Running head: SEARCHING DIGITAL MUSIC LIBRARIES

Searching digital music libraries

David Bainbridge, Michael Dewsnip, and Ian H. Witten

Department of Computer Science

University of Waikato

Hamilton

New Zealand

Abstract

There has been a recent explosion of interest in digital music libraries. In particular, interactive melody retrieval is a striking example of a search paradigm that differs radically from the standard full-text search. Many different techniques have been proposed for melody matching, but the area lacks standard databases that allow them to be compared on common grounds—and copyright issues have stymied attempts to develop such a corpus. This paper focuses on methods for evaluating different symbolic music matching strategies, and describes a series of experiments that compare and contrast results obtained using three dominant paradigms. Combining two of these paradigms yields a hybrid approach which is shown to have the best overall combination of efficiency and effectiveness.

Searching digital music libraries

Corresponding author: David Bainbridge

Phone: +64 7 838 4407

Fax: +64 7 858 5095

E-mail: d.bainbridge@cs.waikato.ac.nz

A shorter version of this paper was presented at *International Conference on Asian Digital Libraries*, Singapore, December 2002.

Keywords: music libraries, melody retrieval, music matching, evaluation

Introduction

There has been a recent explosion of interest in digital music libraries—indeed, Apple's iPod has been called the world's first consumer-oriented digital library. In all human societies, music is an expression of popular culture. Different generations identify strongly with different musical styles. People's taste in music reflects their personality. Teenagers, in particular, feel that their musical preferences are strongly bound up with who they are. Many researchers seek to capitalize on this natural interest by building digital music libraries (Bird & Downie, 2000; Downie & Bainbridge, 2001).

Digital music libraries are an attractive area of study because they present interesting and challenging technical problems, solutions to which are likely to be highly valued by enthusiastic end-users. This paper addresses the problem of searching a music library for a known melody. In other words, given a fragment of an unknown melody, typically played or sung by a user, return a list of possible matches to a large digital library collection. This operation is not supported by the information structures provided by traditional libraries, except insofar as knowledgeable music librarians are able to provide human assistance. For scholarly work on melody, there is a book that provides a paper index of themes (Parsons, 1975), but its scope is restricted to the older classical repertoire and it does not provide flexible searching options. And yet the problem of melody retrieval is of great interest to a wide range of potential users—so much so that there are popular radio programs that feature human abilities to "guess that tune".

A practical scheme for searching digital music libraries requires robust implementations of several supporting components. First, it is necessary to assemble a large database of music in searchable form—which implies some kind of symbolic representation. Normally this is accomplished by manually entering a large number of melodies on a keyboard, a highly labor-intensive process. The alternative is to automatically infer notated information from an analog form of the music, such as a

recording of a performance, or a paper score. The latter possibility, OMR for "optical music recognition," is a well-advanced technology (Bainbridge & Bell, 2001) but is not addressed in this paper. Second, the audio query, generated by the user singing, whistling, or humming it—or playing it on a keyboard—must first be transcribed into the same symbolic representation. This is a far easier proposition than inferring a musical score from a recording of a performance, because the input is monophonic—only one voice is present. However, again we do not address the problem in this paper: the transformation is accomplished by standard signal-processing techniques of pitch detection (Gold & Rabiner, 1969), followed by quantization of the pitch track in both frequency (placing the notes on a standard musical scale) and time (placing the notes within a standard rhythmical framework). Third, the music searching operation must take place within a context that allows the user to examine search results and request that generalized "documents" be presented. Such documents might include stored musical performances, performances synthesized on the fly from some symbolic representation, facsimile images of the score, scores created on demand from a symbolic representation by musical typesetting techniques, and so on. Suitable general contexts for browsing and document presentation exist, for example (Bainbridge, Nevill-Manning, Witten, Smith, & McNab, 1999); again, they are not addressed by this paper.

We focus here on the central problem of music matching. We assume that the material to be searched is stored in symbolic form in terms of the notated music. We assume that the audio query has been transcribed into the same symbolic form. We assume that a suitable infrastructure is in place for examining and presenting results.

Three basically different approaches to symbolic music matching have been proposed: dynamic programming (Mongeau & Sankoff, 1990), state matching (Wu & Manber, 1992), and n-gram-based methods that employ standard information retrieval techniques (Downie, 1999). All have been used to implement practical melody retrieval

systems. Dynamic programming techniques work by calculating, in an efficient manner, the "edit distance" between the query and each melody in the database. The lower the distance, the better the match. Rudimentary edit operations include adding a note, deleting a note and substituting one note for another, along with more subtle changes such as consolidating a series of notes at the same pitch into one note of the composite duration. State based matching also works by adding, deleting and substituting notes to provide an approximate match, but its implementation takes a different form. It uses a matrix of bits that records the state of partial matching so far, and achieves efficiency by encoding the matrix as an array of machine words. Given this data-structure, only shifts and bitwise Boolean operators are needed to implement the matching progress. Unlike dynamic programming, state-based matching does not keep track of which edits were made, and its running time is proportional to the number of errors that are allowed. N-gram-based methods work by mapping both queries and melodies to textual words (n-letters long) and then using full-text retrieval to locate documents in the database that contain the "words" included in a given query (Witten, Moffat, & Bell, 1999).

The aim of this paper is to provide a comparative evaluation of these three methods for searching digital music libraries. We begin by introducing the workbench we have developed to perform these experiments, and then describe the experiments themselves and the results obtained. Notable amongst the experiments is a hybrid technique that achieves the computational expediency of the n-gram approach while maintaining the more favorable recall and precision of state-based matching. The paper concludes with a summary of our findings.

A workbench for symbolic music information retrieval

To support practical research in this area we have developed a workbench for symbolic music information retrieval. Figure 1 gives on overview of the system, which fits

into a larger digital library software architecture, Greenstone (Witten, R., Boddie, & Bainbridge, 2000; Witten, Bainbridge, & Boddie, 2001) Work is divided into two phases: the assimilation phase and the runtime phase. The former is responsible for collecting files together and creating from them the necessary indexes and/or databases. The latter, guided by user input, supports experimentation and evaluates performance measures. While assimilation is typically performed once for a given collection, the runtime phase is executed many times to gather results from different experiments.

Internally the workbench represents music files using the XML version of Guido (Hoos, Renz, & Gorg, 2001) which is general, expressive enough for our needs, and straightforward to parse. During the assimilation phase the workbench processes these files, building algorithm-specific data structures that allow efficient searching.

