A Novel Representation of Sequence Data based on Structural Information for Effective Music Retrieval

Chia-Hsiung Lee, Chung-Wen Cho, Yi-Hung Wu, and Arbee L. P. Chen*

Department of Computer Science, National Tsing Hua University,
Hsinchu, Taiwan 300, R.O.C.
alpchen@cs.nthu.edu.tw

Abstract. In this paper, we propose a novel representation of sequences based on the structural information of the sequences. A sequence is represented by a set of rules, which are derived from its subsequences. There are two types of subsequences of interest. One is called frequent pattern, which is a subsequence appearing often enough in the sequence. The other is called correlative pattern, which is a subsequence composed of highly correlated elements. The rules derived from the frequent patterns are called frequent rules, while the ones derived from the correlative patterns are called correlative rules. By considering music objects as sequences, we represent each of them as a set of rules and design a similarity function for effective music retrieval. The experimental results show that our approaches outperform the approaches based on the Markov-model on the average precision.

Keywords: Music retrieval, Music representation, Sequential pattern, Markov-model.

1 Introduction

A *sequence* is an ordered list of events, with or without concrete notions of time [7]. In recent years, there has been an enormous growth in the amount of sequence data, such as customer transactions, music data, and DNA sequences. The analysis of such data is useful for various applications, e.g. customer purchase behavior prediction [2], music classification and retrieval [3], [4], etc. Therefore, providing an effective way to analyze the larger amount of sequence data has become one of the most important issues nowadays.

Similarity search [1], [6], [10], in a sequence database is one of the most important topics in the analysis of sequence data, which consists of two major issues. First, for each sequence, how to extract representative features from its contents? Second, based on the features extracted, how to measure the similarity between two sequences? Both issues are often addressed together and depend on the characteristics of the sequence data in various applications.

In this paper, we propose a novel approach to represent a sequence by its subsequences. For the ease of presentation, we call the subsequence of a sequence the *pat-*

* To whom all correspondence should be sent.

tern and a pattern with k elements the *k-pattern*. Given a sequence database, the *support* of a pattern stands for the percentage of the sequences containing it in the entire database. A pattern is *frequent* if its support is not below the minimum support threshold *minsup*. In our approach, the representation of a sequence is derived in three steps. First, for a sequence with length n, we slide a window with size w ($w \leq n$) over this sequence to decompose it into a set of segments, where a segment is one of the n consecutive subsequences extracted by the sliding windows. In this way, a sequence is transformed into a segment database that contains all the segments extracted from this sequence.

Second, two types of patterns are derived from the segment database. The first type is the frequent pattern. Given a sequence and a sliding window with size w, a pattern with m elements ($m \leq w$) will repeatedly appear in (w-m+1) segments. In this case, the support of a pattern will be overestimated if the support definition of the frequent pattern is adopted. Therefore, the support of a frequent pattern is defined as the percentage of segments in the entire segment database, which contain this pattern and have the same first element as the pattern has. For instance, among the segments <abcd>, <bcd>, and <cde>, only <cde> supports the pattern <cd>.

Given two patterns α and β, we have a rule in the form of $\alpha \rightarrow \beta$ where α and β are called *predicate* and *consequent*, respectively. The *confidence* of rule $\alpha \rightarrow \beta$ is defined as the support of $\alpha\beta$ divided by the support of α. We call this rule the *frequent rule* if α, β, and $\alpha\beta$ are frequent patterns and its confidence is not below the minimum confidence threshold *minconf*.

Representing a sequence based on the frequent rules may have two limitations. First, only the frequent patterns can form frequent rules. Therefore, it lacks the correlation between frequent patterns and non-frequent ones. Second, the minsup is a fixed value and the support of a long sequence tends to be small. Therefore it is more difficult for a long sequence to be a frequent pattern. For a pattern α and an element X, we call a rule in the form of $\alpha \rightarrow X$ the *primary rule* and observe the following property.

Property 1. The primary rule $a_1 \ldots a_{k-1} \rightarrow a_k$ has the highest confidence among all the rules that exactly cover the sequence $<a_1 a_2 \ldots a_k>$.

