter the second clustering stage, *sequence-based clustering*. This stage considers the music's temporal continuity.

```
Procedure:  Alignment
Input:      Two sequences SE = {se1, se2, ..., sen} and
            SQ = {sq1, sq2, ..., sqm};
Output:     The similarity score scr;
1.   let MX_{SE→SQ}[sr_{n,m}] be a sequence similarity matrix,
     where sr is the similarity score;
/*Steps 2-6: Initialize the values in the matrix*/
2.   initialize v_{0,0} = 0;
3.   for i = 0 to n do
4.       sr_{i+1,0} = sr_{i,0} - 2;
5.   for j = 0 to m do
6.       sr_{0,j+1} = sr_{0,j} - 2;
/*Steps 7-16: Compare two sequences, symbol by symbol*/
7.   for i = 1 to n do
8.       for j = 1 to m do
9.           if se_i = sq_j then
/*Step 9: The ith symbol of SE and the jth symbol of SQ
are the same*/
10.              sr_{i,j} = sr_{i-1,j-1} + 2;
11.          else if se_i ≠ sq_j then sr_{i,j} = sr_{i-1,j-1} - 1;
12.          end if
13.          sr_{i,j} = max{(sr_{i-1,j} - 2), (sr_{i,j-1} - 2), (sr_{i,j})};
/*Step 13: The value of sr_{i,j} is the maximum value of
{(sr_{i-1,j} - 2), (sr_{i,j-1} - 2), (sr_{i,j})}*/
14.      end for
15.  end for
16.  scr = sr_{n,m};
//scr indicates the similarity of two sequences SE
and SQ.//
17.  return scr;
```

Figure 5. Sequence alignment procedure. This algorithm calculates the similarity between two subsequences by comparing the nine symbols of both.

To capture the temporal relationship between two music sequences, we adopt the sequence alignment shown in Figure 5 as the distance function for executing a partition-based K-means algorithm. The method we propose could deploy any of several contemporary clustering methods, such as DBSCAN (density-based spatial clustering of applications with noise), CURE (clustering using representatives), or Birch (balanced iterative reducing and clustering using hierarchies). In this article, we've adopted K-means as our clustering component because it is effective and simple to implement.

In the preprocessing phase, we adopt nine sequential symbols in a symbolic string as a subsequence to perform alignment. In other words, the frame rate is 9. Finally, one second of music can be encoded into four perceptual patterns, and each music string contains a set of perceptual patterns; for example, a 30-second music string consists of 120 perceptual patterns. After sequence-based clustering, each symbolic string for music can be transformed into a set of connective perceptual patterns.

Online Prediction

Online prediction involves inferring the user's musical preference from context information and the perceptual patterns generated in the previous stage. In general, this phase begins when an active user (u_i) visits the system and activates the recommender.

First, we define the involved user item matrix MX as $MX_{U \to I}[v_{n,m}]$, where U is the set of users $\{u_1, u_2, \ldots, u_n\}$, I is the set of items $\{itm_1, itm_2, \ldots, itm_m\}$, and v is the rating value. For example, assume that $MX_{U \to I}[v_{n,m}]$ contains a set of users {Alice, Andre, Ben, Eric, Juice, David} and a set of items $\{itm_1, itm_2, itm_3, itm_4, itm_5\}$, and that Alice rates items $\{itm_1, itm_3, itm_5\}$ at three recommendation transactions. Her rating values are $v_{1,1} = 3$, $v_{1,3} = 3$, and $v_{1,5} = 5$, and the other values of the first tuple are 0. That is, the relevant tuple of the user-item matrix is {3, 0, 3, 0, 5}.

For prediction, our proposed recommender examines the target items related to u_i, one by one.

Step 1. Generating the Refined Submatrix

The goal of this step (lines 1–3 in Figure 6) is to find the users and items most relevant to u_i under similar context conditions. Our recommender then stores the most relevant users and items in a new submatrix. Table 1 lists the different dimensions of context information collected and the related possible values for each dimension. The recommender conducts an iterative procedure, dimension by dimension, to find the most relevant users and items.