In the runtime phase, users issue commands to interact with the workbench. They can provide sample inputs and match them against the database using different matching algorithms, examining and comparing the results. Each matching method has optional arguments that modify its behavior. For example, one can seek matches only at the start of melodies, rather than at any position within them. Instead of using absolute pitch values, one can use pitch intervals (differences in pitch value) or pitch contour (whether the pitch value rises, falls, or stays the same). The workbench implements many other matching options. The outcome of a search is held as a result set from which statistics are extracted, graphs plotted, tables generated, and so on.

Interactive use has its limitations, particularly when setting up and running large experiments. Consequently there is a facility for users to develop a "script" that defines a particular series of experiments. This script is then run by the workbench in batch mode, and the results are recorded in files for later examination by the user.

A third mode of operation is to allow a different process, rather than an online user or a pre-prepared script, to access the facilities of the workbench. The workbench can be

accessed through a web-based user interface, using the CGI mechanism, to perform music-content retrieval and format the data returned in a suitable format. This allows its use directly by digital library software, for example, Greenstone. The advantage is that exactly the same implementation and options are used for live retrievals as have been evaluated in interactive and off-line experiments.

The music information retrieval workbench is released under the GNU public license. It provides a uniform basis for evaluating melody matching algorithms. More importantly, other research groups will be able to add their retrieval algorithms to it, allowing a comprehensive comparison of their strengths and weaknesses against the prior state of the art without the need to continually re-implement earlier methods. An alternative strategy, which has been adopted in other communities (for example text compression (Arnold & Bell, 1997) and machine learning (Blake, Keogh, & Merz, 1998)), is to develop a standard corpus of material against which different algorithms are evaluated, and publish the results of these evaluations. Indeed a public domain workbench is complimentary to such an approach, however in the context of music, on-going efforts to form such a tested have been stymied by issues of copyright. Recently, a small but richly diverse database of music has been created for this purpose (Goto, Hashiguchi, Nishimura, & Oka, 2002), avoiding copyright issues by producing and recording all original music.

Experimentation

The purpose of our experiments is to shed light on how well commonly-used music information retrieval algorithms perform under a wide variety of conditions. This provides the basic information needed to design and configure a digital music library. Such information is necessary to make a sensible choice of algorithms used to support query by music content in practice; it is also necessary to fine-tune particular parameter settings. Conditions differ from one digital library to the next, depending on factors such as the

user community being served, the computer infrastructure that underpins the service, and the type and size of the collection. Our aim is to provide design data for digital music libraries. If, in addition, a library uses our workbench to respond to queries, the implementation is guaranteed to be the same as was used to produce the design data.

Datasets

For evaluation, we need to use standard corpora of melodies. Recall the legal difficulties, mentioned above, of creating and distributing corpora. Due to the absence of a globally used corpus at the time of experimentation, we have used two datasets that were available to us internally. The first dataset combines songs from the Essen and Digital Tradition collections (Bainbridge et al., 1999) to form a dataset of nearly 10,000 short folksongs (about 530,000 notes in total). The second dataset consists of 1000 tunes extracted from MIDI files of various genres (pop, rock, classical and alternative) obtained from the internet. These tunes are much longer than those in the folk collection, so despite having only one-tenth the number of songs, the MIDI collection is much larger (about 2,000,000 notes in total).

Summary of experiments

Our experiments are based on earlier work by McNab (1996), Downie (1999), and Rand *et al.* (2001). Where possible, we follow the same experimental procedures, expanding and extending the details appropriately.

Our first set of experiments studies the recall and precision rates of the various algorithms. This helps establish some limits on how well they perform in terms of the mix of relevant and non-relevant documents that are returned, but does little to convey how much effort a user must exert to locate a sought after tune. This is the purpose of our second set of experiments which, through artificially constructed examples, considers the number of songs returned to a given query. The usefulness of a matching algorithm is also

dependent on how long it takes to execute. This forms the basis for our third set of experiments. Additional information about our experimental procedures can be found in (Dewsnip, 2002).

Recall and precision

Recall and precision are the dominant measures of an information retrieval system's ability to present the relevant documents in response to a query. Recall is the proportion of the relevant documents returned, and precision is the proportion of returned documents that are relevant.

Calculating recall and precision requires knowing which documents in a collection are "relevant" to a query, as determined by human evaluation. Human relevance judgements can be distributed with a collection and associated queries, and used to measure the recall and precision performance of different retrieval techniques. These resources are not yet available for the music retrieval community, however, so the approach taken by Downie (1999) is used instead. This approach defines the relevant documents to be those in the collection that include the query in its entirety.[1] In the workbench, these are determined using either of the matching algorithms (state-based or dynamic programming) in exact matching mode. The experiments presented in this section were run on the folk collection, using the same 150 queries for each. Downie (1999) tested 4-grams, 5-grams and 6-grams; here 3-grams also are considered for their potential robustness to errors. The dynamic programming algorithm is not tested in the recall and precision experiments, since in exact mode it returns the same results as the state-based matching algorithm, but takes a lot longer to do so (see Section).

The first experiment measures the optimum performance of the algorithms. This occurs when the algorithms are in exact mode (state-based matching allows no errors, n-grams perform boolean AND queries) and the queries are free of errors. Each algorithm

is tested using just rhythm, just pitch, and both pitch and rhythm. Additionally, where pitch is used, the three pitch representations are considered: absolute pitch (abs), pitch interval (int) and pitch contour (con). The precision values for the five algorithms are shown in Table 1; recall is always perfect (1.0) in this situation because there is no chance a relevant document can be missed.

Table 1 shows some interesting results. First, retrieval performance using only rhythm is very poor: at best, only one document in ten returned is relevant. This is due to the small amount of variation in tune rhythm (there are only a few common note durations, so the entropy (Shannon, 1948a, 1948b) of the rhythm information is low and it does not discriminate well). Retrieval performance using only the pitch information is also poor. For absolute pitch, the best achieved is about six relevant documents in ten returned, pitch interval achieves four in ten, and pitch contour does not even manage one in a hundred. Pitch contour does poorly for the same reasons as using only rhythm does: it has only three different symbols so there is very little variation to distinguish the documents.

Using both pitch and rhythm, performance is much better. When absolute pitch or pitch interval are used, well over 90% of the documents returned are relevant. Pitch contour performs poorer, but still manages around 80% precision for all algorithms except 3-grams. It is clear from these results that using both aspects of music information is necessary for acceptable performance, and using only one aspect is not considered further in this section.