This property motivates us to find a novel kind of patterns, which is composed of highly correlated elements and called the *correlative pattern*. The *correlative 1-pattern* has the same definition as that of a frequent 1-pattern, i.e., using the minsup to filter out the 1-patterns with small supports. For any (k-1)-pattern α and element X, where $k \geq 2$, αX is the *correlative k-pattern* if the confidence of $\alpha \rightarrow X$ is not below the minconf. That is, α is highly correlated to X. Furthermore, given a correlative k-pattern $\beta\gamma$ where β and γ are patterns, we call the rule $\beta \rightarrow \gamma$ the *correlative rule* if its confidence is not below the minconf. By this definition, the primary rule whose confidence satisfies the minconf is also a correlative rule.

Comparing the two types of patterns, the correlative pattern can reveal the structural information hidden in the sequence that cannot be found by the frequent patterns, and vice versa. A pattern is frequent but not correlative if its elements are weakly correlated. On the contrary, a pattern is correlative but not frequent when its support is not large enough but its elements are strongly correlated.

After mining patterns from the segment database, the last step of our approach is to provide a representation for sequences based on the two types of patterns. In this paper, we propose the characteristic matrix to keep either all the frequent rules or all the correlative rules derived from a sequence. Based on this representation, we design the similarity measure for effective music retrieval. The experimental results show that our approach based on the frequent rules can achieve 23% improvements on the average precision to the approach based on the Markov-model [16], while our approach based on correlative rules can achieve 56% improvement.

The rest of this paper is organized as follows. In Section 2, we present the related work. In Section 3, we introduce the methods to derive the correlative patterns and the correlative rules. In Section 4, we describe the characteristic matrix and the similarity measure for music retrieval. Section 5 shows the performance evaluation and experimental results. Finally, we conclude this paper with future work in Section 6.

2 Related Work

The sequence representations presented in the literature can be classified into two types, i.e. global representation and local representation. For each sequence in a sequence database, the global representation considers the information of all the sequences, while the local representation concerns the information of the individual sequences themselves.

In the approaches of global representation, the subsequences contained in a large portion of the sequence database are often adopted as the features to represent the sequences [5], [11], [14]. After representing a sequence by frequent subsequences contained in it, the similarity between two sequences can be computed.

In [14], all the frequent subsequences found are used as the features to divide the sequences into clusters. In [11], the frequent subsequences are utilized for sequence classification, where only the frequent subsequences contained in only one class of sequences are selected as the features. In [5], two types of frequent subsequences are respectively selected for sequence clustering. One is the frequent subsequence that is not a subsequence of any other frequent subsequence, while the other is the frequent subsequence satisfying the following constraint: One of its occurrences is not contained by that of any other frequent subsequence in a sequence.

The representation based on frequent subsequences has two drawbacks as follows. First, the subsequences that are potentially useful in representing a sequence can be non-frequent in the entire database. Second, the sequences that are represented by the same set of frequent subsequences are considered similar without considering the number of occurrences for each frequent subsequence in the individual sequences.

The approaches of local representation can be further classified into two categories according to the information they consider, i.e. the entire sequence and the contextual relationships among elements. In the first category, the typical approach is to compute the optimal alignment between two sequences [11] based on the edit distance [6], i.e. the cost of transforming one sequence into the other by a series of edit operations. The idea behind this approach is to align the two sequences against each other so that the

edit distance between them is minimized. Given one sequence, the edit operations are to insert/delete an element into/from the other sequence, and to substitute an element in the other sequence with the one in itself. The actual costs of the edit operations depend on the requirements of applications.

The approach based on optimal alignment has the following two drawbacks. First, the difference between sequence lengths has a great impact on the required number of edit operations. For example, the edit distance between a sequence and a much shorter one cannot be small even if the sequence contains the shorter one. Second, the amount of partial matches between two sequences is also ignored.

In the second category of local representation, Pickens et al. [15], [16] proposed to utilize the Markov-model [10] for representing sequences. For a sequence, the contextual relationships among its elements are collected to compute the transition probability, i.e. the probability that an element will appear immediately after a specific consecutive subsequence. A consecutive subsequence of a sequence is a subsequence whose elements contiguously appear in the sequence. The transition probabilities are then stored into a matrix so that the similarity between two sequences can be estimated from the difference between their matrices.

