Vietnam Journal of Computer Science (2018) 5:251–261
https://doi.org/10.1007/s40595-018-0120-1

REGULAR PAPER

Mining and applications of repeating patterns

Ja-Hwung Su[1,2] · Tzung-Pei Hong[3,4] · Chu-Yu Chin[5,6] · Zhi-Feng Liao[4] · Shyr-Yuan Cheng[6]

Received: 8 November 2017 / Accepted: 31 May 2018 / Published online: 14 June 2018
© The Author(s) 2018

Abstract
Mining the valuable knowledge from real data has been a hot topic for a long time. Repeating pattern is one of the important knowledge, occurring in many real applications such as musical data and medical data. In this paper, our purposes are to contribute an efficient mining algorithm for repeating patterns and to conduct a real application using the repeating patterns mined. In terms of mining the repeating patterns, although a number of past studies were made on this issue, the performance cannot still earn the users' satisfactions especially for large data sets. For this issue, in this paper, we propose an efficient algorithm named Fast Mining of Repeating Patterns, which achieves high performance of discovering the repeating patterns by a novel index called Quick-Pattern Index. In terms of applications, a music recommender system named repeating-pattern-based music recommender system is proposed to deal with problems in music recommendation. Even facing a very sparse rating matrix, the recommendation can still be completed. The experimental results show that our proposed mining algorithm and recommender system outperform the previous works in terms of efficiency and effectiveness, respectively.

Keywords Repeating pattern · Quick-Pattern Index · Data mining · Knowledge discovery · Music recommendation

1 Introduction

The great progress of information technology makes the real data grow rapidly. Actually, there is a large amount of knowledge in these data such as graph data, sequence data, transaction data, and so on. Therefore, data mining on pattern discovery has been studied for many decades. To discover the valuable patterns hidden in the real data, a number of mining algorithms are proposed nowadays. In recent studies, the patterns are categorized into several categories, including association patterns, sequential patterns, cyclic patterns, repeating patterns, and so on. Different patterns are useful to different fields of data engineering. For association patterns, they are motivated by applications of market basket analysis to discover relations between products purchased. For sequential patterns, they are somewhat different from the association patterns, because the sequential patterns are with temporal continuities. For cyclic patterns, the starting pattern is also the ending pattern in a sequence, which can be viewed as the extension of sequential patterns. For repeating patterns, a repeating pattern contains a set of sequential elements, which can also be viewed as a sequence repeats in a regular form. For example, a string {1, 2, 3, 1, 2, 4, 1, 2} contains a repeating substring {1, 2}, which is identified as a repeating pattern. The major difference between the cyclic patterns and repeating patterns is that a repeating pattern does not need to consider the start and end of a sequence must be the same. In fact, repeating patterns are popular, because they can be regarded as a set of representative patterns facilitating

It is an extended version of the paper "A high-performance algorithm for mining repeating patterns" presented in The Ninth Asian Conference on Intelligent Information and Database Systems.

✉ Tzung-Pei Hong
tphong@nuk.edu.tw

1 Department of Information Management, Cheng Shiu University, Kaohsiung, Taiwan

2 Department of Information Management, Kainan University, Taoyüan, Taiwan

3 Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan

4 Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

5 Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan

6 Telecommunication Laboratories, Chunghwa Telecom Co., Ltd., Taoyüan, Taiwan

Table 1 Example of a user-item rating matrix

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	2	0	0	3	0	0
User 2	2	0	0	0	0	4
User 3	0	3	0	0	2	0
User 4	0	2	0	2	0	3
User 5	3	2	0	2	0	0
User 6	1	0	0	0	3	0

object recognition. Lots of repeating patterns appear in our life, such as musical data and medical data.

For musical data, a music piece can typically be transformed into a pattern string. Inspired by this idea, some past studies [2,6,7,9,21] attempted to extract the harmonic feature patterns from the audio object to achieve the music search/retrieval. Although the repeating patterns were successfully applied to music retrieval, no research is proposed for the music recommendation. Basically, the goal of music recommendation is to detect the user listening interest and then provide the potential music preferred. Table 1 shows an example of a rating matrix. In this example, there are six users and six music pieces (called items in this paper). It shows the user ratings for the items, which are ranged from 1 to 5. Note that, zero indicates that the user never rates the item. The major intent of a typical recommender system is to predict the unrated item ratings (also called "unknown ratings") and return the top k unrated items to the user. From this table, traditional recommender systems always suffer from two problems, namely, data sparsity and new item. The data sparsity problem indicates a rating matrix contains, so few ratings that the prediction is not easy to complete, while the new item problem indicates the item never rated by any user is also not easy to predict.

To aim at the issues above, the contributions can be summarized as follows.

I. To accelerate the retrieval of repeating patterns, in this paper, we propose an algorithm named Fast Mining of Repeating Patterns (FMRP) to achieve high performance of mining repeating patterns by the proposed Quick-Pattern Index (QPI). With this index, the occurrences and positions of the patterns are kept to reduce the cost of searching the repeating patterns. Hence, without scanning the sequence iteratively, the repeating patterns can be discovered by only one scan of the input sequence. The experimental results reveal our proposed algorithm performs better than the compared methods in terms of execution time.
II. In addition to provide an efficient mining algorithm, another aim in this paper is to apply the repeating patterns to the real application—music recommendation. The main difference between this paper and previous recommender systems is that the previous recommender systems never consider the sequentially repeating patterns which represent the user listening senses. From the experimental results, we can know that the problems in traditional recommender systems can be alleviated clearly through the repeating pattern-based recommendation.

The rest of this paper is structured as follows. In Sect. 2, a brief review of related work is shown. In Sect. 3, we present our proposed method for mining of repeating patterns and predicting the user ratings on music. Examples are shown in Sect. 4. The evaluations of the proposed methods are depicted in Sect. 5. Finally, conclusions and future work are shown in Sect. 6.

