Linear Time for Discovering Non-trivial Repeating Patterns in Music Databases

Yu-lung Lo Wen-lin Li
Department of Information Management
Chaoyang University of Technology
168, GiFeng E.Rd., WuFeng TaiChung County, Taiwan 413, ROC
yllo@cyut.edu.tw *s9214611@mail.cyut.edu.tw*

Abstract

A repeating pattern is commonly used in analyzing the repeated part of music data and looking for themes. Most of the repeating patterns are key melodies or easy to familiarize and remember for people. Therefore, we can use the themes or the repeating patterns to construct indices that can speedup music retrieval. Non-trivial repeating patterns exclude those patterns, which are all contained in other longer patterns, such that they can reduce the redundancy of the repeating patterns and save the index space. Most of existing algorithms are time consuming for finding non-trivial repeating patterns. In this paper, we propose an approach for efficiently discovering non-trivial repeating patterns in linear time.

Keywords: context-based retrieval, music database, multimedia database, non-trivial repeating pattern, suffix tree

1. Introduction

In the researches of music content-based retrieval, we can extract the features, such as melodies, rhythms and chords, from the music data and develop indices that will help to retrieve the relevant music quickly. Most reports point out that these music features can be presented in string formats then develop efficient string indices for retrieving music data [2] [3] [4] [5] [6] [8] [9] [12] [16] [19] [20] [18] . We can find some sequence of notes appeared more than one time in music data, which are called the repeating patterns. For example, the pattern in Little Bee Song -- "so-mi-mi-fa-re-re" called the repeating pattern that repeatedly appears three times in the song. A repeating pattern is commonly used in analyzing the repeated part of music data and looking for themes [8] [9] [13] [18] . A lot of researches in musicology and music psychology consent that the repeating pattern is one of general features in music structure modeling [1] [10] [15] [17] . Non-trivial repeating patterns exclude those patterns, which are all contained in other longer patterns. The repeating pattern and non-trivial repeating pattern are defined in the following:

Repeating pattern: In string S, there is a sub-string appearing more than once and its length being equal to or greater than 2 symbols, which is called a repeating pattern of the string S.

Non-trivial repeating pattern: The frequency of the sub-string X appearing in the string S is more than it is appearing in any other sub-strings of the same string S [9] [12] .

Generally, most of non-trivial repeating patterns are key melodies or themes of a music object, which is not easy to forget for most of people. Therefore, discovering the non-trivial repeating patterns of a music object then creating the music index can speedup the searching of queries in music databases [9] [13] . In this paper, we propose an approach which can efficiently and effectively discover the non-trivial repeating patterns in a music object.

This paper consists of five sections organized as follows: in section 2, we briefly describe the existing non-trivial repeating pattern discovering techniques for music data. In section 3, we introduce the constructing of suffix tree in linear time which the idea of our approach comes from. Section 4 presents our proposed scheme which can discover non-trivial repeating patterns in linear time. Finally, we make a conclusion and point out the possible future application in the last section.

0-7803-8603-5/04/$20.00 ©2004 IEEE

2. Existing techniques

Recently, several schemes have been proposed to discover non-trivial repeating patterns in music data. Tseng [18] developed the music search system and proposed an approach of key melody extraction to discover the repeating patterns. They compute the frequency of the repeating pattern by combining the adjacent musical symbols to form 2 to 3 musical symbols and up to the longest of the repeating pattern sequentially. Hsu et al. [8] proposed the Correlative Matrix approach which can be done by creating an upper-right-triangular matrix and based on the length of the music object. However, the RP-tree proposed by Hsu et al.[9] [12] and Adaptive Correlative Matrix proposed by Lo et al. [13] are more efficiently for finding non-trivial repeating patterns. We briefly describe these approaches as follows.

2.1 RP-tree

RP-tree [9] [12] uses the string-join approach and consists of several main steps: for string S

(1) By using the string-join approach, discovering all strings are with length of 2^k (2^k , $k \geqq 0$ and $2^k \leqq |S|$) then store the strings in a string set. The string set is with the format of $\{X, Freq(X), (position\_1, position\_2, \ldots)\}$, where X is a repeating pattern in string S, $Freq(X)$ denotes the frequency of X appeared in string S, and $(position\_1, position\_2, \ldots)$ indicates the positions of the string appeared in the string S.
(2) Finding out the longest repeating pattern with length 2^k.
(3) Constructing RP-tree.
(4) Eliminating the trivial repeating patterns in RP-tree.
(5) Finding out all repeating patterns of the length not 2^k and re-constructing RP-tree.
(6) Finally, producing all non-trivial repeating patterns.

This approach is more efficiently than Correlative Matrix approach [8] .

2.2 Adaptive correlative matrix

The authors of this paper, Lo et al., improved the Correlative Matrix approach [8] to propose the Adaptive Correlative Matrix (ACM) [13] . The basic idea is as follows.