The context log contains a set of context transactions representing user votes on an item under a context condition set. We define a log transaction as

{UserName, HT, BT, AT, NV, HY, LT, M, T, S, LN, *Rating*(I)},

where *Rating*(I) indicates the rating value set of $\{itm_1, itm_2, \ldots, itm_m\}$.

Table 2 gives a sample context log containing 13 context transactions. In this case, the first transaction represents that Alice gave itm_1 a rating of 3 under the context conditions

Procedure: Online_Prediction
Input: A set of perceptual-pattern strings M, the user-item matrix $MX_{U \to I}[v_{n,m}]$, the context log, and the target item *targetitm* for active user u_i;
Output: Prediction result true/false;

1. Find the top q similar items to *targetitm* by computing the similarities of context conditions of the context log;
2. Find the top p similar users to u_i by computing the similarities of context conditions of the context log;
3. Combine q similar items I_q and p similar users U_p into a submatrix $MX_{U_p \quad I_q}[v_{p,q}]$;

/*Steps 4-7: Find the positive and negative items rated by the target user.*/

4. **for** $j = 1$ to m **do**
5. **if** $itm_j \notin I_q$ and $v_{u_i,j} \geqq 3$ **then** $POS = POS \cup itm_j$;
6. **if** $itm_j \notin I_q$ and $0 < v_{u_i,j} < 3$ **then** $NEG = NEG \cup itm_j$;
7. **end for**

/* Steps 8-11: Find the positive and negative ratings of the target item.*/

8. **for** $j = 1$ to n **do**
9. **if** $v_{u_j,targetitm} \geqq 3$ and $u_j \notin U_p$ **then** $POS = POS \cup itm_{targetitm}$;
10. **if** $0 < v_{u_j,targetitm} < 3$ and $u_j \notin U_p$ **then** $NEG = NEG \cup itm_{targetitm}$;
11. **end for**

/*Steps 12-13: Find the frequent snippets from the positive item set *POS* and negative item set *NEG*.*/

12. $PF = Gen\_Fre(POS, |POS|)$;
13. $NF = Gen\_Fre(NEG, |NEG|)$;

/*Steps 14-26: Calculate the $INTEREST_{targetitm}$ by performing snippet matching.*/

14. let $targetitm = \{pn_1, pn_2, ..., pn_{|targetitm|}\}$;
15. **for** $w = 1$ to $|targetitm| - winsize$, step $= sp$ **do**

/*Step 15: Slide the window along the target item.*/

16. //*Step* indicates the overlapping pattern cardinality of the sliding windows.//
17. let $ts = \bigcup_{x=w}^{w+winsize-sp} pn_x$;

/*Step 17: Generate the snippet by window size.*/

18. **if** $ts \in (PF \cup NF)$ **then**

/*Step 18: Find the matching snippets in positive and negative snippet sets.*/

19. calculate $TFIDF_{ts}$;
20. let $pcount(ts)$ be the positive count of ts related to PF;
21. let $ncount(ts)$ be the negative count of ts related to NF;
22. $P\_DEGREE_{ts} = pcount(ts)$;

/*Step 22: Count the positive frequency of *ts*.*/

23. $N\_DEGREE_{ts} = ncount(ts)$;

/*Step 23: Count the negative frequency of *ts**/

24. $INTEREST_{targetitm} = INTEREST_{targetitm} + TFIDF_{ts} * (P\_DEGREE_{ts} - N\_DEGREE_{ts})$;

/*Step 24: Accumulate the weighted degree *INTEREST*.*/

25. **end if**
26. **end for**

/*Steps 27-32: Determine *targetitm* as a preferred item by *INTEREST* and *thold*.*/

27. **if** $INTEREST_{targetitm} > thold$ **then**
28. //*Thold* indicates the prediction threshold for the user's interest.//
29. **return** True;
30. **else**
31. **return** False;
32. **end if**

Figure 6. Online prediction algorithm. This phase predicts the user's preference for each target item by discovering the relevant listening interests from the musical content and context information.

Table 1. Context information dimensions and value ranges.