Comparing the algorithms, the state-based matching technique performs best, since it is simply pattern matching in exact mode. The n-grams algorithms perform slightly worse because the documents they class as matches do not necessarily contain the whole query—the n-grams are not required to occur consecutively and in the same order. The number of these extra, irrelevant matches is represented by the difference in precision

between each n-grams algorithm and the state-based matching algorithm. Except where grams are short (the 3-grams algorithm), only a small fraction of the documents returned are irrelevant—this supports Downie's findings (1999).

User entry errors are much more common when searching a music digital library than when searching text. If queries are hummed or sung, many users do not possess the motor training required to do this accurately (McNab, Smith, Witten, Henderson, & Cunningham, 1996). Also, the transcription algorithm identifying the notes entered can be a source of error. Regardless of the query entry method, users may remember the tune incorrectly (forgetting notes, for example). With a text retrieval system, the majority of users can accurately enter the query they have in mind, but with a music retrieval system it is likely only a minority can do so. Therefore, lenient algorithms are essential for music retrieval.

The next experiment investigated the effects of user errors on retrieval performance. This experiment is the same as the previous one, except a randomly chosen error (entering the wrong pitch, entering the wrong duration, or dropping a note) is added into each of the queries. As expected, recall and precision plummet when errors are present: instead of retrieving *every* relevant document, only between 11% and 34% are found. Worse still, there are many more irrelevant documents returned: little more than one in ten matches are relevant to the intended query. Clearly, exact algorithms are unsuitable in an environment where errors are probable.

The leniency of the approximate algorithms promises better performance when errors are present. For the experiments testing approximate matching, the state-based matching algorithm is configured to allow one error when matching. The n-grams algorithms perform boolean OR queries, which are more lenient because they do not require every query n-gram to appear in a matching document. The recall and precision of the algorithms in this situation are shown in Figure 2. When errors are present, the

approximate matching algorithms perform much better than the exact algorithms: most configurations retrieve the majority of the relevant documents, and the best configurations achieve precision rates over 30%. This is reasonable performance, considering that the queries are not perfect (they include an error). Also, for many users the documents classed as "irrelevant" in this discussion may be of interest (if tune variants are being searched for, for example).

The performance of the algorithms could still do with improvement. One method of improving precision and recall is to increase the query length—longer queries mean fewer, more relevant documents are returned. To confirm this, the previous experiment was repeated using ten (rather than eight) note queries. In general, recall and precision performance improved (by up to 35%), and even longer queries would increase performance further. There is a limit on the number of query terms a user can be expected to enter, however, and entering more terms may mean more errors are introduced.

Number of songs returned

The effort required to use an information retrieval system is an important factor in user satisfaction and productivity. This section describes experiments investigating the trade-off in user effort between entering more query notes and searching through larger sets of matches.

These experiments measure the number of matches returned in relation to query length. Effectively, they show an algorithm's discriminative power: the number of query notes it needs to pick out the true matches. The results from these experiments can be used to predict the query length (on average) required to return a particular number of matches. For example, a user desiring fewer than ten matches can estimate the number of query notes necessary to achieve this.

Results from these experiments also can show the limits on retrieval performance.

Using exact matching and all the information available from a melody (absolute pitch and rhythm) shows the best that can be achieved using melody retrieval. For example, it shows that entering short queries will return many matches in a large collection, regardless of the algorithm used.

The number of matches returned from queries depends on many factors: the query length, the information used (pitch, rhythm, or both), the pitch mode, whether rests are used or ignored, whether exact or approximate matching is performed, and the contents of the collection. It also depends on properties intrinsic to a given retrieval algorithm. For example, the n-grams algorithm is less discriminating because the n-grams are not required to occur consecutively and in the same order, and therefore matching documents may not contain the exact query pattern.

The experiments in this section are configured in the following way. Except where noted, queries can match anywhere, rests are ignored, and the state-based matching algorithm is used in exact mode. Query lengths range between 2 notes and 20 notes: using one note is not applicable to pitch interval and pitch contour because they are based on differences in pitch. The experiments are run on the folk collection, using 300 randomly chosen files for each experiment.

The first experiment tested the effects of using only the pitch or only the rhythm information of tunes, and the results reinforce points from the previous section. Using only rhythm provides poor performance: queries 19 notes long (on average) are required to obtain ten matches. Using only pitch provides much better performance: about six notes are required for ten matches. Using both pitch and rhythm, however, only requires four notes. For most query lengths, using both pitch and rhythm returns about one-tenth as many matches as using just pitch, and about one-thirtieth as many as using just rhythm. It is clear that using both pitch and rhythm has beneficial effects on retrieval performance; using both does not usually require much extra work by the retrieval algorithm.

The next experiment explored the effects of the three pitch representations used in the previous section: absolute pitch, pitch interval, and pitch contour. Rhythm is not used in this experiment, in order to clearly show the effects of the pitch representations. The results of this experiment support those obtained with the recall and precision experiments. Absolute pitch is (as expected) the most discriminating representation. Pitch interval is less discriminating, requiring an extra query note for a similar number of matches; pitch contour is worse, requiring another three or four notes again.

Up to this point, rests have been ignored. The next experiment justifies this, showing they have negligible effect on retrieval performance. The experiment compared the number of songs returned when rests are used, and when they are ignored. Using rests returns about 10% fewer matches than ignoring them—clearly, rests do not have much more information content than notes, and are therefore not particularly useful. This result supports the findings of other projects: "rests appeared to be unhelpful in matching" (Uitdenbogerd & Zobel, 1999, page 65). Also, it is often unclear how to handle rests when matching, and they usually require special treatment. It is therefore concluded that rests require more effort than they are worth, and they are not considered further in the experiments described here.

The beginnings of songs are often designed to be catchy, and are therefore remembered well. Research has indicated that many people would form queries from the beginning of songs (McNab et al., 1996). Given this, it seems sensible to have a special option for matching queries at tune beginnings, and this is easily provided by the state-based matching and dynamic programming algorithms. From another experiment (Dewsnip, 2002), it was found that matching only at the start returns ten times fewer matches than when matching anywhere—an effective technique for reducing the number of matches for a user to search through.