(1) Initially, we construct a correlative matrix M same as in [8] and only compute the upper-right-triangular of M.
(2) Let $M_{i,j}$ denote the value of the cell located at i-th row and j-th column of M. If there exists any i and j, such that $M_{i,j} >= 2$, there is a repeating pattern found.
(3) To compute the pattern repeating, we can search only the i-th row and j-th column of the upper-right-triangular of the matrix M for discovering how many values in cells are greater or equal to $M_{i,j}$.

The experimental study shows that performance of this approach is better than RP-tree [13] . However, this technique still needs $O(m^2)$ time where m is the length of the string S.

3. Linear-time suffix tree algorithm

In this section, we describe the linear-time suffix tree algorithm which the idea of your proposed approach comes from. A suffix tree [14] [22] is a tree-like index structure representing all suffixes of a string and provided solutions for string matching problems. Weiner was the first to show that suffix trees can be built in linear time [7] [22] . However, Ukkonen's method is equally fast and uses far less space in practice [7] [21] . The main steps of Ukkonen's algorithm are as follows.

Implicit suffix tree: Ukkonen's algorithm constructs an *implicit suffix tree* for string S. The implicit suffix tree is a tree obtained from the suffix tree for S$ by removing every copy of the terminal symbol $ from the edge labels of the tree.

Edge-label compress: Since an $O(m)$ time algorithm for building suffix trees requires some alternate scheme to represent the edge-labels. Instead of explicitly writing a substring on an edge of the tree, Ukkonen's algorithm only write a pair of indices on the edge, specifying beginning and end positions of that substring in S.

Tricks: Ukkonen's algorithm uses suffix links and applies three tricks of extension rules to reduce the traversal time to something proportional to the number of nodes on the path of suffix tree.

Creating the true suffix tree: The final implicit suffix tree can be converted to a true suffix tree by adding a string terminal symbol '$' to the end of string S.

Completing all these steps, Ukkonen has proofed that the algorithm can build a true suffix tree along with all its suffix links in $O(m)$ time [7] [22] .

4. Linear-time for discovering non-trivial repeating patterns

Usually, to construct a suffix tree of a string S with a terminal symbol '$' take $O(m^2)$ time [7] [22] . Suppose that S is "xabxa", the result suffix tree is shown in Figure 1. Ukkonen's algorithm can build an implicit suffix tree for $S$$ by removing every copy of the terminal symbol $ from the edge labels of the tree as shown in Figure 2. As claimed in [7] [21] this can take only $O(m)$ time.

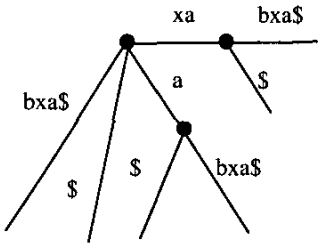

Figure 1. suffix tree for string "xabxa"

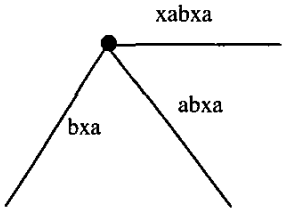

Figure 2. implicit suffix tree for string "xabxa"

Based on the implicit suffix tree, the linear-time algorithm for discovering non-trivial repeating patterns can be derived as follows. If we add a stop symbol '#' into the tail of S, where '#' considered as a regular symbol in S, the implicit suffix tree for string S# can be constructed by Ukkonen's algorithm as shown in Figure 3.

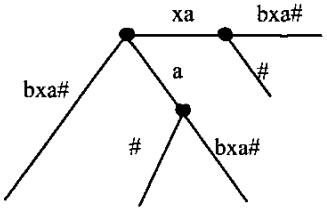

Figure 3. implicit suffix tree for S# (="xabxa#")

From the implicit suffix tree, it can be easily found out that any non-leaf nodes, excepting root node, with the number of out going edges greater than or equal to 2, the label in the incoming edge of that node should be a non-trivial repeating pattern. For example, as shown in Figure 3, there are two nodes following edge labels "xa" and "a" with degree 2. Since the length of a repeating pattern has to be equal to or greater than 2 symbols, "xa" is a non-trivial repeating pattern of S. Furthermore, suppose that a fragment of an implicit suffix is shown in Figure 4. From this implicit suffix tree, we can conclude that the non-trivial repeating patterns found are not only "ab" but also "abcd". It is because that "ab" appears at least three times and "abcd" appears at least two times. That is the repeating of a pattern which can be computed by summing up the degree of a non-leaf node and its children.

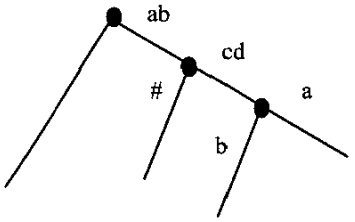

Figure 4. a fragment of an implicit suffix tree

As described in section 3, Ukkonen's algorithm can build the implicit suffix tree in a linear time. Since computing the degrees of non-leaf nodes in an implicit suffix tree can be easily implemented by a recursive algorithm in a linear time, such as postfix order traverse in a tree, to sum up the repeating (or degree) for each non-leaf node also can be done in a linear time. While most of existing techniques take $O(m^2)$ to discover non-trivial repeating patterns, obviously, our new proposed algorithm spends only $O(m)$.

