THE MELODIC SIGNATURE INDEX FOR FAST CONTENT-BASED RETRIEVAL OF SYMBOLIC SCORES

Camelia Constantin
LIP6, Univ. Paris 6, Paris, France
camelia.constantin@lip6.fr

Cédric du Mouza
CEDRIC, CNAM, France
dumouza@cnam.fr

Zoé Faget
Armadillo & Univ. Paris-Dauphine, France
zoe@armadillo.fr

Philippe Rigaux
CEDRIC, CNAM, France
philippe.rigaux@cnam.fr

ABSTRACT

NEUMA is an on-line library that stores collections of symbolic scores and proposes a public interface to search for melodic pieces based on several kinds of patterns: pitches-based, with or without rhythms, transposed or not. In addition, searches can be either exact or approximate. We describe an index structure apt at supporting all these searches in a consistent setting. Its distinctive feature is an encoding of the various information that might be involved in the pattern-matching process with *algebraic signatures*. The properties of these signatures are suitable to represent in a compact and expressive way the sequences of complex features that constitute a melodic description.

1. INTRODUCTION

Context and motivation. NEUMA is a Digital Score Library devoted to the publication of digital music scores. Putting this material on-line offers an opportunity for web-based sharing of musical scores archives, including collaborative production, annotation, and large-scale corpus analysis. In the present paper, we focus on the functionalities that permit to undertake large-scale studies of melodic, harmonic or stylistic material. One of the musical investigations currently conducted by our fellow musicologists working with NEUMA considers a melodic *répertoire* in a given cultural area, and studies how this répertoire is exchanged and borrowed throughout various styles, periods and composers. Using efficient tools to retrieve and compare similar melodies leverages the scope of investigations that can be conducted for such a study. To this end, NEUMA provides a set of functions that support the analysis process. The *pattern-matching function* takes a pattern P and carries out a search over the score collections, looking for *all* the melodic fragments that "match" P. The function can be parameterized by combining one of the following options: *Exact search*, which can itself be refined as *Transposed/non transposed* and/or *With/without rhythm*, and *Approximate search*, which compares P to melodic fragments considered in their full dimensions (pitch, rhythm) and applies a similarity function. The user is free to choose an appropriate combination of these choices (called an *interpretation* in the following), and this yields a quite appreciated flexibility to the system. This flexibility has a cost, though, since the system must be ready to face several possible pattern interpretations.

Indexing the pattern-matching retrieval process. As our collections grow, the need for an indexing mechanism able to directly access the scores of interest for a given pattern became prominent. Building an index for each possible interpretation would have been cumbersome due to the major redundancy of information in the associated descriptors. We rather chose to design a specialized index, able to satisfy several interpretations. This design, and the experiments that validate the resulting structure, constitute the purpose of the present paper.

In short, the principles of our index, called *Melodic Signature Index* (MSI), can be summarized as follows: (i) its kernel structure is that of a traditional *hash file*, with an in-memory directory that refers to a list of on-disk *buckets*; (ii) each entry e in the directory corresponds to the hash value h_e of some fixed-size melodic fragments, called n-grams, present in at least one score of the collections; the associated bucket actually contains the list of *all* the n-gram occurrences that hash to h_e; (iii) the index implementation is consistently built over *algebraic signatures* computed from the melodic n-grams, and representing the various aspects that might be addressed by one of the possible pattern-matching interpretations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.
© 2011 International Society for Music Information Retrieval.

Whereas the first two aspects are drawn from the state-of-the-art in terms of large text-encoded indexing [14], the last one is inspired by recent work on signature-based text processing [7, 10], tailored to the specifics of symbolic music retrieval. The resulting structure enjoys several features that make it a suitable choice for large score libraries indexing, namely (i) *flexibility* – a single index supports several distinct pattern-matching operations, (ii) *compactness* – in spite of the rich information content it contains, the index space requirement is only a fragment of the overall collection storage, and (iii) *efficiency* – as shown by our analytic study and experiments, a few milliseconds suffice to retrieve the result, even for very large patterns searched for in very large collections.

Related work. Two main approaches for off-line indexing score collections have been investigated: tree-based [9, 13, 20] and inverted files [3,5,16]. [5,16] propose to index both the pitch interval and rhythm sequences in an inverted file. We adopt a similar approach, with a much richer encoding that allows to reach a constant search complexity and more flexibility in terms of search options.