The representation based on transition probabilities has two drawbacks. First, given a sequence, this approach only reveals the contextual relationships among the consecutive subsequences instead of the non-consecutive ones, which can be potentially useful in representing the sequence, e.g. the main melody interleaved with grace notes. Second, this approach ignores the frequency information, which may indicate how important a subsequence is.

The other approaches of local representation are based on repeating patterns, the consecutive subsequences of a sequence that appear more than once in that sequence [7]. This approach represents each sequence by the repeating patterns discovered from the sequence itself. In this way, the similarity between two sequences can be estimated by comparing their repeating patterns. Compared with the Markov-model approach, the representation based on repeating patterns takes the frequency information into consideration but still lacks for the contextual relationships among the inconsecutive elements.

3 Correlative Patterns and Correlative Rules

In this section, we first present the algorithm for mining correlative patterns and correlative rules from a sequence. After that, the qualitative analysis of correlative patterns is discussed.

Before introducing the mining algorithm, we illustrate the way to transform a sequence into a segment database as follows. Given a sequence $<s_1 s_2 \ldots s_n>$ and a window size w, each subsequence $<s_i s_{i+1} \ldots s_{i+w'-1}>$ forms the i^{th} segment in the segment database, where w' is equal to w if $i+w-1 \leq n$, or $n-i+1$ otherwise. For example, for <bdbcdbcadb> and window size 4, the segment database is as shown in Table 1.

Table 1. The segment database of <bdbcdbcadb>

SID	Segment	SID	Segment
S1	<bdbc>	S6	<bcad>
S2	<dbcd>	S7	<cadb>
S3	<bcdb>	S8	<adb>
S4	<cdbc>	S9	<db>
S5	<dbca>	S10	<b>

We adopt the bottom-up approach [2] to find the correlative patterns in passes, where the correlative k-patterns are discovered in pass k. As described before, a correlative (k+1)-pattern is extended from a correlative k-pattern via the examination of the corresponding primary rule. We define such an extension process as follows.

Definition 1. Given a k-pattern α and the set of elements Σ, we call αX where $X \in \Sigma$ the extended pattern of α. The set of all the extended patterns of α are denoted as E_α.

In the first pass, the correlative 1-patterns, i.e. the frequent 1-patterns, are found by one scan of segment database. In pass k ($k \geq 2$), for each correlative (k-1)-pattern α, the patterns in E_α are regarded as the candidates of correlative k-patterns and called the *candidate k-patterns*. The segment database is then scanned once to compute all the supports of candidate k-patterns. For each segment, we enumerate all the k-patterns that start at the first element of this segment for support computation. For example, considering the segment S_1=<bdbc> in Table 1, in pass 2, only the 2-patterns <bd>, <bb>, and <bc> are enumerated for support computation. In this way, all the correlative k-patterns are discovered and denoted as CP_k. After the supports of candidate k-patterns are computed, the confidences of the primary rules corresponding to these candidate k-patterns are examined. The following lemma guarantees the correctness of the extension process for mining correlative patterns.

Lemma 1. If α is not a correlative pattern, each pattern in $E\alpha$ is not correlative patterns, either.

Proof: (Case 1: α is a 1-pattern) As described before, the elements that are not correlative 1-patterns are never used to extend any correlative pattern. Therefore, their extended patterns cannot be correlative patterns.

(Case 2: α is a k-pattern where k>1) Let α be a sequence $<a_1a_2\ldots a_k>$. Because α is not a correlative pattern, the rule $a_1a_2\ldots a_{k-1} \rightarrow a_k$ is not a correlative rule, indicating that its confidence, the support of $<a_1a_2\ldots a_k>$ divided by the support of $<a_1a_2\ldots a_{k-1}>$, is below the minconf. For each pattern αX in E_α, the confidence of the rule $\alpha \rightarrow X$ is equal to the support of αX divided by the support of α.

$\because$ support of $\alpha X \leq$ support of $\alpha \leq$ support of $<a_1a_2\ldots a_{k-1}>$

$\therefore$ confidence of $\alpha \rightarrow X \leq$ confidence of $a_1a_2\ldots a_{k-1} \rightarrow a_k <$ minconf

Therefore, αX cannot be a correlative pattern. This lemma is proved.