Abbreviation	Context dimension	Possible values
HT	Heartbeat	60–90 beats per minute
BT	Body temperature	35.8–37.5°C
AT	Air temperature	16.0–36.0°C
NV	Noise volume	0–120
HY	Humidity	0–100 percent
LT	Light	Dark, middle dark, normal, middle light, light
M	Motion	Stop, slow, middle, fast
T	Time	Morning, afternoon, evening
S	Season	Spring, summer, fall, winter
LN	Location	Indoor {living room, kitchen, bedroom, bathroom, study}, outdoor

{HT = 79, BT = 35.9, AT = 25.0, NV = 54, HY = 80, LT = dark, M = stop, T = afternoon, S = summer, LN = bedroom}.

Suppose the active user's condition set is {HT = 88, BT = 36.4, AT = 33.0, NV = 50, HY = 20, LT = light, M = slow, T = morning, S = summer, LN = outdoor}. For the heartbeat dimension (HT), the similarity set of {Alice, Andre, Ben, Eric, Juice, David} is {1, 1, 2, 1, 0, 0}, where 0 means irrelevant, and other numerical values mean relevant. Similarly, the similarity set of {itm_1, itm_2, itm_3, itm_4, itm_5} is {1, 1, 1, 2, 0} in terms of HT. Through determining the similarity sets, dimension by dimension, the recommender can induce the most relevant users and items by normalized score count.

Step 2. Generating Positive and Negative Preference Item Sets

From the generated submatrix, the recommender can obtain the positive and negative item sets (lines 4–11 in Figure 6). Here, the recommender considers an item positive if its rating value is 3 to 5. Otherwise, it views the item as negative (disliked). In addition to the submatrix, there is other useful rating information to collect. The first type is user-based rating information, which indicates the items that aren't in the submatrix but have been rated by the active user. The second type is item-based rating information, which indicates the ratings of the target item. Then, we collect the positive and negative item sets from user-based and item-based information, respectively. These positive and negative item sets give the recommender valuable information for preference prediction.

Step 3. Generating Frequent Positive and Negative Snippets

The offline preprocessing phase converts musical streams into perceptual pattern-based strings, and we can view these composed patterns as genes of music. Accordingly, our intention in this step (lines 12–13 in Figure 6; Figure 7) is to discover the

Table 2. Context log example.

Name	HT	BT	AT	NV	HY	LT	M	T	S	LN	itm_1	itm_2	itm_3	itm_4	itm_5
Alice	79	35.9	25.0	54	80	Dark	Stop	Afternoon	Summer	Bedroom	3	0	0	0	0
Alice	89	36.2	31.0	80	72	Light	Stop	Morning	Spring	Kitchen	0	0	3	0	0
Alice	68	36.3	28.0	81	52	Middle	Stop	Night	Summer	Living room	0	0	0	0	5
Andre	72	35.8	31.5	65	30	Middle	Stop	Night	Fall	Study	0	1	0	0	0
Andre	88	36.9	27.9	45	55	Dark	Stop	Morning	Spring	Bathroom	0	0	0	4	0
Andre	85	37.1	31.0	45	25	Light	Middle	Morning	Summer	Outdoor	0	0	0	0	5
Ben	89	36.4	28.9	25	25	Light	Middle	Morning	Spring	Outdoor	0	2	0	0	0
Ben	90	37.3	31	85	15	Light	Middle	Night	Summer	Outdoor	0	0	0	2	0
Eric	88	37.5	28.1	55	32	Middle	Stop	Afternoon	Summer	Study	1	0	0	0	0
Juice	71	35.4	25.6	33	14	Light	Stop	Morning	Summer	Study	0	2	0	0	0
Juice	75	36.8	22.8	72	74	Dark	Stop	Night	Spring	Kitchen	0	0	1	0	0
Juice	75	37	31.0	53	80	Middle light	Stop	Morning	Summer	Living room	0	0	0	0	1
David	75	36.7	25.9	50	80	Dark	Middle	Afternoon	Spring	Outdoor	0	1	0	0	0