The subjective nature of measuring music similarity lead to the introduction of several error measures. The δ and (δ, α) approximations [2] use exact matching algorithms for similarity search. Many algorithms for efficient computation of similarity matching through exhaustive search have been proposed [1, 4]. In general, indexing can be achieved with a high-dimensional structure whose performances are known to deteriorate as the dimension increases. In the specific context of the edit distance, several indexing methods have been suggested, an overview of which can be found in [15]. A classical technique is to introduce an measure approximating the edit distance but easier to index [12]. The idea of using n-gram for melody retrieval and measuring music similarity is not new in monophonic [17,19] as well as polyphonic pieces [6, 8], although they usually model only some of the music information. Our structure enjoys the nice feature of being able to index both exact search with many variants, and approximate search based on the edit distance. This makes it a structure of choice to solve the addressed problem of index pattern searches in large score databases.

The rest of the paper presents our structure (Section 2) and the pattern-matching algorithms (Section 3). Section 4 briefly reports the performance results obtained over a large collection of scores, and Section 5 concludes the paper.

2. THE MELODIC SIGNATURE INDEX

We outline in this section the index structure in NEUMA, with emphasis on algebraic information put in index records.

2.1 Index overview

NEUMA interprets scores content according to a "model" of symbolic music. The model of interest to this work relies on a synchronized time series approach that sees a score as a superposition of *voices*. Each voice is a sequence of elements in $\mathcal{E} \times \mathcal{D}$, where $\mathcal{E}$ is the domain of musical "events" (notes, chords, rest, etc.) and $\mathcal{D}$ the musical duration. A descriptor can be text-encoded in the form <$\mathtt{e}_1$-$\mathtt{d}_1$; $\mathtt{e}_2$-$\mathtt{d}_2$; ...; $\mathtt{e}_n$-$\mathtt{d}_n$> where each $\mathtt{e}_i$ encodes an event and each $\mathtt{d}_i$ its duration. In the following, we shall blur the distinction between a descriptor and its textual encoding. Given a descriptor d, we denote as $\epsilon(d)$ the sequence of events (without durations) and as $\rho(d)$ the sequence of durations (without events) of d.

Example 1 *Voice v, in score 354, encodes a melody beginning with a G3 (half), followed by an A3 (half), a B3 (flat, quarter), etc. Its descriptor d_v is: (22-2;24-2;25-4;24-4;22-4;21-4;22-4;...) Moreover, $\epsilon(d_v)$ = (22, 24, 25, 24, 22, 21, 22, ...) and $\rho(d_v)$=(2, 2, 4, 4, 4, 4, 4, ...).*

In the example above, note heights are encoded with chromatic notation (number of semi-tones from the lowest possible sound). Rest, chords, and silence are encoded with other, non ambiguous, symbols: we do not elaborate further $\mathcal{E}$ which provides a compact representation of melodic sequences.

Given a pattern P, a search retrieves the scores such that for *at least* a voice v, and *at least* an offset (position) o in v, P matches the fragment $v[o]v[o+1]\ldots$. The semantics of a matching attempt depends on the interpretation of P, chosen by the user at query time. We explain the process with an example: let P be the pattern described by `37-4;35-4;34-2`. Then, under the *exact search, transposed, without rhythm* interpretation, P matches the voice v of Example 1 at offset 3 (offsets start at 0). If we take the rhythm into account, this is no longer true. Using a non-transposed interpretation also leads to a failure, with or without rhythm. Finally, an approximate search likely detects a high similarity between P and v at position 3.

2.2 Algebraic signatures

We interpret our melodic events in $\mathcal{E}$ as elements of a Galois field $GF(2^f)$ of size 2^f. The elements of GF are bit strings of length f. Since $|\mathcal{E}| \leq 255$, we let $f = 8$ in the following. A Galois field is a finite set that supports addition and multiplication. These operations are associative, commutative and distributive, have neutral elements 0 and 1, and there exist additive and multiplicative inverses. A *primitive* element α of GF is such that its powers enumerate all the non-zero elements of the Galois field. Let $D = e_0 e_1 \cdots e_{M-1}$ be a descriptor encoding a sequence of M events interpreted as GF elements. We define an *AS signature* as follows.