Whenever a correlative k-pattern is found, all the rules (except for the primary rule) over it are examined for finding the correlative rules. When no more correlative k-patterns are generated, the mining algorithm terminates. To compute the confidence of a rule, we need the supports of the corresponding pattern and its predicate. The support of a predicate has been computed in the previous passes. Therefore, in pass k, we simply compute the confidences of all the rules.

```
Algorithm CPR
Input: A sequence S, the set of elements Σ, window size
w, minsup, minconf
Output: All the correlative rules CR and the correla-
tive patterns CP1, CP2, …
Variable: The segment database SD derived from S, Pat-
tern set P, Rule sets R1, R2
(1)       SD←Segmentation (S, w)
(2)       k=1
(3)       CP1←Support-Computation (SD, Σ)
(4)       CP1←Pattern-Pruning (CP1, minsup)
(5)       While (CPk≠∅) do
(6)         P←Extended-Pattern-Generation (CPk, Σ)
(7)         P←Support-Computation (SD, P)
(8)         R1←Primary-Rule-Generation (P)
(9)         R1←Confidence-Computation ( R1, minconf)
(10)        For each rule α→X in R1 do
(11)          Add α→X into CR
(12)          Add αX into CPk+1
(13)          R2←Rule-Derivation (αX)
(14)          R2←Confidence-Computation (R2, minconf)
(15)        k=k+1
```

The above shows the mining algorithm *CPR*. Initially, *Segmentation* transforms the input sequence into a segment database SD. After that, *Support-Computation* scans SD once to compute the support of each element in the sequence and then *Pattern-Pruning* returns the correlative 1-patterns CP_1. In the main loop, each pass consists of three steps. First, *Extended-Pattern-Generation* generates all the candidate k-patterns by extending each correlative (k-1)-pattern and then the supports of all the candidate k-patterns are computed. Next, *Primary-Rule-Generation* generates all the primary rules from CP_k and then *Confidence-Computation* computes their confidences and returns only those satisfying the minconf. Finally, for each primary rule left, a correlative pattern and all the correlative rules over this pattern are discovered. Note that for a k-pattern, *Rule-Derivation* always returns the k-2 rules over it without the primary rule.

Example 1. Consider the sequence and segment database in Table 1 as an example. Let minsup and minconf be 0.3 and 0.6, respectively. At first, the elements b and d are collected into CP_1. Among the 8 primary rules, only b→c, b→d, d→b, and d→c are correlative rules, which lead to 4 correlative 2-patterns. Similarly, among the 16 primary rules, three of them are correlative rules, corresponding to the correlative 3-patterns <bcd>, <bdb>, and <dbc>. The other rules over the correlative 3-patterns are also examined, e.g. d→bc. Since the 12 primary rules derived from the correlative 3-patterns do not satisfy the minconf, no correlative 4-pattern is generated and the algorithm terminates.

4 Characteristic Matrix

In this section, we present the characteristic matrix, which is composed of either frequent rules or correlative rules derived from a sequence. Moreover, we introduce the similarity measure between two characteristic matrices.

In the characteristic matrix, each entry corresponds to a rule and keeps 1 if it is a frequent (correlative) rule or 0 otherwise. The leftmost column keeps the predicates of the rules, while the topmost row keeps all the consequents. This matrix *owns* a frequent (correlative) rule if it can be derived from the sequence and the corresponding entry keeps 1.

We measure the similarity between two characteristic matrices by following two criteria. The number of common rules they share increases the similarity degree, while the number of different rules decreases the similarity degree. Let the two numbers be denoted as N_{common} and $N_{different}$ respectively. The similarity between two characteristic matrices is computed as formula (1).

$$N_{common} / (N_{common} + N_{different}) . \quad (1)$$

Example 2. Consider the two characteristic matrices M_1 and M_2 shown in Table 2. The common rules are a→b and a→c, while the different rules are a→bc, a→cd, a→d, ab→c, ac→d, b→c, c→b, and c→d. N_{common} and $N_{different}$ are 2 and 8 respectively. Therefore, the similarity between M_1 and M_2 is 0.2.