The last two experiments in this section compare the performance of the matching

algorithms to the n-grams algorithms, and show the effects of different gram sizes. The first experiment compares the state-based matching algorithm to the n-grams algorithm (using 3-grams and 6-grams—the extremes of the gram sizes investigated) when exact matching is performed. The results from this experiment are expected to show that a longer gram size is more discriminating, but that the n-grams techniques never quite meet the performance of the matching algorithms. Figure 3(a) shows that these expectations are met. The 3-grams algorithm returns some extra, irrelevant matches to queries, but surprisingly few (supporting Downie's findings). The 6-grams is indistinguishable from the state-based matching algorithm, but actually returns extra documents very occasionally. The 4-grams and 5-grams (not shown) also return very few extra documents.

The recall and precision experiments in the previous section showed the necessity of approximate matching. The next experiment investigates the increase in documents returned when approximate matching is performed. This experiment tests the state-based matching algorithm; an equivalent graph could be computed for the dynamic programming algorithm, but they are not directly comparable due to their different leniency measures. The n-grams algorithms (3-grams and 6-grams again) perform boolean OR queries. Figure 3(b) shows the results from this experiment, which look very unusual at first glance. The state-based matching algorithm performs as expected, but the n-grams algorithms seem to perform very strangely: longer queries mean *more* matches? This behavior is due to the boolean OR queries, which match any document containing at least one of the query n-grams. More query terms therefore mean *more* matching documents, rather than less. The n-gramming process used explains why the number of songs returned drops sharply for very short queries, and then rises after a specific query length. When the query length is less than the gram size, the query cannot be broken up into n-grams. Therefore, a single gram of the query length must be searched for in the index, and every document containing it is returned. Once the query length exceeds the gram size,

however, more than one n-gram can be formed, and a boolean OR query can be performed. Documents containing *any* of the n-grams are returned, and the number of matches rises.

Comparing the performance of the state-based matching algorithm between Figures 3(a) and 3(b) shows that allowing an error when matching requires an extra note or two to get a similar number of matches. This supports the results from the previous section that showed that recall and precision decrease when approximate matching is used. When searching using approximate techniques, users should be encouraged to enter slightly longer queries (if they can), especially with large collections.

Algorithm efficiency

Fast algorithms are vital for searching large music collections. The experiments in this section measure the time algorithms take to perform queries. The experiments were run on a 1000MHz computer with 512MB of memory. The time taken to perform queries was measured using the built-in Java timer, which measures real time rather than process time—the computer was therefore dedicated to running only these experiments. Additionally, the results for each experiment were averaged over 300 repetitions to smooth out any small variations that occur.

Two experiments were run to determine the effects of using approximate matching on the time taken by the algorithms. The time taken by the dynamic programming algorithm is not affected by the type of matching (exact or approximate) performed. The state-based matching algorithm, however, takes time proportional to the number of errors allowed. When matching in exact mode, the state-based matching algorithm takes approximately 240 ms to search the folk collection (for all query lengths). When allowing one error, it takes 270 ms, and allowing more errors would increase the time taken proportionally.

The equivalent of exact and approximate matching for the n-grams algorithm is

boolean AND and boolean OR queries. An experiment was run to compare the time taken by these types of queries on the folk collection. For large gram sizes (five and above), AND and OR queries were almost indistinguishable. For small gram sizes, however, OR queries take up to eight times longer (depending on the query length).

The following experiments measure the effects on query time of query length and collection size, using the folk collection. The time taken to query the (larger) MIDI collection is investigated in Section .

The time taken by an algorithm to search a collection is often affected by the length of the query. Results from the previous sections have shown that longer queries are more effective, so two experiments were run to estimate search times for queries of increasing length. Both were run on the folk collection, using approximate matching and queries between 2 notes and 20 notes long. The first experiment investigated the performance gain when matching only at the start of documents—the size of the search space is effectively reduced because far fewer melody events need to be compared. The state-based matching algorithm takes 270 ms to search for queries anywhere in a document, and between 185 and 210 ms (depending on query length) to match at the start of documents only. The dynamic programming algorithm achieves a similar (relative) improvement, since it also takes time directly proportional to the collection size. The n-grams algorithm (as implemented) does not have the native ability to match queries only at the start of documents.

A second experiment compared the performance of the three algorithms. From the discussion of the algorithms at the start of the paper, the time taken with respect to the query length is theoretically linear with the dynamic programming algorithm, constant with the state-based matching algorithm, and sub-linear with the n-grams algorithms. The results from this experiment, shown in Figure 4, clearly support these facts.

The dynamic programming algorithm has the worst performance for all query

lengths. For queries of reasonable length, eight notes long for example, about three seconds are required to perform the query—likely to be unsatisfactory for many users. Worse still, this performance is for a modest music database of less than 10,000 tunes. As shown in the next section, the time taken by the dynamic programming algorithm also increases linearly with respect to the database size.

The state-based matching algorithm performs well, requiring about one-quarter of a second to search the folk collection. It is important to remember that this performance can only be maintained for queries shorter than the machine word size (32 on the machine used), however. To support longer queries, more complex operations would be needed to calculate the match state, and more memory would be needed to store it. However, results from the previous section indicate that it is unlikely that queries this long would ever be necessary.

The n-grams algorithms (3-grams and 6-grams were tested) are the clear winners. Except for very short queries, the time taken by the 6-grams algorithm to search the collection of nearly 10,000 tunes is less than one-hundredth of a second. This is about twenty times faster than the state-based matching algorithm, and hundreds of times faster than the dynamic programming algorithm. The graph seems to show that the time taken is constant with respect to the query length, but it actually increases very slightly as queries get longer (the graph's resolution is insufficient to show this). The 3-grams algorithm is slower, but is still over five times faster than the state-based matching algorithm.

The n-grams algorithm takes much longer (relatively) to process very short queries (the rise at the 2 note point of Figure 4). This is due to the support for queries shorter than the gram size, as explained earlier.

Algorithms whose performance scales well are vital for the large music collections of the future. An experiment was run to compare the search performance of the three

algorithms as collection size increases. This experiment uses eight note queries, and tests ten subcollections of the folk collection, of stepped sizes. The number of melody events ranged from 0 to the total number in the folk collection (approximately 530000). The time taken by the three algorithms to search these collections are shown in Figure 5.

The graph shows that the dynamic programming algorithm is linear with respect to the collection size. The time taken is increasing quite sharply however, so this algorithm soon becomes impractical for large collections. The state-based matching algorithm is also linear, but it has a much lower constant of proportionality. The n-grams algorithm is sub-linear, although it is hard to determine this from the graph.