Definition 1 *The* AS α-signature *of a descriptor D is defined by*

$$AS_\alpha(D) = e_0 + e_1 \cdot \alpha + e_2 \cdot \alpha^2 \ldots + e_{M-1} \cdot \alpha^{M-1} \quad (1)$$

If we consider m primitive elements $\alpha_1, \alpha_2, \ldots, \alpha_m$, the m-symbols signature $NAS_m(D)$ is obtained by concatenating the set of $AS_{\alpha_i}(D), 1 \leq i \leq m$, seen as bit strings. This allows to obtain a signature of size m.

Given a descriptor D, we are interested in *partial* algebraic signatures calculated from substrings of D.

Definition 2 *Let $l \in [0, M-1]$ be any* offset *in D. The* Cumulative Algebraic Signature *(CAS) at l, $CAS(D, l)$, is the algebraic signature of the prefix of D ending at e_l, i.e., $CAS(D, l) = AS(e_0 \ldots e_l)$.*

The *Partial Algebraic Signature (PAS) from l' to l* is the value $PAS(D, l', l) = AS(e_{l'} e_{l'+1} \cdots e_l)$, with $0 \leq l' \leq l$, We most often use the PAS of sub-sequences of length n, i.e., of n-grams.

Definition 3 *The n-gram Algebraic Signature (NAS) of D at l is $NAS(D, l) = PAS(D, l-n+1, l)$, for $l \geq n-1$.*

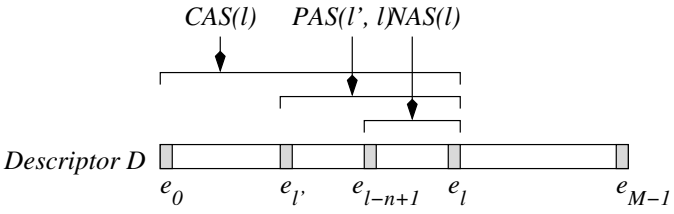

Figure 1. $CAS(l)$, $PAS(l', l)$ and $NAS(l)$ in descriptor D

We may drop D whenever it is implicit for brevity's sake. Figure 1 shows the respective parts of the record that define the CAS, PAS and NAS at offset l. The following simple properties of algebraic signatures are useful for what follows. Properties 2 and 3 let us incrementally calculate next CAS and NAS while indexing the score, or preprocessing the pattern, instead of recomputing the signature entirely. This speeds up the process considerably.

$$CAS(l) = CAS(l-1) + e_l \cdot \alpha^l \quad (2)$$

$$NAS(l) = \frac{NAS(l-1) - e_{l-n}}{\alpha} + e_l \cdot \alpha^{n-1} \quad (3)$$

Property 4 finally is fundamental for the match attempt calculus. For $0 \leq l' < l$:

$$CAS(l) = CAS(l') + \alpha^{l'+1} PAS(l'+1, l) \quad (4)$$

We refer the reader to [11] for more details about definitions and properties of algebraic signatures. The above are sufficient to describe the MS-index features.

2.3 The Melodic Signature index

The Melodic Signature Index (MS-Index) is a classical hash file, denoted HD[$0..L-1$], with directory length $L = 2^v$ being a power of 2 (Figure 2). Elements of HD refer to buckets or *lines* of variable length.

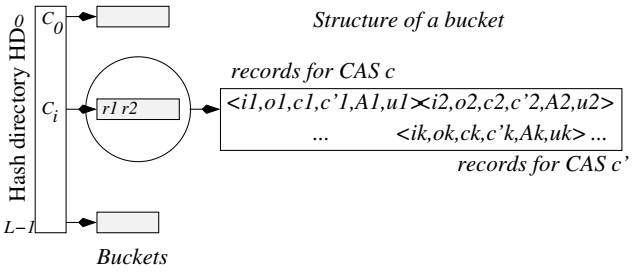

Figure 2. Structure of the MS-Index

.