2 Related work

It is because the aims of this paper are the mining algorithm and the related applications that the review of the predecessors includes three parts, namely, algorithms of mining of repeating patterns, applications of repeating patterns, and music recommendation.

2.1 Algorithms of mining of repeating patterns

Hsu et al. [9] proposed a method to generate the repeating patterns. In this method, a string-join operation and a data structure called RP tree were proposed to achieve high performance of mining of repeating patterns. The basic idea of this approach is to iteratively join two short repeating patterns into a long one. To speed up the join procedure, a tree structure named RP tree was proposed. Although the tree structure can reduce the time complexity of join operations, the checking cost is so high that the generation of repeating patterns is inefficient. In addition to RP tree, another tree structure for generating the repeating patterns is suffix tree. Basically, the suffix tree is the compressed tree for the nonempty suffixes of a string. Since a suffix tree is a compressed tree, it consists of an important idea that the procedure of mining of repeating patterns highly refers to its sub-trees. Once constructed, several operations can be performed quickly, for instance, locating a substring, locating matches for a regular expression pattern, etc. Suffix trees also provide the linear-time solution for the longest common substring problem [16,20]. Unfortunately, the construction of such a tree for a string takes much time and space.

2.2 Applications of repeating patterns

In the field of medical biology, repeating subsequences are a kind of repeating patterns. Actually, the repeating patterns

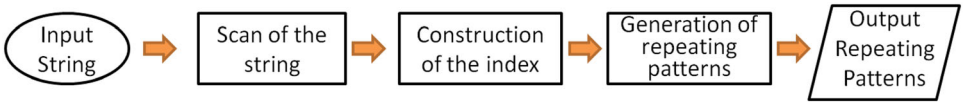

Fig. 1 Workflow of the proposed method

in biological cell DNA occur in multiple copies throughout a genome. The functions and descriptions of these subsequences are currently being characterized by scientists. Tandem repeat is a kind of repeating patterns occurring in DNA when a pattern of nucleotides repeats and the repetitions are directly adjacent to each other. Several protein domains also form tandem repeats within their amino acid primary structures. In practical, tandem repeats are very helpful in the field of bioinformatics. In addition to bioinformatics, repeating patterns are usually supported for the matching of local image features. They can be modeled as a set of sparse repeated features in which the crystallographic group theory. Muller et al. [5] proposed an approach to detect symmetric structures and to reconstruct a 3D geometric model. Liu et al. [10,24] proposed a new method for detection of repeated patterns following a *Kronecker Product formulation.* They handled problems of pose variation and varying brightness by employing the low-rank part of the rearranged input facade image. *Automatic video summarization* [11,23] was proposed as an effective way to accelerate the video browsing and retrieval. The video structure is first analyzed by spatial–temporal analysis. Then, the video non-trivial repeating patterns are extracted to remove the visual-content redundancy among videos.

2.3 Music recommendation

In general, music recommendation can be categorized into two main classifications, namely, user-based and item-based recommender systems. For user-based music recommender systems [8,14], the basic idea is to predict the unknown ratings by the known ratings of relevant users. Bobadilla et al. [1] defined the significances of the users and items to predict the users' ratings. In contrast to user-based recommender systems, the item-based recommender systems [4,19] adopted known ratings of relevant items to predict the unknown ratings. In addition to the above recommender systems, Qi et al. [13] computed the users' similarities based on the inferred tag ratings to conduct the recommendation. In [12], the user profiles and tag information are fused to generate a framework of joint item-tag recommendation. Su et al. [17,18] integrated information of tags, play counts, and artists to improve the recommendation quality. Cheng et al. [3] integrated acoustic features and user personalities to conduct a personalized recommendation service. Rahman et al. [15] proposed a personalized recommender system by fuzzy influences. Xue et al. [22] attempted to discover the preference factors by matrix factorization models. Although the above recommender systems perform well, they still encounter problems of *new item* and *data sparsity*.

3 Proposed methods

3.1 Mining of repeating patterns

3.1.1 Overview of the proposed mining algorithm

In recent years, several schemes have been proposed to discover repeating patterns. However, the execution performance is limited in the search strategy, which is not satisfactory. To improve the execution performance, in this paper, we propose an efficient algorithm that utilizes a novel data structure to facilitate the discovery of repeating patterns. In other words, without scanning a given sequence iteratively, the repeating patterns can be discovered by only one scan of the sequence. As shown in Fig. 1, the mining process can be divided into two primary phases, namely, construction of the index and generation of repeating patterns.

I. Construction of the index

This is the basis of the proposed method, and the major goal of this phase is to construct an informative index to reduce the cost of mining the repeating patterns. Basically, this index called Quick-Pattern Index (QPI) contains two of pattern information, including positions and occurrences.

II. Generation of repeating patterns

In this phase, the main goal is to search the repeating patterns using the constructed index. To this end, a novel mining method called Fast Mining of Repeating Patterns (FMRP) is proposed in this work. Through the prefix-search strategy, the repeating subsequences (also called repeating patterns) are mined successfully.

Before describing our proposed method, the repeating pattern is defined as Definition 1.

Definition 1 For a string B, if a substring A appears more than once, its length is larger than one and no any other substring contains A, we call A is a repeating pattern of B. □

For example, the repeating pattern of a string {1, 2, 3, 1, 2, 3} is {1, 2, 3}.