patterns that exceed the preset minimum frequency in positive and negative sets.[4] To consider both the consecutive continuity and the duration of patterns, a subsequence called a *sliding window* (defined as *winsize* in Figure 6) slides along a sequence to generate snippets, each consisting of four consecutive patterns. Figure 7 shows the procedure that mines the frequent snippets to represent the potential positive and negative preferences for the active user.

```
Procedure:   Gen_Fre(X, minsupp)
Input:       A set of musical pattern strings X and the
             related threshold minsupp;
Output:      A set of frequent snippets F;
/*Steps 1-11: Compare the snippets of X and all items,
window by window, and count the frequencies of the
snippets.*/
1.    for j = 1 to | X | do
2.        let itm_j = {pn_1, pn_2, ..., pn_|itm_j|};
3.        //pn denotes the perceptual pattern in item itm_j
4.        for w = 1 to |itm_j| - winsize, step = sp do
/*Step 4: Sliding the window along the jth item.*/
5.           let s = pn_x (x = w to w+winsize-sp);
/*Step 5: Generate the snippet by window size.*/
6.           //s denotes the snippet composed of perceptual
             patterns within a sliding window.//
7.           count(s)++;
/*Step 7: Count the frequency of the snippet s.*/
8.           //Count(s) denotes the count of s.//
9.           if s ∉ S then insert s into S;
10.       end for
11.   end for
12.   F = {s | count(s) ≥ minsupp, s ∈ S, |s| = winsize};
/*Step 12: Find the frequent snippets from S.*/
13.   return F;
```

Figure 7. Generating frequent positive and negative snippets. This procedure mines the frequent snippets to represent the potential positive and negative preferences for the active user.

Step 4. Predicting Pattern-Based Preference

An item containing frequent positive snippets could be viewed as a potential recommendation. However, the item might also contain frequent negative snippets. Determining whether the item is a good recommendation for u_i is a challenging task. To deal with this issue (lines 14–32 in Figure 6), the recommender judges the degree of preference for item *targetitm* using the *Interest* measure:

$$Interest_{targetitm} = \sum_{its \subset targetitm \cap PF} TFIDF_{ts} * (P\_degree_{ts} - N\_degree_{ts}) \quad (2)$$

where *ts* is the snippet of *targetitm*, *P_degree* and *N_degree* are the positive and negative frequencies (occurrences) of a snippet (obtained by counting the occurrences of a snippet from the positive and negative items), and *TFIDF* represents the weight of snippet *ts*. Suppose there exists a set of distinct snippets *DS* in *I* and $ts \in DS$. We define the *TFIDF* for *ts* as

$$TFIDF_{ts} = TF_{ts} * IDF_{ts} \quad (3)$$

where

$$TF_{ts} = \frac{\text{no. of occurrences of } ts \text{ in } I}{\sum_{ds \in DS} \text{no. of occurrences of } ds \text{ in } I}$$

and

$$IDF_{ts} = \log\left(\frac{\text{no. of items}}{\text{no. of items containing } ts}\right)$$

Indeed, with regard to pattern discriminability, *TFIDF* is helpful for identifying the snippet. That is, if $TFIDF_{ts}$ is high, *ts* is a good description of the content of *targetitm*. With regard to the *term frequency–inverse document frequency* (tf-idf) weighting scheme, we can view *Interest* as a weighted degree derived from the accumulated difference between positive and negative frequencies. After calculating the Interest values for the target items unrated by the active user, the recommender can rank the target items by the related Interest values. In this way, our proposed recommender can effectively make mobile music recommendations even in cases of diminished information, as when facing a new item or user.

Experimental Evaluations

Now that we have described how uMender works, we turn to the experimental results of our performance evaluation of the system.

Experimental Data

To evaluate uMender's performance, we conducted experiments using semireal data. The experimental data consisted of two parts: music-rating data and context data. For the music-rating data, we gathered real rating logs from Amazon.com. Because very little context data is publicly available, we constructed a simulator to build the synthetic context data.

The music-rating data contains six musical genres: classic rock, classical, jazz, Latin, opera vocal, and rock.

Table 3. Evaluating uMender: Experimental settings.

