Content-based Retrieval and Indexing Methods for Music Databases

Yuzuru Hiraga
University of Library and Information Science
1-2 Kasuga, Tsukuba, 305-8550 JAPAN
hiraga@ulis.ac.jp

Abstract

This paper presents a conceptual framework for matching, indexing, and retrieval of music databases. The framework, called SIR, emphasizes use of structural descriptions as search keys. This "structural indexing scheme" enables both elaborate matching and efficient retrieval. The key concepts of the framework are presented, and relation with existing work and implementation issues are discussed.

1 Introduction

With the advance and wide acceptance of global networks and multimedia systems, there has been increasing interest in music data retrieval in the recent few years (e.g. [2, 3, 4]). Access to music data by content (typically, melody lines) requires matching/search schemes that are far more elaborate than text-based access, such as retrieval of bibliographical information. This is because, for one thing, music data is inherently multi-dimensional, and for another, specification of retrieval queries tend to be inaccurate, fragmental, having deviation that does not conform with the internal representation of the database.

This paper presents a conceptual framework of matching, indexing and retrieval of music databases called SIR (Structural Index Retrieval). The basic idea is to extract as much structural information as possible, and utilize them as search keys. The formalism assumes a symbolic representation and is based on syntactic pattern matching. This is in contrast with, for example, statistic matching of numerical vectors. The discrepancy between the query and stored data is intended to be absorbed by exact matching at abstract and reduced description levels. This abstraction mechanism enables both elaborate matching and efficient retrieval. The framework is intended to be applicable not only to simple melody search, but for music theoretic purposes as well.

2 Existing Methods

Query to music databases may deviate from the intended target representation for various reasons (e.g. inaccurate input, deviation in the stored data, system errors as seen in conversion of audio input to symbolic note forms). Existing works deal with such deviations by allowing a degree of tolerance in the matching process. Their approach can be roughly divided into two types — note-wise matching and time-span matching.

Note-wise matching treats individual notes as units of match, and matches the note sequences [2, 4]. Tolerance of matching is realized by use of elaborate matching schemes such as DP matching, and use of multiple keys. Sequences of pitch, note length (=span), and/or their relative values (intervals, time ratios) are matched separately. A further elaboration is the use of coarse coding, in which attribute values are classified into broad categories such as up/down, long/short. While flexible matching is possible, the drawback is that matching is invoked for every search instance and efficient index-lookup is not possible.

Time-span matching quantizes the time axis into unit frames, and records the pitch within each frame in a vector form [3]. Time spans are normalized so that source and target vectors can be directly matched. Various types of deviation can be handled within a single framework, and use of standard indexing schemes is possible for efficient retrieval. However, certain kinds of deviation, particularly those along the time axis, are not so well handled.

Both methods use a uniform distance measure over their pattern space to evaluate the strength of match. In other words, various types of deviation are collectively treated and reduced to a single or small number of measures. Sequences are seen to be equally similar if the distance values are the same. The problem associated with this can be illustrated by the following simplified example.

Suppose the matcher allows deviation of one semitone, and is given the pitch sequence:

C – D – E – F – G
as query. This will match with sequences such as:

\[ \text{C - D - E - F}^\# - G \]
\[ \text{C - D - Eb - E - G} \]
\[ \text{C - D - E - F - F}^\# \]
\[ \text{C - D - E - E - G} \]

which have different musical connotations, inducing different judgment of similarity with the original. On the other hand, sequences like “C - C - E - F - G” are not matched. This may lead to retrieval of noise results that not only fails to fulfill the user's intentions, but are also difficult to comprehend why they were chosen.

A further, perhaps more fundamental problem is that they cannot handle broad, abstract matching such as variations to its theme (or vice versa). They are bound to the specificity of the query.

3 Structural Indexing

The above observation suggests that in order to improve the quality of matching, we should look more into musical contents and structure (including cause of deviations). Existing methods are limited in this aspect, appealing more to general purpose methods that are not domain specific.

SIR is based on the intuition that if a query is to have successful retrieval results at all, then it should retain some characteristic feature(s) of the intended target. If such features are explicitly extracted from both the query and database entries, they can be directly matched. This, in effect, shifts complexity of the matching process to diversity of multiple representations. Each of the representations can serve as search keys, leading to multiple index retrieval.

Structural feature extraction in SIR (based on previous work [1]) can be categorized into categorical abstraction, segmentation, and functional abstraction. Music data is assumed to be single melody lines of ordinary tonal music in symbolic form, such as score notation or MIDI files.