Fig. 2 Process of construction of the index

Input: A string $ST=\{s_1, s_2, ..., s_k\}$ containing g unique symbols (also called pattern);
Output: A set QPI;
Process: Construction of the index
1. initialize the string array $S[i]$ as a null array;
2. i=0;
3. **for** each pattern s_i in ST **do**
4. i++;
5. insert s_i into $S[i]$;
6. $p_i \rightarrow count$++; // $p_i \rightarrow count$ denotes the occurrence of pattern s_i, where $s_i = p_i$;
7. $p_i \rightarrow pos[p_i \rightarrow count]=i$;// $p_i \rightarrow pos[p_i \rightarrow count]$ denotes the position of pattern s_i;
8. $P= P \cup p_i$; //P is the set containing patterns, occurrences and positions;
9. **end for**
10. **return** $S[i]$ and P as QPI;

3.1.2 Construction of the index

Figure 2 shows the process of constructing the index. In this process, two indexes are constructed, including an array storing all patterns and an array storing the pattern positions in the input string. Through these two indexes, the next process called generation of repeating patterns can perform efficiently.

3.1.3 Generation of repeating patterns

Figure 3 shows the algorithm of generating the repeating patterns. Basically, it can be regarded as a prefix-search paradigm that mines the repeating patterns prefixed by a pattern. Hence, as shown in Line 1, the whole procedure starts with one of the distinct patterns, and then mines the prefixed patterns. Under the prefixed pattern p_x, Line 2 shows that the patterns in the next positions of p_x are grouped into the set $p_x.NextSteps$ and they are linked into the paths, where the root is p_x. Next, Lines 3–10 show that, for each pattern p_i in $p_x.NextSteps$, all patterns in the next positions of p_i are further grouped into the set $p_x.NextTwoSteps$. In this loop, if the number of patterns in the next positions of p_i is larger than 2, the algorithm keeps going to Line 1. That is, this is an iterative process which does not stop until all prefixed paths are generated. In detail, the whole search is divided into a number of search sub-paths prefixed by p_i, where the root is p_x. In the next loop of the algorithm, Lines 12–19 show that the prefixed traversals are performed to mine all repeating patterns.

3.2 Music recommendation by the repeating patterns

3.2.1 Overview of the proposed music recommender system

Previously, most data mining works only focused on how to efficiently generate the patterns. Though the mining algorithms are shown to be very efficient, the practical applications are always lacked. This problem makes the readers hard to know the real effectiveness of the generated patterns. Consequently, in this paper, we propose a recommender system to clearly reveal the values of the repeating patterns. In addition, the proposed music recommender system based on the repeating patterns can deal with traditional recommendation problems of *data sparsity* and *new item*. For these purposes, as shown in Fig. 4, the proposed recommender system can be decomposed into two stages, namely, offline preprocessing and online recommendation.

I. Offline preprocessing

In this stage, the major operations include low-level feature extraction, music symbolization, repeating-pattern generation, and rating classification modeling. In detail, the major intents for our proposed recommender system are: (1) to describe the music by a set of sequential acoustic patterns and (2) to bridge the listening senses and the user preferences by the repeating-pattern-based rating classification model.

II. Online recommendation

For an active user, in this stage, the unknown item ratings are predicted by the repeating-pattern-based rating classification model. Finally, the items are sorted by the predicted ratings, and top k items are returned to the active user. Note that, because the sorting operation is straightforward, the main attention of the proposed method is paid on the rating prediction.

3.2.2 Offline preprocessing stage

A. Music low-level feature extraction

In this paper, because mel-scale frequency cepstral coefficients (MFCCs) are nearer the listening senses, they are used as the low-level features of music. Figure 5 shows the structure of music features, which can be divided into three levels.

Fig. 3 Algorithm of generating of repeating patterns

Input: A set *QPI*;
Output: A set of repeating patterns *RPS*;
Process: Generation of the repeating patterns;
1. **for** each unique p_x **do**
2. find all patterns in the next positions of p_x, which are called set of $p_x.NextSteps$;
3. **for** each $p_i \in p_x.NextSteps$ **do**
4. find all patterns in the next positions of p_i, which are called set of $p_x.NextTwoSteps$;
5. link p_i and p_x as a path ph_x;
6. **if** $|p_x.NextTwoSteps|>=2$ **then**
7. let $p_x = p_i$;
8. **goto** Line 1;
9. **end if**
10. **end for**
11. **end for**
12. **for** each path ph_x **do**
13. **for** each sub-path *sp* of ph_x **do**
14. **if** $sp \notin RPS$ **then**
15. $RPS = RPS \cup sp$;
16. let p_i be the leave of *sp*;
17. keep $|p_i.NextSteps|$ as the count of *sp*;
18. **end if**
19. **end for**
20. **return** *RPS*;