Table 2. Two characteristic matrices M_1 and M_2

M_1	<b>	<c>	<cd>	<d>
<a>	1	1	1	1
<ac>	0	0	0	1
<c>	1	0	0	1

M_2	<b>	<bc>	<c>
<a>	1	1	1
<ab>	0	0	1
<b>	0	0	1

5 Experiments

With the growth of available music data, it is getting attention on the similarity search in music databases. Considering music data as sequences, our approach is applied to the representation and similarity measure of music data. To evaluate the effectiveness of our approach upon music data, two experiments are performed. First, we consider the similarity search for music variations and compare our approach with the one based on the Markov-model [16]. In the second experiment, we make the quantitative analysis on the two types of rules derived by our approach.

The variations of a music sequence cause a similar auditory sensation to the listeners and often have similar structural information in contents, although the placements of the notes in them can be very different. For example, variations of the same music may have the main melody interleaved with different styles of grace notes. In [16], the authors propose a harmonic description to transcribe polyphonic music, where more than one note can be played at the same time. Furthermore, they represent music sequences by the Markov-model and make experiments on music variations.

The Markov-model approach is briefly introduced as follows. First, each music piece is reduced to a sequence of *simultaneities*, where the simultaneity refers to a set of pitches played at the same time. Second, 24 major and minor triads are selected, including their relative distances, as *chords* according to the musicology. Third, the simultaneity is represented as a 24-dimensional vectors, where each dimension stands for a chord and keeps the probability that the chord contributes to the simultaneity. Fourth, each music sequence is represented as a Markov-model, where each chord is a state and the probabilities kept in the vectors are used to compute the *transition probabilities* between states. For an n^{th}-order Markov-model, a $24^n \times 24$ matrix is constructed, where rows and columns represent the previous states and the current states respectively, and each entry keeps the transition probability. Finally, the similarity between two matrices is based on the *Kullback-Liebler divergence* [14].

In this experiment, we adopt the source collection and query sets provided by Jeremy Pickens [16]. There are around 3000 music pieces in the form of the harmonic description as described above. Besides, three additional sets are randomly selected as queries: 26 variations of 'Twinkle, twinkle, little star', 22 versions of John Dowlands's 'Lachrimae Pavan', and 17 variations of 'Les Folies d'Espagne'. For each selected query, the remaining variations in the same set are regarded as the correct answers. To adapt our approach to polyphonic music, we replace the simultaneity with the chord, which corresponds to the dimension with the highest probability in the vector. Moreover, according to the study in [9], a *phrase* is a meaningful unit of music object and three quarters of phrases contain from 6 to 10 notes. Therefore, the window size is varied from 6 to 10 in the experiments and the results indicate that the window size 8 can yield better precision than the others on average. In addition to the Markov-model approach, we also compare our approach with the one based on the *Time-Invariant Markov-model*, which is a more general version of the Markov-model approach.

Table 3 and 4 respectively show the average precision yielded by the Markov-model approach and our approach based on correlative rules. The average precision is the average of *11-pt precision/recall*. Besides, the number with boldface indicates the

best result achieved by an approach for a query set. In the tables, a row stands for a query set, while a column refers to an approach with a particular parameter setting, e.g., mm1 denotes the first order Markov-model and 6-40 indicates that minsup and minconf are 0.06 and 0.4 respectively.

For the Twinkle queries, the best result of our approach is a little better than that of the Markov-model approach. In this variation set, we find that the music sequences have short lengths on average and the distinct notes are randomly distributed over them. Moreover, the correlative patterns tend to be short. This makes our approach difficult to distinguish the main melody from the grace notes. On the contrary, the other two sets have longer music sequences and lead to a larger number of long patterns. The common characteristics in each variation set are better captured by our approach. Therefore, our approach achieves a higher precision on average.