3.1 Categorical Abstraction

"Categorical abstraction" corresponds to coarse coding mentioned before. Pitch and (time-)span sequences are generated at various levels of abstraction (i.e. coarsity), using either numeric or symbolic values. Sequences of base forms (having offset values from a fixed origin) or relative forms (taking origin in preceding note, like intervals and span ratios) are used whenever applicable.

For example, the C-D-E-F-G sequence at the chromatic (=semitone) level is written as:

\[ \{0, 2, 4, 5, 7\} \] (base form) and
\[ \{*, 2, 2, 1, 2\} \] (relative (interval) form)

where "*" marks the position of the origin. The pitch domain has abstraction levels of: diatonic (key-specific), diatonic (key-free), chord, up to the most abstract level of direction (up/down/similar). For example, C-D-E-F-G becomes:

\[ \{*, 1, 1, 1, 1\} \] (diatonic level)
\[ \{*, p, 1, p, 1\} \] (chord level)

where "p" denotes non-categorical passing tones. The sequence of 1's in the former (as well as latter, ignoring the p's) indicate stepwise ascending motion.

In the time domain, spans are classified on a logarithmic scale. The base form imposes a (qualitative) meter hierarchy in which onset/offset of spans are classified as on or in-between beats of each meter level (e.g. quarter- or eighth-note level). Abstraction over the relative form is more straightforward, distinguishing only the magnitude of span ratios (i.e. log. values).

The extraction process uses simple fitting algorithms over local cues, working in coordination with other phases such as segmentation and key identification. Each unit process may return no results (if information is insufficient) or multiple, competing results (in case of ambiguity).

3.2 Segmentation

Note sequences are segmented into groups according to local cues, based on continuity of intervalic/temporal relationships. For example, sequences with stepwise motion or equal span are seen as continuous, while a longer note disrupts continuity so a boundary is expected at either its start or end (see [1]). A flat sequence with no segmentation is at the lowest level of abstraction, and those with more segmentation are at higher levels.

Segmentation serves two purposes. One is for use in functional abstraction (described below), and the other is as potential starting points for partial matching over segments (next section). Since there is ambiguity in boundary selection, competing indices can be generated for retrieval purposes.

3.3 Functional Abstraction

Functional abstraction is intended to provide a concise, structural description of a given sequence. It has the dual function of reduction and annotation. Reduction picks out structurally significant notes, while annotation describes motion of other notes governed by that structural note. A reduced sequence is a sequence of structural note pitches, with the span set to the total span of notes they govern.

For example, the sequence in Fig.1(a) can be described as having C and G as structural notes, with annotations “C ascends stepwise to G” (at
the diatonic level) and “G divides into two quarter-notes”. Fig.1(b) shows the reduced sequence C-G with half-notes and occurrence of annotations with solid lines. A reduced sequence is by itself an abstraction of the original, while annotations characterize the contour of the sequence.

Functional abstraction consists of two levels. Level 1 merges notes with the same pitch (and also, non-categorical tones like passing notes and chromatic inflections). Level 2 gives the more general abstraction described above, working on the segmented sequences. If necessary, these are recursively applied to reduced sequences for higher level abstraction. Structural notes are selected by the criteria of taking e.g. the longest, head, or tail note. Annotations are basically note sequences with simple patterns (e.g. ascending pitch, dotted rhythm).

Annotations can be parameterized to give more abstract representation. For example, the annotation for C in Fig.1(a) can be described as “ascending stepwise sequence from start to goal with equal span”, with actual values such as C for start and G for goal. The parameterized description can match sequences with arbitrary length or underlying pitch category.

4 Matching and Retrieval

These feature extraction methods are combined to generate representations of the source at various levels of abstraction or described aspect. Representations for the database entries are pre-processed and used as index of retrieval, while those for the query are used as search keys.

The details of the following (especially that of 4.2) are not yet worked out, so we present the key issues and intended plan.

4.1 Matching with Tolerance

Basically, the matching between key and index is required to be an exact-match. That is, if a query and target is to be matched, they must agree on some index (e.g. at a certain level of abstraction).

Inexact match is obtained when the deviation is encapsulated in the features dropped by the

abstraction. For example, an index at the level of “diatonic (key-free), non-segmented, same pitch merged” will ignore repetition of same pitch, chromatic inflections, and mode (major/minor). This by itself produces results similar (though not identical) to uniform distance matching in 2, but are more confined and explainable (in combination with analysis of difference at lower levels). Parameterized functional abstractions enable matching of sequences with different length.