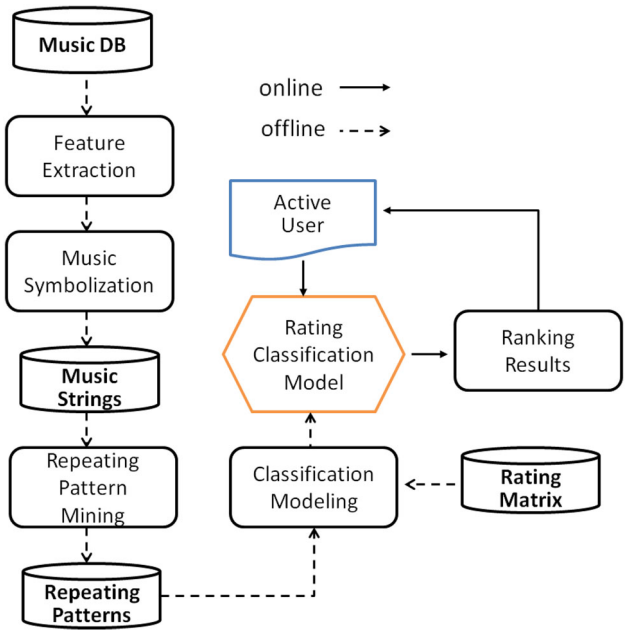

Fig. 4 Framework of the proposed music recommender system

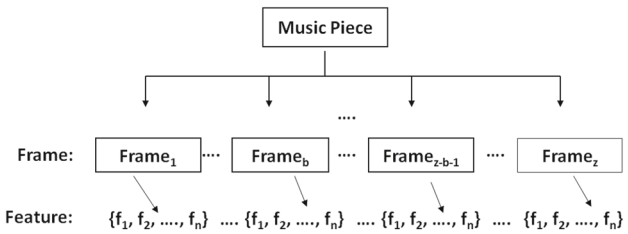

Fig. 5 Structure of MFCCs

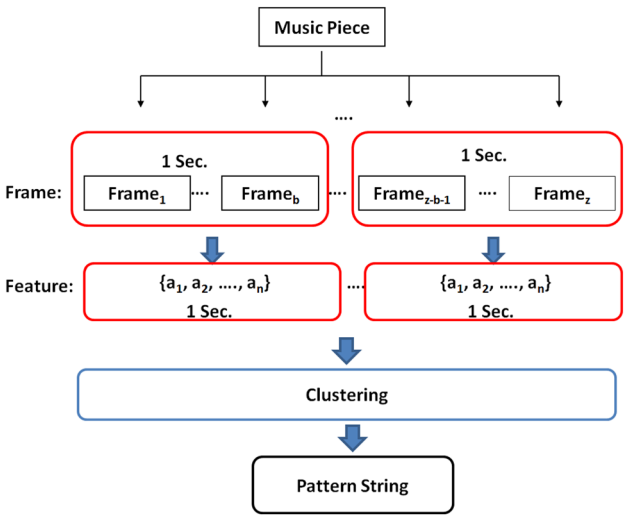

Fig. 6 Process of merging frames into a second-based pattern string

An analog music piece is composed of a number of frames, and a frame consists of a set of feature dimensions.

B. Music symbolization

After extracting the low-level features, the music pieces are represented by different numbers of frames. In this operation, as shown in Fig. 6, the frames in 1 s are first merged. That is, a music piece is further represented by several feature vectors, where a feature vector represents 1-s MFCCs. Next, the second-based feature vectors are grouped into a set of clusters. Then, each second-based feature vector is encoded by the cluster number, and finally, a music piece is transformed into a pattern string.

C. Mining of repeating patterns

This operation is to generate the repeating patterns of each pattern string by our proposed mining algorithm. Here, the mining procedure is not described again. Finally, the Term Frequencies (called TF in this paper) of the repeating patterns are calculated, which can be defined as Definition 2.

Definition 2 Assume that a music database contains a set of music strings DB $= \{I_1, I_2, \ldots, I_j, \ldots, I_{|\mathrm{DB}|}\}$ and a set of unique repeating patterns $P = \{p_1, p_2, \ldots, p_i, \ldots, p_{|P|}\}$. Then, the jth music item can be represented as

$$I_j = \{p_1.\mathrm{TF}, p_2.\mathrm{TF}, \ldots, p_i.\mathrm{TF}, \ldots, p_{|P|}.\mathrm{TF}\},$$

where

$$p_i.\mathrm{TF} = p_i \,.\, F / \sum_{0<y\leq|p|} p_y \,.\, F,$$

p_i.TF denotes the term frequency of the ith pattern p_i and $p_i.F$ denotes the frequency of the ith pattern p_i in the jth music pattern string I_j. □

Note that, for music recommendation, the repeating pattern is somewhat different from that in Definition 1. In formal, assume that there are two repeating substrings A and C in a string B, where A contains C. Here, even if A contains C, A and C are both kept as our repeating patterns for music recommendation. For example, the repeating patterns of a string {1, 2, 3, 1, 2, 3} include {1, 2, 3}, {1, 2} and {2, 3}.

D. Construction of rating classification model

After three above operations, a music piece is represented by a number of repeating patterns with the related term frequencies TFs. In this operation, for each user, a known rating of the jth item is regarded as a training vector, which is defined as

$$TI_j = \{I_j, v_j\},$$

where I_j denotes a set of pattern TFs referred to Definition 2, and v_j denotes the known rating of the jth item. Then, these training vectors for each user are used to construct the personal rating-based classification models. In this paper, LIBLINEAR [25] is adopted as the classifier.

3.2.3 Online recommendation stage

This stage is triggered with a visit of an active user. Basically, the main idea behind this stage is to regard the rating prediction as a rating classification. Because each user's model has been generated in the offline preprocessing stage, all unknown ratings of the active user can be predicted quickly.

Table 2 Example of the resulting set P

Pattern	Occurrence	Position
p_1	4	1, 5, 9, 13
p_2	1	4
p_3	4	2, 6, 10, 14
p_4	1	8
p_5	2	3, 11
p_6	1	12
p_7	2	7, 15
p_8	1	16

4 Illustrative examples

To make the proposed methods easy to understand, two examples are shown in this section.

4.1 Example for mining of repeating patterns

4.1.1 Construction of the index

In this example, we give a string $ST = \{1, 3, 5, 2, 1, 3, 7, 4, 1, 3, 5, 6, 1, 3, 7, 8\}$ which contains 16 symbols (called patterns in the example) and the kind of unique patterns is 8. In this process, first, the string array $S[\,]$ is initialized as a null array. Second, ST is loaded into the array $S[\,]$. Finally, a table containing occurrences and positions of patterns is constructed, as shown in Table 2.

4.1.2 Generation of repeating patterns

By referring to Fig. 3, the exampling process is shown as follows.