Parameter	Default value, synthetic context data	Default value, real context data
Sliding window size, *winsize*	1	1
No. of patterns	100	100
No. of items, $\|I\|$	130	130
No. of users, $\|U\|$	251	36
Window step, *sp*	1	1
No. of context condition paths	144	9
Prediction threshold for user's interest, *thold*	0	0
Testing rate (%)	30	30

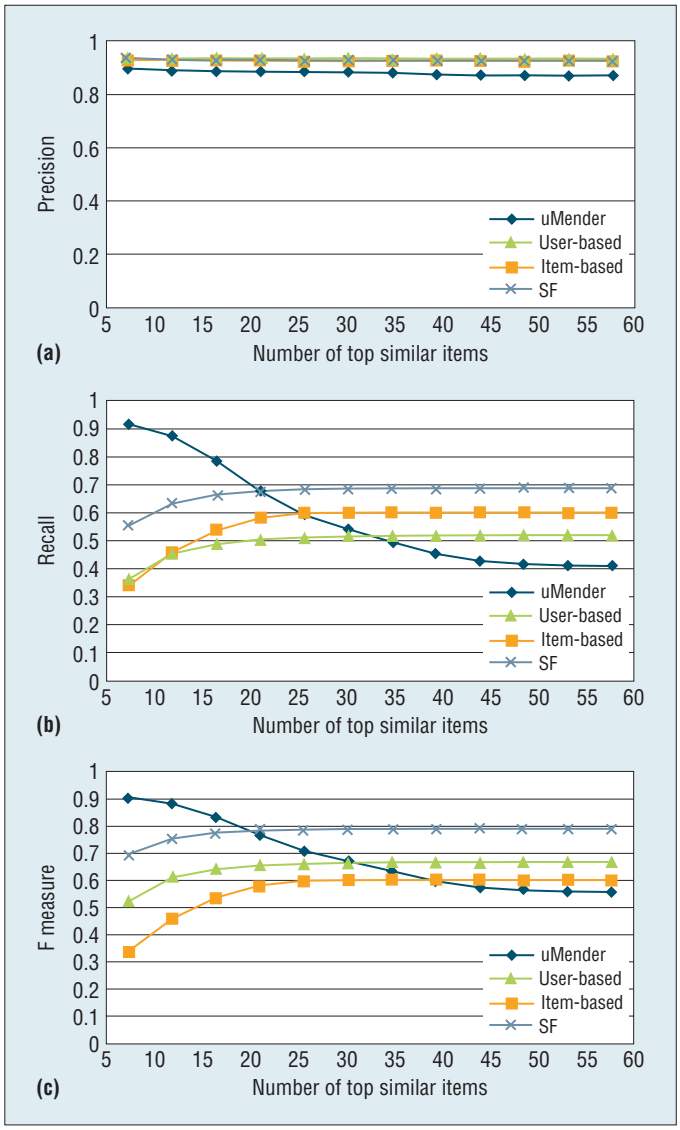

Figure 8. Comparison of user-based, item-based, similarity fusion (SF), and uMender recommendation methods for the top *q* similar items: (a) precision, (b) recall, and (c) *F* measure.

The collection consists of 130 albums (each is an item) and 251 users. The data's total duration is about 25 hours, and its data size is about 1.5 Gbytes.

For the context data, we adopted two kinds of data sets. One is synthetic data, and the other is real. To make the evaluation complete, we constructed a simulator to build the synthetic data using our context dimension definitions. Our simulator views context data within a tree structure, with each path in the tree representing a composition of conditions. That is, a context transaction within the context dimensions can be classified in a condition path by traversing the context condition tree. We associated the music-rating data with this context data to form a transaction set. The main consideration behind the simulation is to connect a condition path and a submatrix as a semireal context transaction.

In addition to synthetic data, we implemented a system prototype and invited 36 users to test the system. To gather real rating transactions, we asked the testers to rate music under different context conditions (location, time, air temperature, noise, light, humidity, motion, and so on). Overall, we collected 2,785 real rating transactions.

We adopted a 10-fold cross-validation approach to carry out the evaluations. Table 3 lists the parameter settings.

In our experiments, we adopted four basic rates to evaluate the recommendation techniques: true positive (*TP*), false positive (*FP*), true negative (*TN*), and false negative (*FN*). Using these measures, we constructed three criteria for evaluating the recommender:

- $Precision = TP/(TP + FP)$;
- $Recall = TP/(TP + FN)$; and
- $F = (2 \times precision \times recall)/(precision + recall)$.

Precision reveals the fidelity of the prediction, whereas recall reveals the completeness of the prediction. To consider precision and recall simultaneously, the F measure reveals the harmonic mean of both.