Another case is partial matching, where description over some attributes and/or segments agree, but others don’t. Examples of partial attribute match are when the pitch sequence is matched but the rhythm is different, or when the exact pitch is altered but overall contour is retained (as in children’s songs). Partial segment match allows inducing overall match when the query matches a part of the target well. Segmented matching is useful not only for partial matching, but also to filter out fragmental portions of a query.

In the above sense, SIR allows tolerance of matching in a confined manner. This is intended to capture significant aspects of the query leading to musically sensible results, while retaining sufficient flexibility. The exact-match criteria enables use of standard indexing schemes for efficient retrieval.

4.2 Retrieval

Since retrieval is over multiple (possibly conflicting) indices, the general case is that each target will match with some indices while not with others. This leads to the problems of rank ordering of results and search method.

While there is partial ordering in both the indices themselves (a higher level index subsumes a lower level index) and the matched results (on basis of set inclusion of matched indices), general comparison of match over heterogeneous indices is more difficult. Some general criteria for comparison include: (a) prefer match of pitch over match of span, (b) inter-level matching (e.g. matching a reduced sequence to a ground sequence) is permissible but less preferred. In order to provide a serial rank ordering of the results, a scoring scheme based on index types would be necessary.

This is related to how the search is carried out. Trying out all indices at once would be inefficient. A more practical approach is to start with a default, limited key set and perform iterative search by including other keys. Inclusion of new keys has a dual function that more specific keys will restrict the retrieved set, and more general keys will enlarge it. Determining the ordering of how new keys are introduced effectively amounts to defining the scoring scheme mentioned above. The process can also be interactive, being controlled by the user.

ICMC Proceedings 1999
The user may also directly specify the level of operation for specialized purposes.

4.3 Index Generation

Indexing a database of full music piece entries requires segmentation of the piece into proper units. The bottom line is to generate indices starting at every note [2] or dispense with the indexing and perform a full-piece search. However, a reliable segmentation will help reduce both time/space costs and retrieval noise.

Automated segmentation (including ours) may not be necessarily reliable. But the problem can be concealed in that if the same segmentation method is used for both the query analysis and index generation, they are likely to coincide. However, it is also our belief that to improve the quality of a database (including indexing), careful tuning by hand would be necessary. In this sense, we are not directed to open search of resources such as those on the Web [4].

5 Discussion

Currently, we are at the stage of testing the index generation phase, and the entire retrieval system is yet to be implemented. So as of now, a concrete evaluation of SIR cannot be presented. The below gives some provisional discussion of our framework in relation to existing work.

5.1 Relation with Existing Work

SIR is closest in objective and formulation to the work by Lemström and Laine [2], and can be seen as its extension. They use symbolic representation of music, and make use of coarse coding (categorical abstraction) over pitch. Sonoda et al. [4] also incorporate coarse coding and DP-based coarse-to-fine matching, but their emphasis is more on the manipulation of audio level data. So their approach and objectives can be seen as complementary to ours. Nishihara et al. [3] developed a concise and efficient system based on time-span matching. SIR presents a near-equivalent alternative to time-span matching by use of functional abstraction.

The major difference with these (with the possible exception of [2]) is our emphasis on feature extraction preceding the retrieval. The cited works, on the other hand, place more substantial processing in the matching and retrieval phase.

5.2 Retrieval Quality

Retrieval failures can be categorized as either: (1) failure to include the intended target, or (2) inclusion of extraneous entries (noise). The existing work are more inclined to reducing type 1 failures at the possible cost of increasing noise results. Our attention, comparatively, is more directed to type 2 failures, in reducing noise and producing sensible and explainable results.

The competence of SIR with respect to type 1 failure is a matter of question. While various deviations can be absorbed through abstraction, it is weak against sporadic errors that occur outside of normal musical context. Ironically, uniform distance methods fare better with these types of error. One way to deal with the problem is to look for out-of-context notes in the feature extraction phase, and ignore them. This however, is not easy, and in a practical implementation, we need not be that stoic and make parallel use of other methods as well.

6 Conclusion

This paper presented the conceptual framework of SIR, which emphasizes the use of structural representations as retrieval keys. Explicit representations will help identifying characteristic features of the query (at abstract levels), alleviate the retrieval process, and make the results more understandable.

The actual validity of this claim must be tested on an actual implementation, which is being worked on. One of the crucial factors is the actual patterns and types of deviation that occur in user queries. This will require careful investigation through psychological and analytical studies, which is a topic of future work. The results of such studies may lead to revision of what types of features we should look for and represent.

Although we make no claim that our framework is universal, it may at least provide complementary views and approaches that would supplement shortcomings of other approaches, introducing novel applications such as use for research purposes.

References