Step 1: Considering Table 2, a pattern is selected from Table 2, referring to Line 1 of Fig. 3.
Step 1.1: The selected pattern p_x is conducted as the root and the next positions of p_x are grouped into the set $p_x.NextSteps$. Here, the example pattern p_x is p_1 and its $NextSteps$ is P_3.
Step 1.2: For each p_x, find each pattern p_i in $p_x.NextSteps$. If the occurrence of p_i is more than 1, link p_i and p_x as the path where the root is p_x. Then, all patterns in the next positions of p_i are further grouped into the set $p_x.NextTwoSteps$. If the occurrences of p_i are more than 1, link p_i and all patterns in the next positions of p_i. In this example, as shown in Fig. 7, the green grid is defined as {pattern#, (next positions), (parents)}. In Fig. 7, $\{p_3, (3, 7, 11, 15), (p_1)\}$ denotes that, the p_i is p_3, where the next position set is {3, 7, 11, 15} and the parent is p_1.

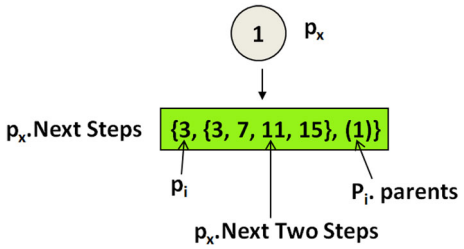

Fig. 7 Example of {pattern#, (next positions), (parents)}

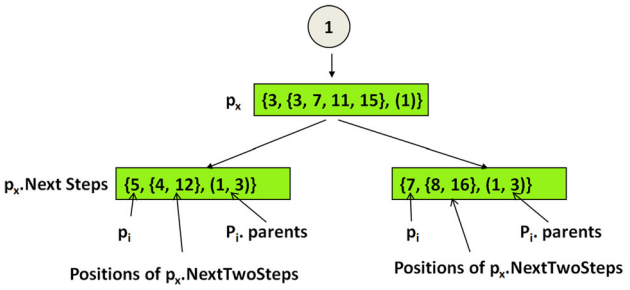

Fig. 8 Example for generating new *NextSteps*, containing p_5 and p_7, where $p_x = p_3$

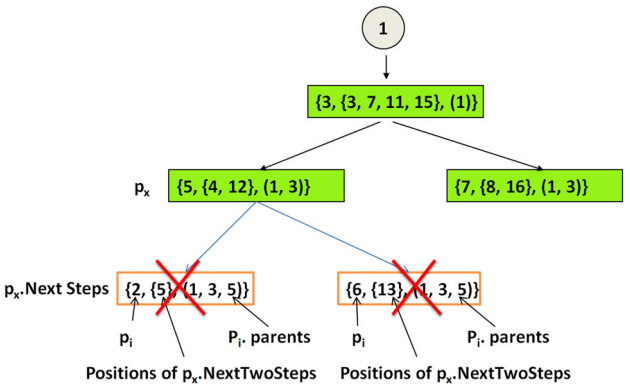

Fig. 9 Example of generating new *NextSteps*, containing p_2 and p_6, where $p_x = p_5$

Next, because the occurrences of all patterns $\{p_5, p_7\}$ in the p_3.*NextSteps* are more than 1, $\{p_3\}$ and $\{p_5, p_7\}$ are linked. After generating p_x.*NextTwoSteps*, for each pattern p_i in p_x.*NextSteps*, let $p_x = p_i$, as shown in Figs. 8, 9 and 10. If the occurrences of all patterns $\{p_2, p_6, p_4, p_8\}$ in the p_3.*NextTwoSteps* are less than 2, check whether the paths whose root generated in Step 1 have existed in the result set *RPS* or not. If the paths have not existed in the result set *RPS*, insert them into *RPS*. In this example, this step does not stop until p_x.*NextTwoSteps* contains only one pattern. Because the occurrences of all patterns $\{p_2, p_6, p_4, p_8\}$ in the p_3.*NextTwoSteps* are less than 2, the loop stops. Finally, $\{p_1, p_3, p_5\}$ and $\{p_1, p_3, p_7\}$ are inserted into *RPS*, as shown in Fig. 9.

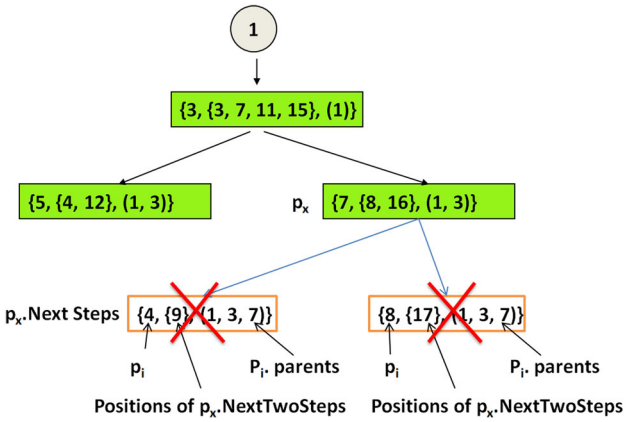

Fig. 10 Example of generating new *NextSteps*, containing p_4 and p_8, where $p_x = p_7$

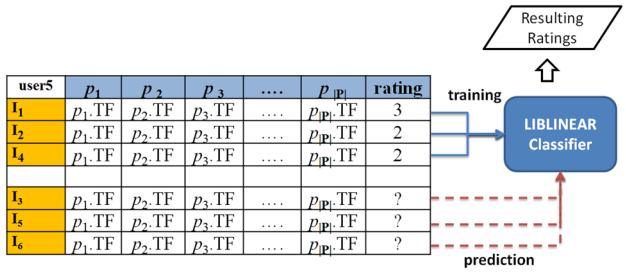

Fig. 11 Example of rating predictions

Step 2: Repeat Step 1 until all repeating patterns are generated.