Table 3. The Markov-model approach

	mm0	mm1	mm2	mm3	timm0	timm1	timm2	*Ave.*
Twinkle query	0.120	0.109	**0.151**	0.133	0.041	0.077	0.110	*0.106*
Lachrimae query	**0.214**	0.142	0.088	0.164	0.096	0.196	0.146	*0.149*
Folia query	**0.590**	0.166	0.280	0.275	0.028	0.003	0.000	*0.192*

Table 4. Our approach based on correlative rules

	5-40	6-40	7-40	8-40	9-40	10-40	11-40	12-40	*Ave.*
Twinkle	0.133	0.132	0.138	0.136	0.134	0.139	0.141	**0.157**	*0.1391*
Lachrimae	**0.389**	0.339	0.341	0.351	0.350	0.357	0.352	0.314	*0.3498*
Folia	**0.502**	0.442	0.437	0.439	0.450	0.469	0.486	0.465	*0.4618*

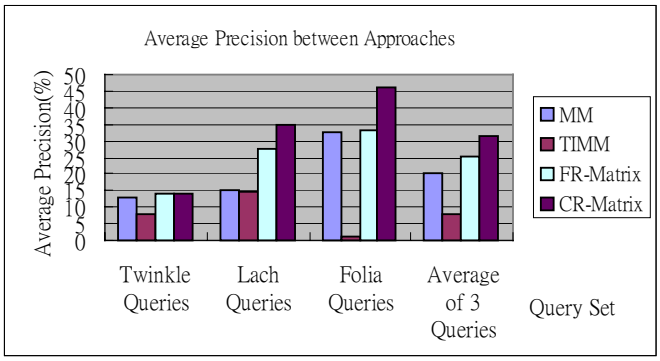

Figure 1. The average precision of each approach

Figure 1 shows the average precision achieved by each approach. MM and TIMM denote the two approaches based on the Markov-model and the time-invariant Markov-model, respectively. Moreover, FR-Matrix and CR-matrix denote the two approaches based on the frequent rule and the correlative rule, respectively. Among them, CR-Matrix has the best precision for all the query sets. Considering the average of all the query sets, CR-Matrix achieves 56% and 307% improvements on average

precision of MM and TIMM, respectively. Moreover, FR-Matrix also respectively achieves 23% and 221% improvements.

For each query set, Table 5 shows the average precision of FR-Matrix and CR-Matrix under different settings of the minsup. For Twinkle queries, FR-Matrix and CR-Matrix both have low precisions with different values of minsup. This is because both types of rules cannot capture enough information from the short sequences in this variation set.

Table 5. Average precision of FR-Matrix and CR-Matrix for three query sets

Query set	Method	Minsup(%)							
		5	6	7	8	9	10	11	12
Twinkle Precision(%)	CR-Matrix	13.34	13.25	13.82	13.66	13.44	13.95	14.10	15.73
	FR-Matrix	15.52	14.43	13.22	14.22	14.52	15.17	13.71	13.78
Lachrimae Precision(%)	CR-Matrix	38.98	33.93	34.19	35.18	35.07	35.79	35.26	31.44
	FR-Matrix	31.77	31.72	31.77	30.97	26.04	19.93	20.24	19.98
Folia Precision(%)	CR-Matrix	50.23	44.28	43.71	43.94	45.06	46.97	48.69	46.57
	FR-Matrix	42.18	38.88	36.90	36.58	32.83	29.24	25.54	22.98

For Lachrimae queries and Folia queries, we have observed three common phenomena as follows:

1. CR-Matrix always performs better than FR-Matrix. It means that the correlative rules well represent the sequences with strong correlation between their elements. Moreover, among the variations of a music piece, there indeed exists such kind of patterns.
2. When the minsup increases, both FR-matrix and CR-Matrix may degenerate. This is because long patterns tend to be filtered out and fewer rules can be derived. It shows that long patterns are important to the similarity search on music variations.
3. As the minsup reaches 9%, the trend of FR-Matrix has a drop greater than that of CR-Matrix. It verifies that the frequent pattern is more sensitive to the minsup than the correlative pattern. The reason is because longer patterns have smaller supports and are easier to be filtered out via the minsup. In a short summary, the correlative rule is more suitable to be applied to the music variations.

6 Conclusion and Future Work

In this paper, we propose a novel approach to discover two types of patterns and rules hidden in the individual sequences and use them to represent sequences. Moreover, we provide the similarity measure between two sequences based on each type of rules.