A hybrid algorithm

The previous section has found the matching algorithms (state-based and dynamic) to be more *effective*: fewer query notes are required to return reasonable sets of matches, and recall and precision are usually better. The n-grams algorithms, however, are more *efficient*: queries are processed much faster. An ideal algorithm would be a combination of both these approaches, potentially achieving a good balance between effectiveness and efficiency. This section applies knowledge from the previous evaluation to the design of a "hybrid" approach, and evaluates it. Hybrid algorithms are sequential applications of the fundamental algorithms in the workbench. The first algorithm in the sequence must search the entire collection, but subsequent algorithms only search the documents the previous algorithm identified to be matches. In this way, each algorithm in the sequence refines the set of matches further.

The n-grams algorithm does not have the ability to search an arbitrary subset of a collection, so it can only be used as the first algorithm in the sequence. This is ideal: it is extremely fast and has the best recall for small gram sizes. The second algorithm used is therefore one of the matching algorithms. Our experiments in Section showed the

dynamic programming algorithm is a little more effective than the state-based matching algorithm, but a lot less efficient; consequently, we chose the state-based matching algorithm to be the second (and last) algorithm of the hybrid. The two algorithms in the hybrid use both the pitch and rhythm aspects of the tunes, and pitch intervals. Absolute pitch requires unreasonable accuracy from users, and the precision achieved using pitch contours is generally low. Lastly, a gram size of 3 is used for the n-grams algorithm—Figure 2 shows that this configuration has perfect recall on the collection tested, even with errors present. Its poor precision is not an issue for the hybrid approach, because it is followed by the state-based matching algorithm (which achieves much better precision).

The remainder of this section describes a series of experiments run to evaluate the performance of this hybrid algorithm. The experiments are based on those in the previous sections, but here only approximate matching is considered, since it is so important for music retrieval. Therefore, the 3-grams component performs boolean OR queries, and the state-based matching component allows one error. The experiments match queries anywhere in the documents, and compare the hybrid algorithm to the state-based matching algorithm alone and the 3-grams algorithm alone. The majority of the experiments in this section are run on the MIDI collection, since its larger size shows the performance trends better.

The first two experiments show the recall and precision performance of the hybrid algorithm. The first is a repetition of the experiment summarized in Figure 2: searching the folk collection for eight note queries containing one random error. The results from this experiment are shown in Figure 6. The results for the state-based matching and 3-grams algorithms are the same as those from Figure 2—they are shown here for comparison. This experiment also shows the performance of the hybrid algorithm using absolute pitch and pitch contour.

Figure 6 shows that the hybrid algorithm achieves recall and precision performance equal to the state-based matching algorithm by itself (which achieved the best performance in the original experiment). The reason for this is that the 3-grams algorithm has perfect recall for the folk collection, and therefore removes no relevant documents from consideration. The state-based matching component of the hybrid receives a smaller set of matches to search, but one that still contains all the relevant documents. Similar results were obtained for the MIDI collection.

Recall and precision are not sufficient to show the benefits of the hybrid approach. After all, using the state-based matching algorithm alone gives the same performance without the complexity of the hybrid approach. The key to the hybrid is the 3-grams algorithm—it removes many of the irrelevant documents, but leaves all the relevant ones. The number of irrelevant documents removed by the 3-grams algorithm is investigated by the next experiment.

This experiment is an application of the 'number of songs returned' evaluation technique to the MIDI collection. It is equivalent to the one shown in Figure 3(b), except it tests the hybrid algorithm; Figure 7 summarizes the results from this experiment.

The number of songs returned from a hybrid approach is always less than the number of songs that each of its components would return—it is actually the intersection between the sets of documents that each part would return by itself. Figure 7 shows that the n-grams algorithm (even using 3-grams) removes between 500 and 1000 tunes (usually about 800) from consideration. Effectively, the state-based matching algorithm only has to search 5% to 50% of the collection (usually about 20%), but all of the relevant documents are still present. The benefits of the hybrid approach stem from this.

The fact that the state-based matching algorithm is only required to search a small subset of the collection means that the total search time is much reduced. The time taken by the state-based matching algorithm is directly proportional to the collection size, so

the total time taken by the hybrid approach is a linear combination of the time taken by the separate algorithms alone. If $time_{3grams}$ is the time taken by the 3-grams algorithm to search the whole collection, and $time_{state-based}$ is the same for the state-based matching algorithm, then:

$$time_{hybrid} = time_{3grams} + p\, time_{state-based} \quad (1)$$

where p is the fraction of the collection left by the 3-grams algorithm.

This formula is supported by an experiment measuring query time, similar to the one shown in Figure 4. The time taken to query the MIDI collection with the three algorithms was measured as query length increases; results are shown in Figure 8.

The graph shows some interesting results. First, the state-based matching algorithm is slightly faster when the query length is small. This is due to the fact that very short queries match most documents exactly, and therefore the state-based matching algorithm does not have to calculate the approximate match state. Second, even on the larger collection the n-grams algorithm is extremely fast, taking about one-hundredth of a second.

The performance of the hybrid algorithm seems unusual, until Figure 7 and Equation 1 are considered. For queries between two and four notes long, the 3-grams algorithm performs exact (boolean) queries, and the number of matches found drops sharply. The p value of Equation 1 gets smaller, and the total time taken is reduced. After four notes, the 3-grams algorithm performs approximate (OR) queries, and increasing numbers of matches are found—p increases, and the total query time gets longer.

The last experiment shows the performance of the hybrid algorithm in relation to collection size. This experiment is the same as Figure 5, except the MIDI collection is used. Figure 9 shows the results from this experiment. The state-based matching algorithm achieves reasonable performance, searching the entire MIDI collection in about

425 ms. The hybrid algorithm does a great deal better, however, searching the collection in just over 60 ms—a factor of seven faster. The 3-grams algorithm does about eight times faster again (requiring a mere 8 ms to search nearly 2.5 million melody events), but its precision is much lower. Assuming the trend continues as it is shown in the graph, it is expected that the collection could grow by a factor of 16 (to approximately 40 million melody events) before the hybrid approach would require one second to search it (assuming the indexes fit into memory).

To summarize the important points from this section, a hybrid algorithm with an excellent balance between effectiveness and efficiency has been designed and tested. The algorithm uses pitch intervals and rhythm, and consists of the n-grams algorithm using $n = 3$, followed by the state-based matching algorithm. The algorithm was shown to equal the best recall and precision achieved when errors are present, to return reasonable sets of matches, and to perform queries very efficiently. Searching the MIDI collection with eight note queries takes about 60 ms, so it is likely the algorithm could search much larger collections in a reasonable amount of time. A fact not mentioned is that the hybrid algorithm also can perform matching at the start of documents, since the state-based matching algorithm supports this. Previous experiments showed that matching at the start of documents is an effective method of reducing the number of matches returned, and the time taken.

