SEMEX – An Efficient Music Retrieval Prototype

Kjell Lemström and Sami Perttu
Department of Computer Science
University of Helsinki
{klemstro,perttu}@cs.helsinki.fi

The origins of music information retrieval (MIR) are in manual collections of *incipits*, short melodic fragments obtained from the beginning of pieces of music. The collections were manually compiled and usually covered a narrow field of music. Recently, computerized content-based MIR systems have appeared. They apply standard methods from general string matching.

The applied techniques are based on the assumption that music is representable symbolically. This is the case for most music; consider, e.g., the score notation of western music or MIDI. These are not, however, ideal representations; the score notation requires a conversion into a computer-readable form, and MIDI contains a lot of irrelevant information. Instead, music is represented by sequences of symbols (or *strings*), where the symbols are taken from an *alphabet* corresponding to some particular attribute of music, such as the duration or the pitch of a note.

The choice for the representation in current MIR systems is a string of pitches (or pitch intervals). This is because of a pitch string of a melody is easy to remember, and a pitch string has a good discrimination ability. Although such a representation loses some characteristics of music, there are many reasons to do so. First, it facilitates the modeling of music (in particular polyphonic music) and the modeling of distortions in the pattern. Moreover, when a piece of music has been performed in another musical genre, very often only the rhythmic pattern has been subject to a rearrangement. Therefore, in a MIR application it can be advantageous to rely the query only on the pitch information.

Features of SEMEX

Our MIR prototype, SEMEX (Search Engine for Melodic Excerpts), performs queries on pitch sequences. It is capable of locating transposition invariant matches of monophonic query patterns in both monophonic and polyphonic music databases. The current version allows approximate matching in the monophonic case, but not in the polyphonic one.

In order to take into account *transposition invariance*, SEMEX calculates intervals when accessed. Such an arrangement is useful because the *interval representation* has certain drawbacks: In the case of a distortion where an extra interval has been inserted (or a proper interval has been deleted), the resulting sequence does not anymore represent the original piece of music (because the distortion forces a modulation to take a place). SEMEX offers various reductions for the accessed intervals. This is a useful feature when an extra amount of proximity is needed, as e.g. in the case of *query-by-humming*. First, the queries can be based on the *musical contour*, which gives only the direction of the intervals. This method, however, requires long query patterns to reach good discrimination ability. A better discrimination, still maintaining tolerance to errors, is obtained by using the *QPI classification* [Lemström and Laine, 98].

To perform the queries SEMEX applies *bit-parallelism*, a particularly efficient algorithmic technique. Bit-parallel algorithms do not require excessive amount of the main memory (which

happens with the indexing techniques at times). Still they can achieve a good performance; in the case of short enough patterns, they achieve a linear time running complexity. These properties are of crucial importance since music databases can be very large. For monophonic database SEMEX employs Myers' approximate bit-parallel algorithm [Myers, 98]. The fast scanning phase is succeeded by verification where a separate metrics is used for ranking matches. For searching exact occurrences of a "distributed" melody within a polyphonic database, SEMEX applies the MonoPoly filtering algorithm [Lemström and Tarhio, 2000], which comprises a bit-parallel marking phase and a somewhat slower checking phase.

About the Performance

Let us compare the estimated performances of the bit-parallel approach and an indexing approach using suffix trees. The estimations are based on two databases described in [Bainbridge et.al, 1999]. When using suffix structures, exact pattern matching can be performed in a time depending only on the length of the pattern. The main drawback is the large amount of space that such a structure needs. In practice, even the most space-efficient implementation needs a space (of the main memory) that is at least 10 times the length of the database while, as already mentioned, the bit-parallel techniques can usually be executed without any, or with only a small amount of extra space. However, their running times are dependent on the length of the database.

Let us first consider the moderate sized database comprising nearly 10,000 folksongs, having an average length of just under 60 notes. Assuming that one note fits in one computer byte, concatenating the whole database in a single sequence would result in a sequence of around 600,000 notes needing a space of 0.6 MB. In this case, the indexing structure would take at least 6 MB of the main memory, while the efficient online algorithm scans through such a database within 0.05 seconds if run with present efficient Pentium III computers. The superiority of the bit parallel implementation becomes obvious in the second case. Now, the database contains the MIDI files (easily) available on the Internet (the exact duplicates have been removed). The database, containing 99,000 files having a total of 528 million notes, would require a suffix tree taking around 5 GB of the main memory! The bit parallel implementation, however, would scan through it in approximately 50 seconds. Although the performance of the online method is not particularly fast anymore, the approach could still be used. This is not the case with the excessive amount of space needed by the indexing approach.

References and Suggested Readings

Bainbridge, David; Nevill-Manning, Graig; Witten, Ian; Smith, Lloyd and McNab, Rodger. 1999. *Towards a Digital Library of Popular Music*. In: Proc. ACM Conference on Digital Libraries, pp. 161-169.

Crawford, Tim; Iliopoulos, Costas and Raman, Rajeev. 1998. *String Matching Techniques for Musical Similarity and Melodic Recognition*. Computing in Musicology, 11, pp. 71-100.

Downie, J. Stephen. 1999. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic n-grams as Text*. PhD thesis, Univ of Western Ontario, Faculty of Inf. and Media Studies.

Lemström, Kjell and Laine, Pauli. 1998. *Musical Information Retrieval Using Musical Parameters*. In: Proc. 1998 International Computer Music Conference (ICMC'98), pp. 341-348.

Lemström, Kjell and Tarhio, Jorma. 2000. *Searching Monophonic Patterns within Polyphonic Sources.* In: Proc. Content-Based Multimedia Information Access (RIAO'2000), pp. 1261-1279 (vol 2).

Lemström, Kjell. 2000. *String Matching Techniques for Music Retrieval*. PhD thesis, Univ of Helsinki , Dept of Computer Science, (to appear).

Myers, Gene. 1998. *A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming.* In: Proc. Combinatorial Pattern Matching, pp. 1-13.

Perttu, Sami. 2000. *Combinatorial Pattern Matching in Musical Sequences*. MSc thesis, Univ of Helsinki, Dept of Computer Science (http://www.cs.helsinki.fi/u/perttu/mscthesis.ps).