Each bucket stores a list of *hash records* (records in short), each indexing some fixed-size fragment of a voice descriptor, called n-gram. Fragment *(24-4;22-4;21-4)* is for instance a 3-gram extracted from the descriptor of Example 1. The actual value of n is a parameter of the MS-Index, to be discussed next. To build the index, we process all n-grams in the score library. From each n-gram G of the form $\mathtt{e}_1\mathtt{-d}_1\mathtt{;\ldots;e}_n\mathtt{-d}_n$ we derive a number of algebraic signatures that determine the index organization and content.

We first use signatures to calculate the index i of the line that refers to G. Let τ be the transform that extracts from G a (n-1)-gram with the sequence of pitch intervals. We calculate i by hashing on the intervals signature. Let $s = NAS_m(\epsilon(G))$ be the m-symbol signature of G for some m (see below), interpreted as a large, unsigned integer and compute index i as:

$$i = h_L(S) = S \mod L$$

Since $L = 2^v$, this amounts to extracting the last v bits of S. m should be such that $m \leq n$ and $m \geq \lceil v/f \rceil$.

Example 2 *Let G be the 4-gram (24-4;22-4;21-4;22-4). Then $\epsilon(G)$=(24, 22, 21, 22) and $\tau(\epsilon(G))$=(-2, -1, 1) (e.g., the pitch interval encoding). Assume $m = 3$. We select three independent primitive elements α_1, α_2, and α_3 in the Galois Field. The index of G in the hash file is:*

$$AS_{\alpha_1}(\tau).AS_{\alpha_2}(\tau).AS_{\alpha_3}(\tau) \mod L$$

where . represents bit string concatenation.

The properties of AS signatures ensure a balanced distribution of the hash values in the range $[0..L-1]$. Next, we insert in HD[i] a *record* describing G, defined as follows:

Definition 4 *Let G be an n-gram at offset o in a descriptor D. The* record *indexing G, denoted $R(G)$, is a 6-uplet $(id(D), o, c_\epsilon, c_\rho, AS_\rho, \perp)$ where*

1. *c_ϵ is $CAS(\epsilon(D), o)$, i.e., the* event *CAS of G at o;*
2. *c_ρ is $CAS(\rho(D), o)$, i.e., the* rhythm *CAS of G at o;*
3. *AS_ρ is $NAS_m(\rho(D), o)$, i.e., its rhythm signature;*
4. *$\perp$ is the minimal pitch index in G, representing (along with the previous signatures) its absolute height.*

The hash record of an n-gram contains all the information necessary to evaluate matching attempts at run time, by combining the signatures with the Galois Field operators to evaluate the required pattern interpretation.

Example 3 *Consider again the 4-gram G of Example 2, assuming it is found at offset 3. Then c_ϵ and c_ρ are obtained from the cumulative values at offset $o-1$, thanks to Property 2; A_ρ is the NAS signature of $\rho(G)$=(4, 4, 4, 4); $\perp$ is 21, the minimal pitch of the n-gram.*

Construction time complexity. The MS-index is built in linear time in the size of the score library. Note in particular that the cumulative signature at offset o can be derived from the cumulative at offset $o-1$.
Space complexity. The size of the directory, HD, is negligible. Given a descriptor D, a record occupies $3+2+1+1+1+1=9$ bytes, and the index size is therefore $|L| \times \tau_D \times 9$, where τ_D denotes the ratio of descriptor's size with respect to a full score size. Standard indexed file compression techniques (e.g., variable bytes compression) further reduce the space requirements. As shown by our experiments, τ_D is typically of the order of $1‰$ and, in spite of its rich content, our index occupies a small fraction of the whole library space.

3. SCORE RETRIEVAL

Due to space limitation, we give in this section an informal presentation of the algorithms.