5. Conclusions

The non-trivial repeating patterns are usually the themes in a music object. If we can create an index for music database by using these repeating patterns, it may effectively help the searching for music queries. In this paper, a linear time for discovering non-trivial repeating pattern has been proposed. The idea of our proposed scheme is based on an implicit suffix tree constructed by Ukkonen's algorithm and we also can travel the implicit suffix tree in a linear time to discover the non-trivial repeating pattern in a music

object. While most of existing non-trivial repeating patterns finding techniques take $O(m^2)$ time, the new proposed approach spends only $O(m)$ time. We expect that this approach can be used to discover not only repeating patterns on a music object but also repeating sequences on the DNA and protein.

6. References

[1] Bakhmutova, V., Gusev, V. D., and Titkova, T. N., "The Search for Adaptations in Song Melodies," *Computer Music Journal*, vol. 21, no. 1, 1997, pp. 58-67.

[2] Blackburn, S., and DeRoure, D., "A Tool for Content-based Navigation of Music," *In Proc. of ACM Multimedia*, 1998, pp. 361-368.

[3] Chen, J. C. C., and Chen, A. L. P., "Query by Rhythm An Approach for Song Retrieval in Music Databases," *In Proc. of the 8th Int'l Workshop on Research Issues in Data Engineering*, 1998, pp. 139-146.

[4] Chen, A. L. P., Chang, M., Chen, J., Hsu, J. L., Hsu, C.H., and Hua, S. Y. S., "Query by Music Segments: An Efficient Approach for Song Retrieval," *In Proc. of IEEE Int'l Conf. on Multimedia and Expo*, 2000, pp. 873-876.

[5] Chou, T. C., Chen, A. L. P., and Liu, C. C., "Music Databases: Indexing Techniques and Implementation," *In Proc. of IEEE Int'l Workshop on Multimedia Data Base Management Systems*, 1996, pp. 46-53.

[6] Logan Ghias, J., Chamberlin, D., and Smith, B. C., "Query by Humming: Musical Information Retrieval in an Audio Database," *In Proc. of ACM Multimedia*, 1995, pp. 231-236.

[7] Gusfield, D., "Algorithms on Strings, Trees, and Sequences:computer sience and computational biology," *The Press Syndicate of The University of Cambridge*, New York, 1997.

[8] Hsu, J. L., Liu, C. C., and Chen, A. L. P., "Efficient Repeating Pattern Finding in Music Databases," *In Proc. of ACM Int'l Conf. on Information and Knowledge Management*, 1998, pp. 281-288.

[9] Hsu, J. L., Liu, C. C., and Chen, A. L. P., "Discovering Non-Trivial Repeating Patterns in Music Data," *IEEE Trans. on Multimedia*, vol. 3, no. 3, 2001, pp.311-325.

[10] Krumhansl, C. L., "Cognitive Foundations of Musical Pitch," *Oxford University Press*, New York, 1990.

[11] Liu, C. C., Hsu, J. L., and Chen, A. L. P., An "Approximate String Matching Algorithm for Content-Based Music Data Retrieval," *In Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems*, 1999, pp. 451-456.

[12] Liu, C. C., Hsu, J. L., and Chen, A. L. P., "Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases," *In Proc. of IEEE Data Engineering*, 1999, pp. 14-21.

[13] Lo, Y. L., Yu, H. C., and Fan, M. C., "Efficient Non-trivial Repeating Pattern Discovering in Music Databases," *Tamsui Oxford Journal of Mathematical Sciences*, vol. 17, no. 2, 2001, pp. 163-187.

[14] McCreight, E., "A Space-Economical Suffix Tree Construction Algorithm," *journal of the ACM*, vol. 23, no. 2, 1976, pp. 262-272.

[15] Narmour, E., "The Analysis and Cognition of Basic Melodic Structures," *the University of Chicago Press*, Chicago, 1990.

[16] Pfeiffer, S., Fischer, S., and Effelsberg, W., "Automatic Audio Content Analysis," *In Proc. of ACM Multimedia Conference*, 1996, pp. 21-30.

[17] Sundberg, J., Friberg, A., and Fryden, L., "Common Secrets of Musicians and Listeners: An Analysis-by-Synthesis Study of Musical Performance, In Representing Musical Structure," *Academic press*, London,1991.

[18] Tseng, Y. H., "Content-Based Retrieval for Music Collections," *In Proc. of ACM SIGIR'99*, 1999, pp. 176-182.

[19] Uitdenbogerd, A. L. and Zobel, J., "Manipulation of Music for Melody Matching," *In Proc. of ACM Multimedia*, 1998, pp. 235-240.

[20] Uitdenbogerd, A. L. and Zobel, J., "Melodic Matching Techniques for Large Music Databases," *In Proc. of ACM Multimedia*, 1999, pp. 57-66.

[21] Ukkonen, E., "On-Line Construction of Suffix Tree," *Algorithmica*, vol. 14, 1995, pp. 249-260.

[22] Weiner, P., "Linear Pattern Matching Algorithms," *In Proc. of the 14th Ann. Symp. on Switching and Automata Theory*, 1973,pp. 1-11.