4.2 Example for repeating-pattern-based music recommendation

Based on Table 1, Fig. 11 shows an example of how to predict the unknown ratings. In this example, user 5 is set as the active user. Assume that, the ratings of items 3, 5, and 6 are unknown and the ones of items 1, 2, and 4 are known. That is, items 1, 2, and 4 are used to train the classifier, and items 3, 5, and 6 are the target items to predict. It is because the classification model has been conducted already in the preprocessing stage that the rating prediction is very fast.

5 Empirical study

Recalling the goals of this paper, the experiments contain two main parts, namely, evaluations for mining of repeating patterns and evaluations for repeating-pattern-based music recommendation. In the following subsections, the individual experimental results are shown separately in great detail.

Table 3 Parameter settings for mining of repeating patterns

	Minimum	Maximum	Interval
#Transactions	500	1000	100
#Transaction length	500	1000	100
#Pattern categories	10	200	50

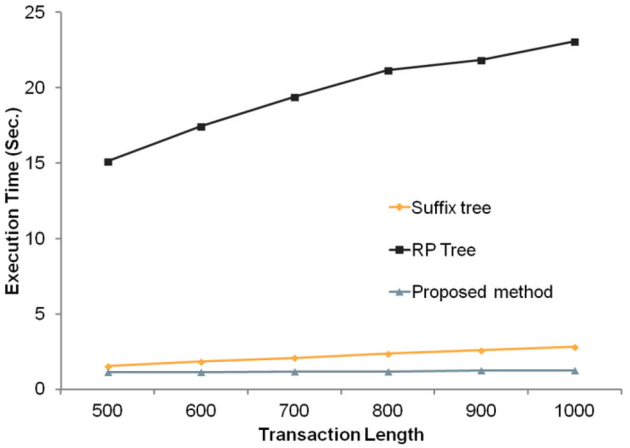

Fig. 12 Execution time of the compared methods under different transaction lengths

5.1 Evaluations for mining of repeating patterns

5.1.1 Experimental settings

In this evaluation, all of the experiments were implemented in C++ and executed on a PC with Intel Core i5-4590 @ 3.3 GHz of CPU and 16 GB of memory. The parameter settings for yielding the experimental data are shown in Table 3. The compared methods with our proposed method are RP tree [9] and Suffix tree [16].

5.1.2 Comparisons of the proposed method and other methods in terms of execution time

The first evaluation we want to show is the performance by varying the transaction length under the data size is 1000 and the number of pattern categories is 200. The related experimental results are shown in Fig. 12 revealing several points. First, the longer the transaction length, the larger the execution time. Second, RP tree performs much worse than the proposed method and Suffix tree. Third, although the performances of our proposed method and suffix tree are pretty close while the transaction length is 500, our proposed method is better than suffix tree, while the transaction length increases. Fourth, the execution time increases slightly for the proposed method. In summary, even facing the larger transaction length, our proposed index can deal with issue of efficiency.

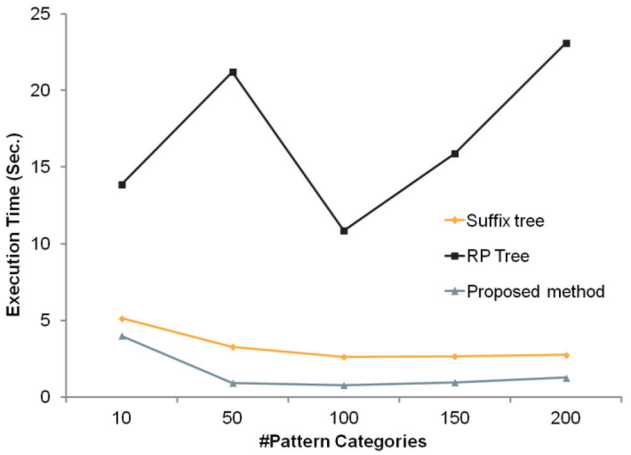

Fig. 13 Execution time of the compared methods under different numbers of pattern categories

The second experiment is to evaluate the proposed method under different numbers of pattern categories. This evaluation was conducted under the data size is 1000 and the transaction length is 1000. Figure 13 depicts the experimental results which delivers some aspects. First, the best method is our proposed method and the worst method is RP tree. Second, as the number of pattern categories increases, the execution time decreases. Note that, this result is limited in the same data size and transaction length. This is because the occurrences of patterns reduce significantly, so that the number of repeating patterns thereby reduces. Hence, few patterns decrease the execution time. The final empirical evaluation for mining algorithm is to show the impact of the data size. This experiment was made under the transaction length is 1000 and the number of pattern categories is 200. Figure 14 shows the experimental results that the execution time of RP tree increases significantly in contrast to the other compared methods. It says that RP tree needs much effort to identify the repeating patterns for a large amount of data. To sum up, if the parameter value is small, the execution time of our proposed method and suffix tree is pretty close. On the contrary, our proposed method performs more stably than the suffix tree if enlarging the data parameter values. In overall, whatever the data is, our proposed method performs better than other compared methods.

5.2 Evaluations for repeating-pattern-based music recommendation

In the previous section, the efficiency of algorithm for mining the repeating patterns has been shown. Another issue in this paper is to convince the readers of usage of repeating patterns. Therefore, our intent for the following evaluations is to reveal how useful the repeating patterns are.