Even when the disadvantages of hybrid approaches are considered (they are more complex, and require more disk space and memory), they are excellent choices for achieving good efficiency and effectiveness.

Conclusion

We conclude this paper by relating the outcomes of our experimentation to forming a digital music library.

First, effective algorithms should use both the pitch and rhythm information of the melodies. Using both aspects has a beneficial effect on the number of songs returned, and improves recall and precision—users are likely to get fewer, more relevant matches to queries. Using both aspects does not usually require much extra work to implement, for example extending the state-based matching and n-grams algorithms to use rhythm require trivial changes (Dewsnip, 2002). Second, pitch interval was found to be a good representation to use, although it is more susceptible to some entry errors. Absolute pitch is not in general practical because it is key-dependent, and pitch contour requires too many query notes to be useful. Third, rests were found to be unhelpful in matching. Fourth, matching only at the start of documents is an effective method for reducing the number of query results and the time taken to search the collection. Melody retrieval systems may wish to allow the user to specify that their query is from the beginning of a song—or at least starts within the first few notes—and take advantage of these benefits.

Approximate matching is a necessity for melody retrieval. There are many opportunities for errors to be introduced into queries: poorly remembered tunes, pitch and duration errors (if queries are hummed or sung), and errors introduced by the transcription process (again, if queries are hummed or sung). These errors have dire consequences on the performance of the exact algorithms, and this makes exact algorithms unsuitable for melody retrieval. Also, approximate algorithms allow users to find tunes that are *similar* to a query. To achieve the best performance with approximate algorithms, however, users should be encouraged to enter an extra note or two.

Recommendations also can be made considering the three algorithms evaluated.

The state-based matching algorithm has a good balance between efficiency and effectiveness. Recall and precision performance is satisfactory in approximate mode, even when errors are present. Efficiency is reasonable: the MIDI collection can be searched in under half a second. The time taken by the algorithm is also independent of query length

(provided it does not exceed the machine word size), so users are not penalized for entering longer queries. The state-based matching algorithm is a suitable choice for searching moderate-sized collections of music.

The dynamic programming algorithm in comparison is slightly more effective in some cases and a little more flexible in its measure of approximation. Indeed a side effect of the algorithm is that it can detail the edits that are necessary to transform string A into string B—a factor that should not be overlooked when considering usability issues in a digital music library. This is something the state-based matching algorithm cannot do without effectively recomputing the match with additional information stored. The main detraction of the algorithm is its relatively high computational cost, taking on average 8 times longer to perform a match than its state-based counterpart—a problem that grows more acute as the collection size increases. One way to counter this trend is to base matching around identified themes in the songs rather than the entire piece, however this would require further experimentation to study the impact this has on recall and precision rates.

The n-grams algorithm is very efficient, but not as effective as the other two algorithms. Where the gram size is small, recall is excellent (and often perfect—even when queries contain errors), but precision is generally poor. Increasing the gram size can alter this balance: recall is reduced, but precision improves. The time taken by the n-grams algorithm to search the two collections was under one-hundredth of a second for boolean AND queries, and slightly longer for boolean OR queries. Unless queries are guaranteed to be free of errors, the poor precision of the n-grams algorithm is likely to make it unsuitable for use in a melody retrieval system, however.

Finally, the hybrid algorithm tested (3-grams followed by stated-based matching) has an excellent combination of efficiency and effectiveness. Using this algorithm, users can expect 80% of the relevant documents to be returned, and 60% of the documents returned

to be relevant (for eight note queries containing one error). Furthermore, they can expect queries to be processed very quickly (well under one-tenth of a second, on the machine used). The only disadvantages of the hybrid algorithm are related to its complexity and space usage, which are likely to be of little concern to the end users of the system. The hybrid approach has the best overall combination of efficiency and effectiveness, and it is therefore recommended for digital music libraries and other melody retrieval systems.

References

Arnold, R., & Bell, T. (1997). A corpus for the evaluation of lossless compression algorithms. In *Proceedings of the ieee data compression conference* (p. 201-210). Snowbird, Utah.

Bainbridge, D., & Bell, T. (2001). The challenge of optical music recognition. *Computers and the Humanities*, *35*(2), 95-121.

Bainbridge, D., Nevill-Manning, C., Witten, I., Smith, L., & McNab, R. (1999). Towards a digital library of popular music. In *The 4th ACM conference on digital libraries* (p. 161-169). Berkeley, California.

Bird, D., & Downie, J. (Eds.). (2000). *Proceedings of the 1st. int. symposium on music information retrieval: Ismir 2000.* Plymouth, Massachusetts. (Available through *www.music-ir.org*)

Blake, C., Keogh, E., & Merz, C. (1998). *UCI repository of machine learning databases.* http://www.ics.uci.edu/˜mlearn/mlrepository.html, University of California, Department of Information and Computer Science, Irvine, CA, Irvine, CA.

Dewsnip, M. J. (2002). *Evaluating melody retrieval algorithms.* Unpublished master's thesis, University of Waikato, Hamilton, New Zealand.

Downie, J. (1999). *Evaluating a simple approach to musical information retrieval: Conceiving melodic n-grams as text.* PhD. thesis, University of Western Ontario, Canada.

Downie, J., & Bainbridge, D. (Eds.). (2001). *Proc. of the 2nd int. symposium on music information retrieval.* Indiana University, Bloomington, IN. (Available through *www.music-ir.org*)

Gold, B., & Rabiner, L. (1969). Parallel processing techniques for estimating pitch periods of speech in the time domain. *J. Acoust. Soc. Am.*, *46*(2), 442-448.

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2002, October). RWC music database: Popular, classical, and jazz music databases. In M. Fingerhut (Ed.), *3rd international conference on music information retrieval (ismir 2002)* (pp. 287–288).

Hoos, H., Renz, K., & Gorg, M. (2001). GUIDO/MIR: An experimental musical information retrieval system based on GUIDO music notation. In J. S. Downie & D. Bainbridge (Eds.), *Proc. of the 2nd int. symposium on music information retrieval: ISMIR 2001* (p. 41-50).

McNab, R. (1996). *Interactive applications of music transcription.* MSc thesis, Department of Computer Science, University of Waikato, NZ.