3.1 Exact search, basic algorithm

We explain (Figure 3) an exact search, transposed and without rhythm (that is, we consider as a match any sequence of pitch intervals similar to that of P). First, we preprocess P for three signatures: (i) of the initial n-gram S_1, (ii) of the final n-gram S_2 and (iii) of the suffix S_p of P after S_1. Hashing on S_1 locates the bucket with every record r_1 hashing to the signature of S_1. Likewise, hashing on S_2 locates the bucket with every r_2 hashing to the signature of S_2. We only consider pairs of records that are in the same voice and at the right distance among them (looking at offsets). We

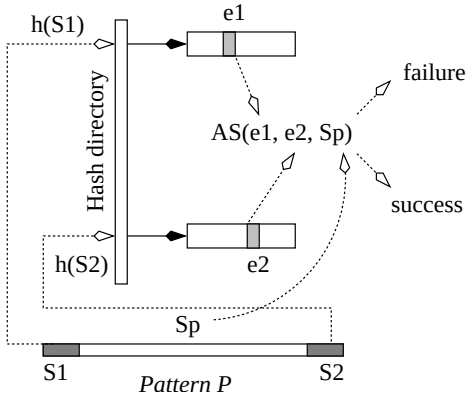

Figure 3. A matching attempt with MS-Index

thus locate any descriptor D matching P on its initial and terminal n-gram, at least by signature. An algebraic calculation $AS(r_1, r_2, S_p)$, based on the cumulative signatures, determines whether S_p may match the suffix of D as well.
Search complexity. By limiting disk accesses to the two buckets associated to the first and last n-grams of the P, MS-Index search runs independently from P's size. The cost of the search procedure outlined above is reduced to that of reading two buckets. The hash directory is cached in RAM. With an appropriate dynamic hashing mechanism that evenly distributes the records in the structure and scales gracefully, the bucket size is expected to remain uniform enough to let the MS-Index run in constant time.

3.2 Exact search, other interpretations

Other interpretations than the basic one are obtained with straightforward extensions to the above algorithm, namely 1) non-transposed search, without rhythm, is obtained by comparing the minimal pitch index of P's initial n-gram and the value $\perp$ of r_1 ; 2) searching with rhythm implies a calculus similar to that on intervals, using $r_1.c_\rho$, $r_2.c_\rho$ and A_ρ as input ; 3) any combination of these criteria is possible to achieve the required interpretation.

The cost analysis remains similar, since the signatures comparison is negligible regarding that of buckets access.

3.3 Approximate search

Our index supports the similarity measure using n-grams introduced by Ukkonen [18]. The more n-grams the two strings have in common, the higher the similarity. The *$n-$gram profile* is a vector G_P such that $G_P[S]$ is the number of occurrences of the n-gram S in P. The "distance" between two strings P and Q is then:

$$A_n(P,Q) = \Sigma_{v \in \Sigma^n} |G_P[v] - G_Q[v]|,$$

where Σ^n is the set of all possible $n-$grams.

collection [1]	# files	files size	# desc.	desc. size
bach	280	27.1 MB	1,243	539 KB
gut	137	197.2 MB	352	2,413 KB
hausmusik	452	140.9 MB	1,218	1,944 KB
hymns	1,752	84.6 MB	3,885	1,954 KB
musicxml	405	38.9 MB	1,738	713 KB
wikifonia	3,583	302.7 MB	3,570	2,787 KB
wima	961	427.3 MB	3,110	4,624 KB
misc	94	8.9 MB	101	89 KB
all	7,664	1,227.6 MB	15,517	15,063 KB

Table 1. MusicXML collections used in NEUMA

The approximate search of a pattern P in a symbolic score proceeds as follows. Given a descriptor $D = e_1 \dots e_N$, a pattern $P = p_1 \dots p_m$ we pre-process P to get all the n-grams S_1, S_2, $\dots S_q$ occurring in P. We access the MS index and retrieve, for each $S_i, i \leq q$, the list of the records featured in the document with the same signature than $h(S_i)$. We then sort-merge all lists into one list, ordered with respect to each descriptor. We take the first list of offsets and apply a moving window of size $L = 2m - n + 1$ in which we solve the approximate search problem. Indeed we can show that a window of size L has $2m - 2n + 2$ n-grams, from which at most $m - n + 1$ belong to P and at least $m - n + 1$ do not belong to P. For windows of size greater than $2m - n + 1$, n-grams not belonging to P will always outnumber those who do.

We compute the A_n distance between the pattern and all subsequences starting on the left edge of the window, and keep track of the ending position for the best one inside the window. We repeat this process for all offsets of the list by sliding the window along the list. We return all triplets $(i_{start}, i_{end}, d_i)$ which comply to the maximum error tolerance.