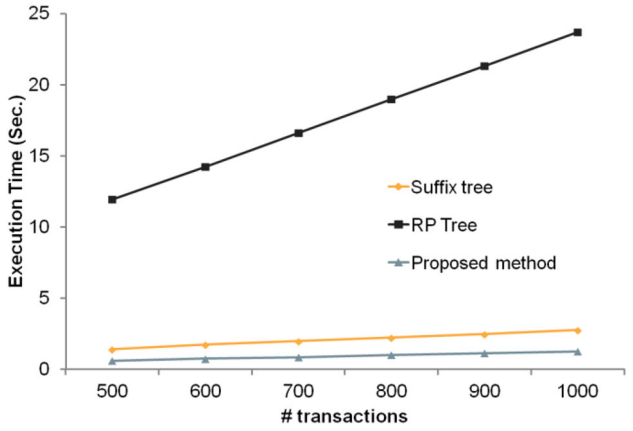

Fig. 14 Execution time of the compared methods under different numbers of transactions

Table 4 Rating densities for different divided subsets

	Data 1	Data 2	Data 3	Data 4
Rating density	10%	7.5%	5%	2.5%

5.2.1 Experimental settings

The experimental data were gathered from the Web, which contain 30 genres. Each genre consists of 20 pieces. In this evaluation, we implemented a real online rating system and 50 volunteers were invited to vote the music items. In detail, for each volunteer, around 75 items were randomly selected from 600 items to vote. Furthermore, because one of the major goals for recommendation is to deal with problem of data sparsity, we decomposed the experimental data into four subsets, namely, data 1, data 2, data 3, and data 4. Table 4 shows the rating densities of different data sets. For data 1, data 2, data 3, and data 4, 15, 30, 45, and 60 known items of each user were randomly selected from original 75 items, respectively, as the unknown ratings to predict. That is, the density decreases from data 1 to data 4. In addition to the data settings, the evaluation measure is RMSE (Root Mean Square Error), which is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{|\text{test}|} (v - \hat{v})^2}{|\text{test}|}}, \tag{1}$$

where v denotes the ground truth, $\hat{v}$ denotes the predicted rating value, and *test* denotes the testing data set. The lower the RMSE, the higher the quality. Note that, if $\hat{v}$ cannot be predicted, $(v - \hat{v})$ is 5.

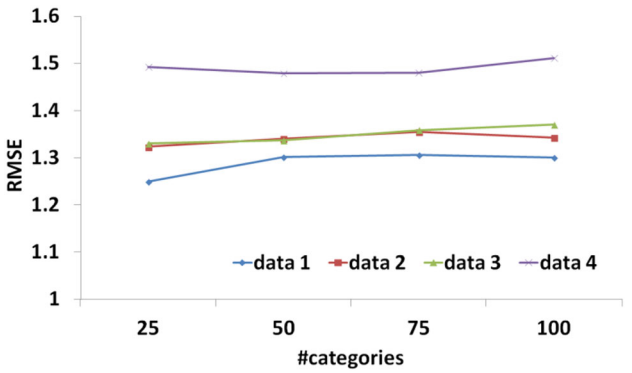

Fig. 15 RMSEs of different music data sets under different numbers of pattern categories

5.2.2 Evaluations for different numbers of patterns on different data sets

Before showing the comparisons with other methods, the effectiveness for different numbers of patterns of different music data sets needs to be clarified. Table 5 shows that the music data are transformed into different kinds (categories) of patterns and each music data set contains different numbers of patterns. The experimental results are shown in Fig. 15 that depicts some points. First, the RMSEs change slightly as the numbers of pattern categories change. It is because the pattern number changes unclearly, referring to Table 5. Second, from data 1 to data 4, the RMSEs increase. It says that, the higher the sparsity, the higher the RMSE. In the following evaluations, the best RMSEs were selected to conduct the overall comparisons.

5.2.3 Comparisons with other methods on different rating data sets

Actually, the sparse data appear frequently in real applications. Hence, the goal of this evaluation is to know how effective the proposed recommender system is if facing very sparse data. In this evaluation, two state-of-the-art methods for music recommendation were compared with our proposed repeating-pattern-based music recommender system (called RPMR), including user-based [14] and item-based [19] recommender systems. Figure 16 shows the experimental results, which delivers some perspectives. First, our proposed method outperforms the compared methods if the data sparsity is large. Second, as the sparsity increases, the compared methods increase significantly. In contrast, our proposed method performs very stably even the sparsity increases. It justifies a point that our proposed method is very effective to cope with problem of data sparsity. The primary reason is that traditional recommender systems need known ratings to calculate the item ratings. Unfortunately, most item

Table 5 #Patterns of different #categories of patterns for music

	#Categories: 25	#Categories: 50	#Categories: 75	#Categories: 100
#Patterns	41,785	43,563	40,806	38,692

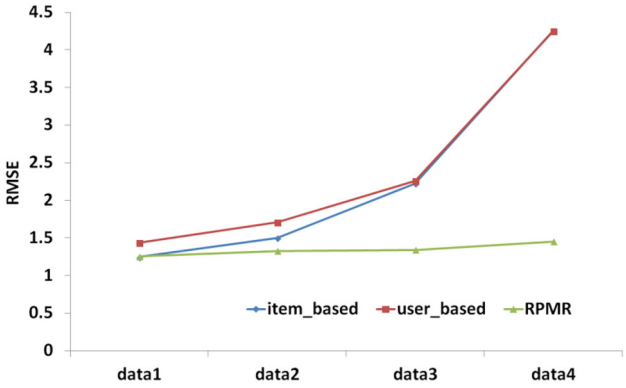

Fig. 16 RMSEs of compared methods on different data sets

ratings in real applications cannot be calculated if the known ratings are very few. On the contrary, our proposed method needs only few ratings to accomplish the rating predictions. In addition, the temporal continuity hidden in repeating patterns is considered as a listening feature in this method. In other words, the temporal continuity and the user preference are connected successfully in the proposed idea.