McNab, R. J., Smith, L. A., Witten, I. H., Henderson, C. L., & Cunningham, S. J. (1996). Towards the digital music library: Tune retrieval from acoustic input. In *Digital libraries '96: Proceedings of the acm digital libraries conference* (pp. 11–18).

Mongeau, M., & Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities*, *24*, 161-175.

Parsons, D. (1975). *The directory of tunes and musical themes.* Cambridge: Spencer Brown.

Rand, W., & Birmingham, W. (2001). Statistical analysis in music information retrieval. In J. S. Downie & D. Bainbridge (Eds.), *Proc. of the 2nd int. symposium on music information retrieval* (p. 25-26). Indiana University, Bloomington, IN.

Shannon, C. E. (1948a, July). A mathematical theory of communication (part 1). *Bell System Technical Journal*, *27*, 379–423.

Shannon, C. E. (1948b, October). A mathematical theory of communication (part 2). *Bell System Technical Journal*, *27*, 623–656.

Uitdenbogerd, A. L., & Zobel, J. (1999). Melodic matching techniques for large music databases. In *Acm multimedia 99* (pp. 57–66). Orlando, Florida.

Witten, I., Bainbridge, D., & Boddie, S. (2001). Greenstone: open source DL software. *Communications of the ACM*, *44*(5), 44.

Witten, I., Moffat, A., & Bell, T. (1999). *Managing gigabytes: compressing and indexing documents and images*. San Francisco, CA: Morgan Kaufmann.

Witten, I., R., M., Boddie, S., & Bainbridge, D. (2000, June). Greenstone: a comprehensive open-source digital library software system. In *Proceedings of the fifth ACM conference on digital libraries* (p. 113-121). San Antonio, Texas.

Wu, S., & Manber, U. (1992). Fast text searching allowing errors. *Communications of the ACM*, *35*(10), 83-91.

Footnotes

[1]Downie uses only the pitch aspect of melodies, and does not consider absolute pitch. Here, the queries are represented using absolute pitch and have rhythm information. This change causes the numerical results presented here to be different from those in Downie's work.

Algorithm	**Rhythm only**	**Pitch only**			**Pitch and rhythm**		
		abs	int	con	abs	int	con
State-based	0.106	0.593	0.396	0.009	1.000	0.963	0.829
6-grams	0.096	0.582	0.389	0.009	1.000	0.963	0.826
5-grams	0.077	0.561	0.373	0.006	0.997	0.958	0.816
4-grams	0.054	0.505	0.324	0.002	0.991	0.954	0.772
3-grams	0.030	0.323	0.234	0.001	0.987	0.943	0.546

Table 1

Precision values for the folk collection, using exact matching and error-free queries.

Figure Captions

Figure 1. A workbench for symbolic music information retrieval.

Figure 2. (a) Recall and (b) precision values for the folk collection, using approximate matching and queries with one random error.

Figure 3. Number of songs returned from the folk collection with the state-based matching, 3-grams and 6-grams algorithms, using (a) exact matching (b) approximate matching.

Figure 4. Time to query the folk collection as query length increases, using dynamic programming, state-based matching, and 6-grams algorithms in approximate mode.

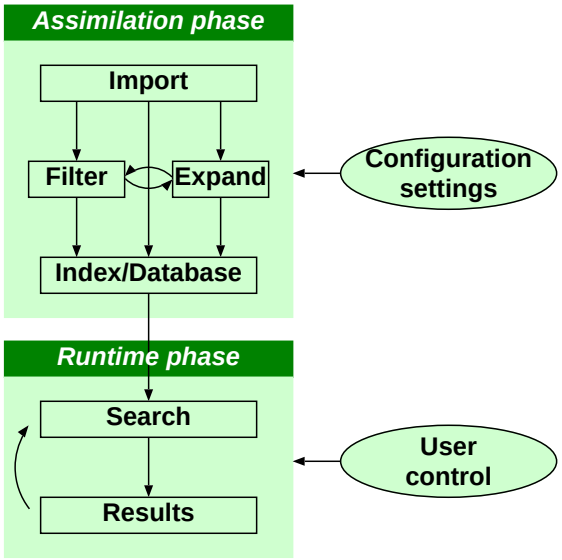
Figure 5. Time to query the folk collection as collection size increases, using dynamic programming, state-based matching, and 6-grams algorithms in approximate mode.

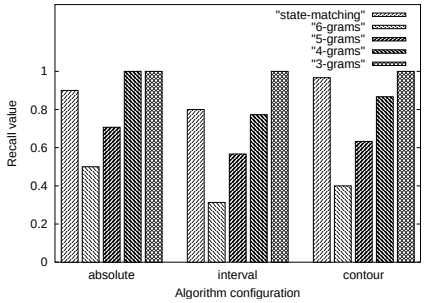
Figure 6. (a) Recall and (b) precision values for the folk collection, using state-based matching, 3-grams and hybrid algorithms in approximate mode and queries with one random error.

Figure 7. Number of songs returned from the MIDI collection using the state-based matching, 3-grams and hybrid algorithms in approximate mode.

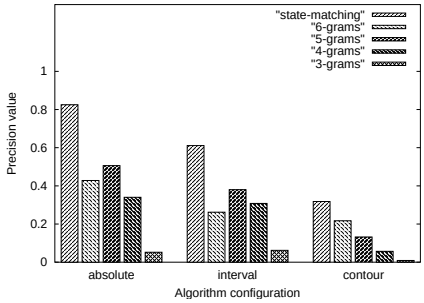
Figure 8. Time to query the MIDI collection as query length increases, using state-based, 3-grams and hybrid algorithms in approximate mode.

Figure 9. Time to query the MIDI collection as collection size increases, using state-based matching, 3-grams and hybrid algorithms in approximate mode.