Two experiments are made for performance evaluation. First, we compare our approach with the Markov-model approach on similarity search for music variations. The results show that our approaches based on frequent rules and correlative rules respectively achieve 23% and 56% improvements on the average precision to that of the Markov-model approach. In addition, the approach based on correlative rules always performs better than the one based on frequent rules. The results verify our intuition that structural information can be useful to represent a sequence and the correlative pattern indeed reveals a significant aspect of structural information hidden in the sequences. We conclude the main advantages of the correlative pattern as follows:

1. Only the correlative 1-patterns must satisfy the minsup. Therefore, the pattern that appears only a few times but has strong correlations among its elements is allowed to be the correlative pattern. This correlative pattern does exist in some applications where 'hot' items are related to 'cold' items, e.g. the common behaviors in Web browsing.

2. The examination on the rule confidences acts like giving a virtual threshold on the supports of correlative patterns, where the threshold decreases as the length of the patterns increases. Therefore, the length has a little impact on the pattern discovery. Moreover, the long pattern, which is potentially useful and often covers a large portion of the sequence, is easy to be recognized as the correlative pattern.

In the second experiments, we evaluate the effectiveness of our approaches based on each type of rules across different parameter settings, including the music classes and the minsup.

Our future works are as follows. In the mining process of correlative patterns, we examine the rule confidences to filter out the patterns directly. However, some of them might be important information to the sequence. For example, assume B, C, and D often appear after A in a sequence. Our approach may not include this information because the confidences of the three rules (i.e., $A \rightarrow B$, $A \rightarrow C$, and $A \rightarrow D$) are averaged. Another interesting issue is to adapt the window size for constructing the segment database to different application needs. It will be useful to design a performance metric such the quality of resultant patterns can be estimated or the total costs for the mining task can be predicted.

7 Acknowledgements

This work was partially supported by the MOE Program for Promoting Academic Excellence of Universities under the grant number 89-E-FA04-1-4, and the NSC under the contract number 92-2213-E-007-094.

References

1. Agrawal R., Faloutsos C., Swami, A.: Efficient Similarity Search in Sequence Databases. Proceedings of International Conference on Foundations of Data Organization and Algorithm. (1993) 69-84.
2. Agrawal R., Srikant R.: Mining Sequential Patterns. Proceedings of IEEE Conference on Data Engineering. (1995) 3-14.
3. Chen H. C., Chen A. L. P.: A Music Recommendation System Based on Music Data Grouping and User Interests. Proceedings of ACM Conference on Information and Knowledge Management. (2001) 231-238.
4. Chai W., Vercoe B.: Folk Music Classification Using Hidden Markov Models. Proceedings of International Conference on Artificial Intelligence. (2001).
5. Guralnik V., Karypis G.: A Scalable Algorithm for Clustering Sequential Data. Proceedings of IEEE Conference on Data Mining. (2001) 179-186.
6. Gusfield D.: Algorithms on Strings, Trees, and Sequences. Cambridge University Press, (1997).
7. Hsu J. L., Liu C. C., Chen A. L. P.: Discovering Nontrivial Repeating Patterns in Music Data. IEEE Transactions on Multimedia, Vol. 3(3). (2001) 311-325.
8. Han J., Kamber M.: Data Mining: Concepts and Techniques. Morgan Kaufmann. (2001).
9. Huron D.: The Melodic Arch in Western Folksongs. Computing in Musicology, Vol. 10. (1995).
10. Kowalski G. J., Maybury M. T.: Information Storage and Retrieval Systems Theory and Implementation. Kluwer Publishers. (1997).
11. Lesh N., Zaki M. J., Ogihara M.: Mining Features for Sequence Classification. Proceedings of ACM Conference on Knowledge Discovery and Data Mining. (1999) 342-346.
12. Moshe S.: Dynamic Programming. Marcel Dekker. (1992.).
13. Manning C. D., Schütze H.: Foundations of Statistical Natural Language Processing. MIT Press. (2001).
14. Morzy T., Wojciechowski M., Zakrzewicz M.: Pattern-Oriented Hierarchical Clustering. Proceedings of Advances in Databases and Information Systems. (1999) 179-190.
15. Pickens J., Bello J. P., Monti G., Crawford T., Dovey M., Sandler M., Byrd D.: Polyphonic Score Retrieval Using Polyphonic Audio Queries: A Harmonic Modeling Approach. Proceedings of International Conference on Music Information Retrieval. (2002).
16. Pickens J., Crawford T.: Harmonic Models for Polyphonic Music Retrieval. Proceedings of ACM Conference on Information and Knowledge Management. (2002) 430-437.