4. EXPERIMENTS

We built a library of MusicXML scores collected from several public on-line collections, reported in Table 1. There exists an important discrepancy in the size of the descriptors. The average descriptor size is 967 bytes, and it ranges from 444B on average in bach to 7,020B in gutenberg (noted gut). The ratio $(descriptorsize/documentsize)$ varies from 9 ‰ in wikifonia to 23 ‰ in hymns.

Table 2 reports the building time and the size of the MS-Index for different datasets. For bach, gut and wima, we choose 4-grams. The building time does not linearly increase with the descriptors size. For instance gut, whose descriptors size is half that of wima, requires a third of the building time of wima, while all (4-gram), with a descriptor size 3 times larger than wima, needs 7.5 times more time. This results from both the handling of hash collisions and variable-bytes compression (not detailed here).

As expected, the size of the index linearly depends on the descriptors size. Finally using larger n-grams has a minor impact on the index size, but an important one on the building time: *e.g* 7-gram index requires 25% more space than 3-gram index thanks to lower compression rate, but a building time 7 times higher, due to less collisions to handle and less compression to perform.

collection	building time	size
bach	0.7 s	1.0 MB
gut	3.3 s	5.1 MB
wima	11.4 s	9.5 MB
all (3-gram)	206.6 s	28.5 MB
all (4-gram)	82.6 s	29.7 MB
all (5-gram)	47.0 s	31.3 MB
all (6-gram)	35.9 s	33.2 MB
all (7-gram)	33.2 s	35.1 MB

Table 2. Building time for different collections

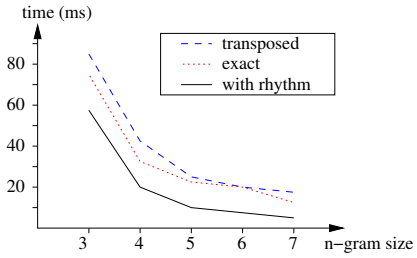

Figure 4. Impact of the n-gram size on matching time

Figure 4 shows that the longer the n-grams, the faster the search, whatever the interpretation [2]. Longer n-grams means less collisions, and thus smaller buckets. Differences between exact, transposed or without rhythm search performances are mostly due to the selectivity of the search criteria. Unlike transposed search (TR), we eliminate for an exact search (EX) records in the first bucket (retrieved using the NAS of the first n-gram) by checking the first note on the n-gram. This decreases the comparisons to perform. Search transposed with rhythm and search exact with rhythm exhibit similar performances, and run faster than TR or EX since we filter records using an additional signature.

Finally we study the search time in Table 3 and compare performances with those of an exhaustive scan. MS-Index overperforms for all datasets the exhaustive search (the ratio ranging from 800% to 10,000%). The search time with MS-Index does not depend on the descriptors size: wima is twice larger than gut but searches are performed 4 times

[1] bach: www.jsbchorales.net, hausmusik: www.hausmusik.ch, gut: www.gutenberg.org/wiki/Gutenberg:The_Sheet_Music_Project, hymns: www.hymnsandcarolsofchristmas.com, musicxml: www.musicxml.org, wikifonia: www.wikifonia.org, wima: www.icking-music-archive.org

[2] We limit the presentation of the results to exact search.

coll.		TR	TR+RY	EX	EX+RY
gut	MS-index	38.1	27.8	36.9	32.4
	SC	323.1	212.2	293.4	302.1
	speed-up	8.5	7.6	7.9	9.3
wima	MS-index	10.4	7.5	9.7	7.5
	SC	637.4	432.1	581.1	595.2
	speed-up	61.3	57.6	59.9	79.3
all	MS-index	41.6	20.7	33.3	24.5
	SC	2,514.2	1,490.2	2,305.3	2,030.1
	speed-up	60.4	72.0	69.2	82.9

Table 3. Impact of the dataset size on search time (ms)

faster, and the same ratio holds when comparing to `all` whereas its size is 3 times larger. Our index performances are more sensitive to the data distribution since skewness leads to large bucket, thus a larger number of tests. Searches with rhythm are faster since they filter out records in the first bucket (resp. n-grams) for the MS-Index (resp. exhaustive scan), skipping useless comparisons. The speed-up is lower for `gut` than for other collections. The rationale is that `gut` presents a few, large files (137) with more records for each document in a bucket. Since the id of the document is also a filtering condition (we try to match an entry of the first bucket with one of the second bucket from the same document), more matching attempts are carried out.