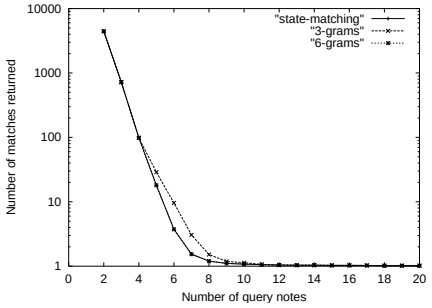

Assimilation phase
Import
Filter
Expand
Configuration settings
Index/Database
Runtime phase
Search
User control
Results


"state-matching"
"6-grams"
"5-grams"
"4-grams"
"3-grams"
Recall value
0
0.2
0.4
0.6
0.8
1
absolute
interval
contour
Algorithm configuration

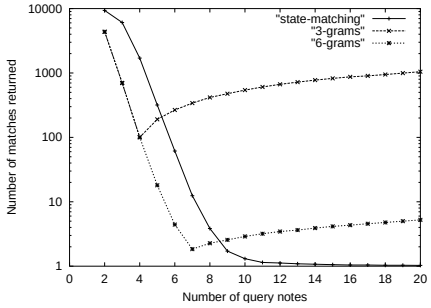
(a)


"state-matching"
"6-grams"
"5-grams"
"4-grams"
"3-grams"
Precision value
0
0.2
0.4
0.6
0.8
1
absolute
interval
contour
Algorithm configuration

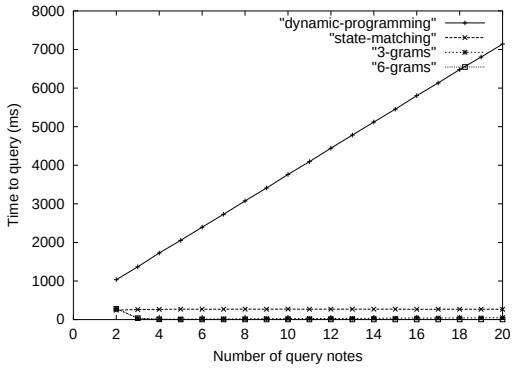
(b)

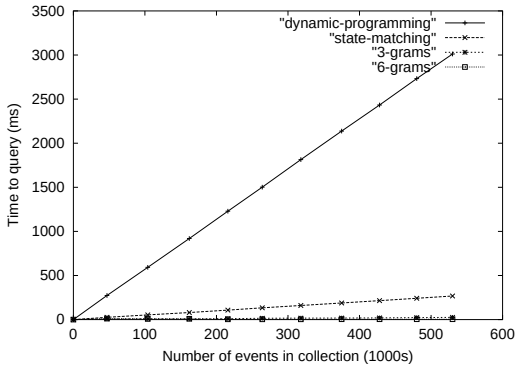

"state-matching"
"3-grams"
"6-grams"
Number of matches returned
Number of query notes
10000
1000
100
10
1
0
2
4
6
8
10
12
14
16
18
20

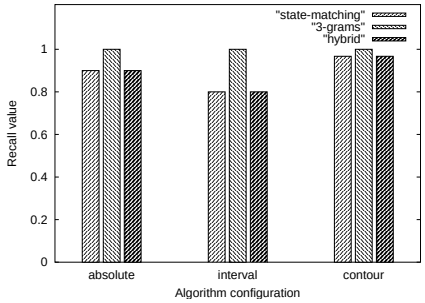
(a)


"state-matching"
"3-grams"
"6-grams"
Number of matches returned
Number of query notes
10000
1000
100
10
1
0
2
4
6
8
10
12
14
16
18
20

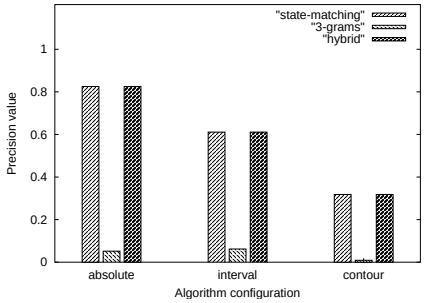
(b)


"dynamic-programming"
"state-matching"
"3-grams"
"6-grams"
Time to query (ms)
Number of query notes
0
1000
2000
3000
4000
5000
6000
7000
8000
0
2
4
6
8
10
12
14
16
18
20

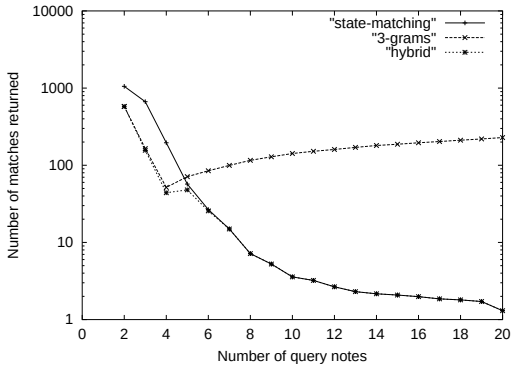

"dynamic-programming"
"state-matching"
"3-grams"
"6-grams"
Time to query (ms)
Number of events in collection (1000s)
0
500
1000
1500
2000
2500
3000
3500
0
100
200
300
400
500
600

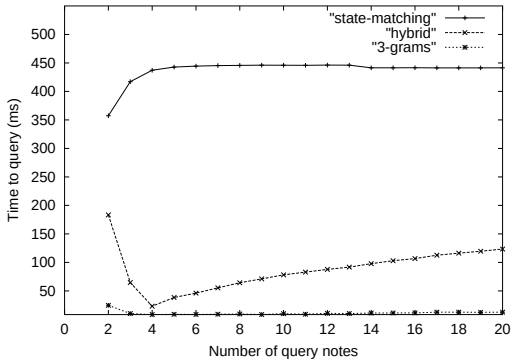

"state-matching"
"3-grams"
"hybrid"
Recall value
0
0.2
0.4
0.6
0.8
1
absolute
interval
contour
Algorithm configuration

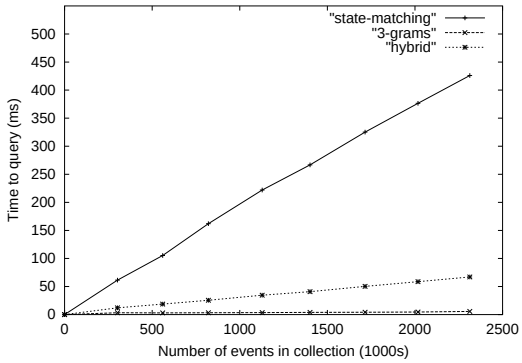
(a)


"state-matching"
"3-grams"
"hybrid"
Precision value
0
0.2
0.4
0.6
0.8
1
absolute
interval
contour
Algorithm configuration

(b)


"state-matching"
"3-grams"
"hybrid"
Number of matches returned
Number of query notes
10000
1000
100
10
1
0
2
4
6
8
10
12
14
16
18
20


"state-matching"
"hybrid"
"3-grams"
Time to query (ms)
Number of query notes
0
2
4
6
8
10
12
14
16
18
20
50
100
150
200
250
300
350
400
450
500


"state-matching"
"3-grams"
"hybrid"
Time to query (ms)
0
50
100
150
200
250
300
350
400
450
500
0
500
1000
1500
2000
2500
Number of events in collection (1000s)