6 Conclusion and future work

Frankly speaking, it is not easy to discover the valuable knowledge from our real life. For this issue, the past studies made on knowledge discovery can be categorized two classifications, namely, algorithm-driven and application-driven studies. Algorithm-driven studies focused on how to increase the efficiency, while the application-driven ones focused on how to successfully apply the mining algorithms to applications. In this paper, on one hand, we propose an efficient mining algorithm to discover the repeating patterns. On the other hand, an application for music recommendation is lifted to justify the motivation of pattern mining. In terms of mining algorithm, first, we construct an informative index called QPI containing positions and occurrences of patterns. It can effectively reduce the cost of mining of repeating patterns. By QPI, a prefix-search-based algorithm named FMRP performs efficiently on mining the repeating patterns. In terms of application, a music recommender system based on the repeating patterns is proposed to cope with problems in traditional recommender systems. The main intent is to link the user preferences and the repeating patterns on music. The experimental results reveal that our proposed mining algorithm is more efficient than the compared methods on different data. Moreover, the proposed recommender system based on the repeating patterns performs more effective in facing very sparse data. This is just the beginning of applying the repeating patterns. In the future, the repeating patterns will further be used as representative features in recognizing objects in other real applications.

Acknowledgements This research was supported by Ministry of Science and Technology, Taiwan, R.O.C. under grant no. MOST 105-2221-E-230-011-MY2 and MOST 105-2632-S-424-001.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Bobadilla, J., Hernando, A., Ortega, F., Gutiérrez, A.: Collaborative filtering based on significances. Inf. Sci. **185**(1), 1–17 (2012)
2. Cooper, M., Foote, J.: Automatic music summarization via similarity analysis. In: The 2002 IEEE International Conference on Music Information Retrieval, pp. 81–85 (2002)
3. Cheng, R., Tang, B.: A music recommendation system based on acoustic features and user personalities. In: The Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 203–213 (2016)
4. Deshpande, M., Karypis, G.: Item-based top-*N* recommendation algorithms. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
5. Friedman, S., Stamos, I.: Online detection of repeated structures in point clouds of urban scenes for compression and registration. Int. J. Comput. Vis. **102**(1–3), 112–128 (2013)
6. Gool, L.V., Zeng, G., Wonka, P., Muller, P.: Image-based procedural modeling of facades. In: The ACM SIGGRAPH Conference on Computer Graphics, pp. 63–130 (2007)
7. Han, B.J., Hwang, E., Rho, S.: An efficient voice transcription scheme for music retrieval. In: The 2007 IEEE International Conference on Multimedia and Ubiquitous Engineering, pp. 28–26 (2007)
8. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: The International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237 (1999)
9. Hsu, J.L., Liu, C.C., Chen, L.P.: Discovering non-trivial repeating patterns in music data. IEEE Trans. Multimedia **3**(3), 311–325 (2001)
10. Liu, J., Psarakis, E., Stamos, I.: Automatic Kronecker product model based detection of repeated patterns in 2D urban images. In: The 2013 IEEE International Conference on Computer Vision, pp. 401–408 (2013)
11. Ma, Y.F., Lu, L., Zhang, H.J., Li, M.J.: A user attention model for video summarization. In: The Tenth ACM International Conference on Multimedia, pp. 533–542 (2002)

12. Peng, J., Zeng, D.D., Zhao, H., Wang, F.: Collaborative filtering in social tagging systems based on joint item-tag recommendations. In: The ACM International Conference on Information and Knowledge Management, pp. 809–818 (2010)
13. Qi, Q., Chen, Z., Liu, J., Hui, C., Wu, Q.: Using inferred tag ratings to improve user-based collaborative filtering. In: Annual ACM Symposium on Applied Computing, pp. 2008–2013 (2012)
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: The International Conference on ACM Computer Supported Cooperative Work, pp. 175–186 (1994)
15. Rahman, M.S., Rahman, M.S., Chowdhury, S.U.I., Mahmood, A., Rahman, R.M.: A personalized music recommender service based on fuzzy inference system. In: The IEEE/ACIS 15th International Conference on Computer and Information Science (2016)
16. Singh, A.: Ukkonen's Suffix Tree Construction. (2014) Retrieved from http://www.geeksforgeeks.org/ukkonens-suffix-tree-construction-part-6/
17. Su, J.-H., Chang, W.-Y., Tseng, V.S.: Personalized music recommendation by mining social media tags. In: The 17th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, pp. 291–300 (2013)
18. Su, J.-H., Chang, W.-Y., Tseng, V.S.: Effective social content-based collaborative filtering for music recommendation. Intell. Data Anal. **21**(S1), S195–S216 (2017)
19. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: The International Conference on World Wide Web, pp. 285–295 (2001)
20. Ukkonen, E.: On-line construction of suffix tree. Algorithmica **14**(3), 249–260(1995)
21. Wang, M., Lu, L., Zhang, H.H.: Repeating pattern discovery from acoustic musical signals. In: The 2004 IEEE international conference on multimedia and expo, vol. 3, pp. 2019–2022 (2004)
22. Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: The Twenty-Sixth International Joint Conference on Artificial Intelligence (2017)
23. Xiao, R.G., Wang, Y.Y., Pan, H., Wu, F.: Automatic video summarization by spatio-temporal analysis and non-trivial repeating pattern detection. In: The 2008 IEEE Congress on Image and Signal Processing, pp. 555–559 (2008)
24. Zhao, P., Fang, T., Xiao, J., Zhang, H., Zhao, Q., Quan, L.: Rectilinear parsing of architecture in urban environment. In: The 2010 IEEE Computer Vision and Pattern Recognition, pp. 342–349 (2010)
25. https://www.csie.ntu.edu.tw/~cjlin/liblinear/. Accessed 5 Aug 2017

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.