5. CONCLUSION

We described in this paper a practical approach to the problem of indexing pattern-based searches in a large score library. Our solution supports exact and approximate searches, and permits to refine exact searches by taking account of the many components that constitute a melodic descriptor. Our experiments show that a few milliseconds suffice to obtain the result in all cases even for significantly large datasets.

A nice feature of our index is that it also acts as an initial filter in a two-steps similarity search method that performs a final check on the candidates against the full descriptor. This leaves the opportunity to adapt the edit distance to the specifics of music score similarity search. We are currently investigating the relevance of such adaptations with our users.

6. REFERENCES

[1] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, M. Mohamed, and M.-F. Sagot. A Pattern Extraction Algorithm for Abstract Melodic Representations that Allow Partial Overlapping of Intervallic Categories. In *ISMIR*, pages 167–174, 2005.

[2] D. Cantone, S. Cristofaro, and S. Faro. Solving the (δ, α)-Approximate Matching Problem Under Transposition Invariance in Musical Sequences. In *ISMIR*, pages 460–463, 2005.

[3] C.-W. Chang and H. C. Jiau. An Efficient Numeric Indexing Technique for Music Retrieval System. In *ICME*, 2006.

[4] R. Clifford and C. Iliopoulos. Approximate string matching for music analysis. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 8, 2004.

[5] S. Doraisamy and S. M. Rüger. A Polyphonic Music Retrieval System Using N-Grams. In *ISMIR*, 2004.

[6] S Doraisamy and S M Rüger. An approach towards a polyphonic music retrieval system. In *ISMIR*, pages 187–93, 2001.

[7] C. du Mouza, W. Litwin, P. Rigaux, and T. J. E. Schwarz. AS-index: a Structure for String Search Using N-grams and Algebraic Signatures. In *CIKM*, pages 295–304, 2009.

[8] R. Hillewaere, B. Manderick, and D. Conklin. String quartet classification with monophonic models. In *ISMIR*, pages 537–542, 2010.

[9] I. Karydis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos. Audio Indexing for Efficient Music Information Retrieval. In *MMM*, pages 22–29, 2005.

[10] W. Litwin, R. Mokadem, P. Rigaux, and Th. Schwarz. Fast nGram Based String Search over Data Encoded Using Algebraic Signatures. In *VLDB*, 2007.

[11] W. Litwin and T. Schwarz. Algebraic Signatures for Scalable Distributed Data Structures. In *ICDE*, pages 412–423, 2004.

[12] N.-H. Liu, Yi-Hung Wu, and A. L. P. Chen. An Efficient Approach to Extracting Approximate Repeating Patterns in Music Databases. In *DASFAA*, pages 240–252, 2005.

[13] Y.-L. Lo and S.-J. Chen. The Numeric Indexing For Music Data. In *ICDCSW*, pages 258–266, 2002.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] G. Navarro, R. Baeza-yates, E. Sutinen, and J. Tarhio. Indexing Methods for Approximate String Matching. *IEEE Data Engineering Bulletin*, 24:2001, 2000.

[16] G. Neve and N. Orio. Indexing and Retrieval of Music Documents through Pattern Analysis and Data Fusion Techniques. In *ISMIR*, 2004.

[17] I. Suyoto and R. Uitdenbogerd. Mirex 2005 symbolic melodic similarity: Simple efficient n-gram indexing for effective melody retrieval. *Music Information Retrieval Evaluation eXchange*, 2005.

[18] E. Ukkonen. Approximate String Matching with q-grams and Maximal Matches. *Theoretical Computer Science*, 92:191–211, 1992.

[19] Julián Urbano, Juan Lloréns, Jorge Morato, and Sonia Sánchez-Cuadrado. Mirex 2010 symbolic melodic similarity: Local alignment with geometric representations. *Music Information Retrieval Evaluation eXchange*, 2010.

[20] J.-Y. Won, J.-H. Lee, K.-I. Ku, J. Park, and Y.-S. Kim. A Content-Based Music Retrieval System Using Representative Melody Index from Music Databases. In *CMMR*, pages 280–294, 2004.