Experimental Results

In the experiments, we compared four recommendation methods—user-based,[5] item-based,[6] similarity fusion (SF),[7] and uMender—under different settings of parameter q for choosing the top q similar items and users. The item-based technique induces music items of interest by finding the most-relevant items; the user-based technique bases its recommendations on the most-relevant users; and SF and uMender base their recommendations on the most relevant items and users.

Our first evaluation using the synthetic data was to compare the precision of the four recommenders. Figure 8a shows that by this measure the four methods are quite close. Because uMender's *FP* is slightly higher than that of the others, its precision is slightly lower. However, Figure 8b reveals that uMender brings out a far higher recall than other recommenders. For recall, a larger q leads to more noise items and users for preference prediction. As a result, *TP* decreases as q increases.

Considering both precision and recall, Figure 8c reveals some important phenomena for the *F* measure. First, for uMender, the best setting of q is 5; for the other recommenders, the best setting is 20. Second, on the basis of the top five items or users (that is, $q = 5$), Figure 8c shows that in terms of the *F* measure, uMender outperforms the item-based method by about 168 percent, the user-based method by about 74 percent, and the SF method by about 30 percent. Third, uMender's *F* measure decreases as q increases.

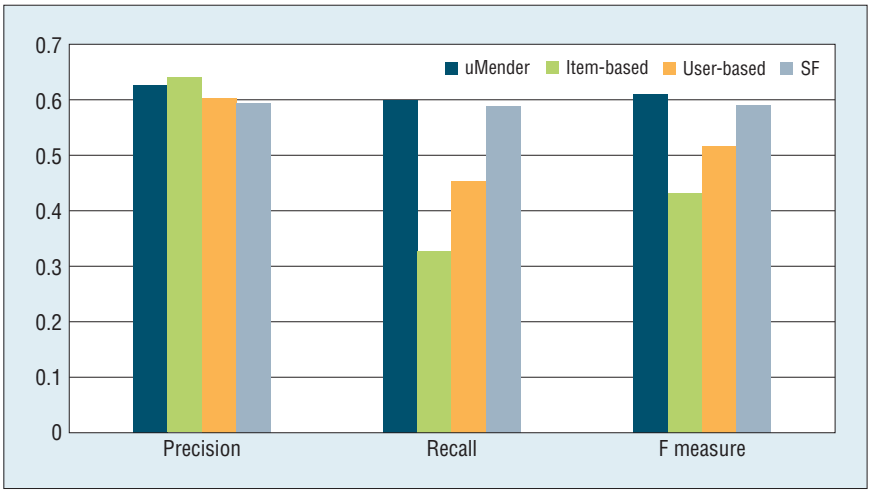

Figure 9. Comparison of user-based, item-based, similarity fusion (SF), and uMender recommendation methods using real data.

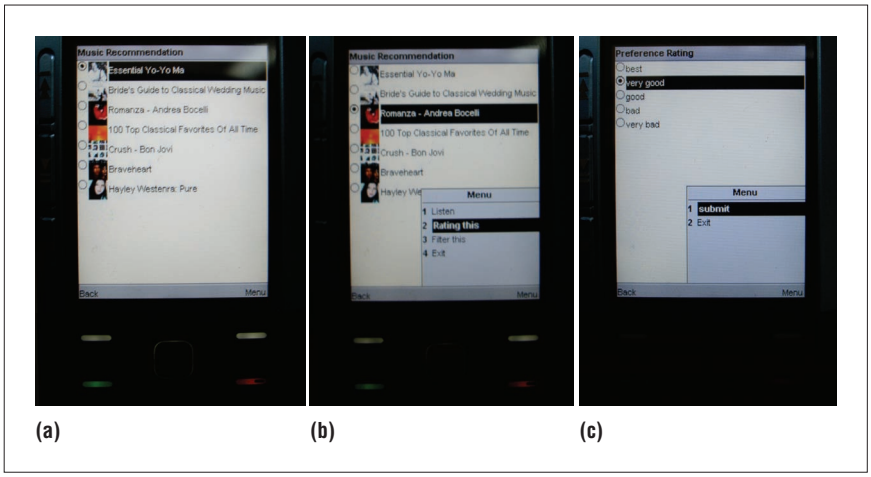

Figure 10. Screen snapshots of the uMender system prototype: (a) recommendation list; (b) music-listening function; (c) preference rating.

This is mainly because the more items there are, the more noise there is and the more diverse the music content is, so uMender's *F* measure is lower. However, in terms of computation cost, uMender offers a significant improvement over the other recommenders with q set to 5. This indicates that context information is very helpful in finding the most-relevant items and users to facilitate music recommendation.

From another perspective, these results indicate that uMender can deliver high quality at minimum cost. That is, uMender performs better than other recommenders, even when little information about items and users is available.

In addition to uMender with synthetic context data, we also evaluated our recommender using a system prototype. Figure 9 shows that uMender also outperforms the other recommendation methods in this testing under real conditions. Overall, our experimental results on both real and synthetic data sets reveal that the power to predict a user's preference is strengthened by integrated mining of music content and context information.

Figure 10 shows screen snapshots of the system prototype. Figure 10a

THE AUTHORS

Ja-Hwung Su is pursuing his PhD in the Department of Computer Science and Information Engineering at National Cheng Kung University in Taiwan. His research interests include data mining and data warehousing. Su has an MS in information management from I-Shou University. Contact him at bb0820@ms22.hinet.net.

Hsin-Ho Yeh is a research assistant in the Institute of Information Science, Academia Sinica, Taipei. His research interests include data mining, multimedia mining, and image and video content analysis. Yeh has an MS in computer science and information engineering from National Cheng Kung University in Taiwan. Contact him at hhyeh@iis.sinica.edu.tw.

Philip S. Yu is a professor in the Department of Computer Science of the University of Illinois at Chicago, where he holds the Wexler Chair in Information Technology. His research focuses on data mining. Yu has a PhD in electrical engineering from Stanford University. He is a Fellow of the ACM and the IEEE. Contact him at psyu@cs.uic.edu.

Vincent S. Tseng is a professor in the Department of Computer Science and Information Engineering and the director of the Institute of Medical Informatics at National Cheng Kung University in Taiwan. His research focuses on data mining. He has a PhD in computer and information science from National Chiao Tung University in Taiwan. He is a member of the IEEE and the ACM. Contact him at tsengsm@mail.ncku.edu.tw.

shows the recommendation list, Figure 10b shows the music-listening function, and Figure 10c shows the preference rating. Through this system prototype, the testing users can rate the music items according to their preferences.

Our experimental results show that uMender outperforms existing recommenders in terms of recommendation quality. For future work, there remain some issues to address. First, we will investigate more dimensions of contextual information to improve the recommender's quality. Second, we hope to adopt perceptual patterns to enhance content-based retrieval. Third, we intend to apply the integrated mining of context information and multimedia content to other multimedia applications. ■

Acknowledgments

This research was supported by Taiwan's National Science Council under grant no. NSC 98-2631-H-006-001, and by Taiwan's Ministry of Economic Affairs under grant no. 97-EC-17-A-02-s1-024.

References

1. G. Reynolds et al., "Interacting with Large Music Collections: Towards the Use of Environmental Metadata," *Proc. IEEE Int'l Conf. Multimedia and Expo*, IEEE Press, 2008, pp. 989–992.
2. J.J. Aucouturier, F. Pachet, and M. Sandler, "'The Way It Sounds': Timbre Models for Analysis and Retrieval of Music Signals," *IEEE Trans. Multimedia*, vol. 7, no. 6, 2005, pp. 1028–1035.
3. Y. Jiao et al., "MDCT-Based Perceptual Hashing for Compressed Audio Content Identification," *Proc. IEEE 9th Workshop Multimedia Signal Processing* (MMSP 07), IEEE Press, 2007, pp. 381–384.
4. R. Agrawal, T. Imielinski, and A.N. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1993, pp. 207–216.
5. R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, 2002, pp. 331–370.
6. B.M. Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l Conf. World Wide Web* (WWW 01), ACM Press, 2001, pp. 285–295.
7. J. Wang, A.P. de Vries, and M.J.T. Reinders, "Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion," *Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval* (SIGIR 06), ACM Press, 2006, pp. 501–508.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*

Engineering and Applying the Internet

IEEE Internet Computing

IEEE Internet Computing reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment.

For submission information and author guidelines, please visit www.computer.org/internet/author.htm