Effectiveness of HMM-Based Retrieval on Large Databases

Jonah Shifrin
EECS Dept, University of Michigan
110 ATL, 1101 Beal Avenue
Ann Arbor, MI 48109-2110
jshifrin@umich.edu

William Birmingham
EECS Dept, University of Michigan
110 ATL, 1101 Beal Avenue
Ann Arbor, MI 48109-2110
wpb@eecs.umich.edu

Abstract

We have investigated the performance of a hidden Markov model QBH retrieval system on a large musical database. The database is synthetic, generated from statistics gleaned from our (smaller) database of musical excerpts from various genres. This paper reports the performance of several variations of our retrieval system against different types of synthetic queries on the large database, where we can control the errors injected into the queries. We note several trends, among the most interesting is that as queries get longer (i.e., more notes) the retrieval performance improves.

1 Introduction

For the past couple years, our group has been working on the VocalSearch (Birmingham 2003) system, part of the larger MusArt project (Birmingham, Dannenberg et al. 2001; Birmingham 2002; Shifrin 2002). VocalSearch is a query-by-humming (QBH) system for music search and retrieval. The system has a database of musical pieces. Each piece in the database is represented by a set of monophonic *themes,* which are encoded as hidden Markov Models (HMMs). Each HMM is built automatically from a single theme. Each query is treated as an observation sequence. Similarity between a query and a theme is assessed by determining the probability the HMM could have generated the query. The quality of a match between a piece and a query is given by the probability its best matching theme generated the query. The pieces are then ranked by match quality.

We, like most researchers in the MIR area, have evaluated our system on relatively small databases. Our largest experiments use a database of about 2600 themes (Shifrin 2002). Yet, small databases pose a problem, as we anticipate that MIR system will need to work on database sizes approaching millions of themes in the relatively near future. Consider that VocalSearch works with Apple's iTunes, a popular computer-based media player. Assume a typical pop song is roughly five minutes long. If songs are stored as 128kbps mp3 files, a 100-gigabyte hard drive can hold 22,365 songs. Given that PCs are already shipping with 100-gigabyte drives, we can expect VocalSearch will need to deal with a database of this size in the very near future.

There are two main reasons why we have been limited to experiments on relatively small databases. First, our system depends on matching sung queries to themes. We can automatically generate themes from a song encoded in MIDI; however, the process is error prone and needs human supervision, thus limiting the size of the database. Perhaps more importantly, it is difficult to get a sufficiently large number of queries to adequately cover a large database(Bainbridge 2002; Downie 2002). Controlling for singer variation in ability, knowledge, and so forth, makes the experimental process long and complex..

In order to sidestep the labor-intensive steps involved in creating and querying a large song database, we synthetically generated both a 50,000-theme database and several corresponding sets of 5100 queries. In this paper, we report on the performance of our HMM-base retrieval system for this database and query set configurations. We also report on how varying parameters of the HMM system impact performance.

2 System Overview

In this section, we provide a brief overview of our system. Figure 1 gives a block diagram overview of our system.

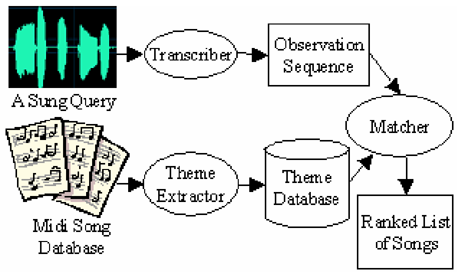
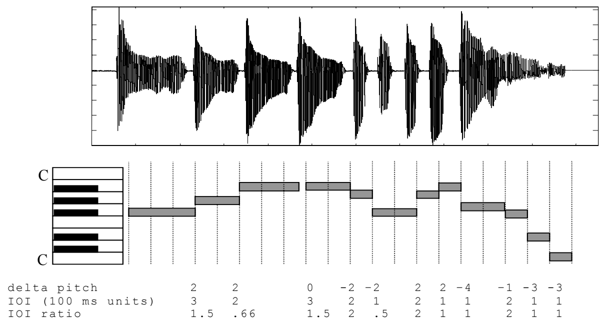


Figure 1: System Diagram

2.1 Queries

For an audio query, a singer records a .wav file using VocalSearch's recording program. The .wav file is transcribed into a MIDI-based representation using a pitch tracker and note segmenter.

Figure 2 shows a time amplitude representation of a query, with its pitch-tracker output on piano roll, and a sequence of values derived from the MIDI representation (the *deltaPitch*, *IOI* and *IOIratio* values).



delta pitch	2	2	0	-2	-2	2	2	-4	-1	-3	-3
IOI (100 ms units)	3	2	3	2	1	2	1	1	2	1	1
IOI ratio	1.5	.66	1.5	2	.5	2	1	1	2	1	1

Figure 2: A sung query (adapted from Shifrin, et al. (Shifrin 2002))

We define the following state model (Shifrin 2002):

- A *note transition* between note *n* and note *n*+1 is described by the duple <*deltaPitch*, *IOIratio*>.
- *deltaPitch*$_n$ is the difference in pitch between note *n* and note *n*+1.
- The *inter onset interval* (*IOI*$_n$) is the difference between the onset of notes *n* and *n*+1.
- *IOIratio*$_n$ is IOI_n/IOI_{n+1}. For the final transition, $IOI_n = IOI_n / duration_{n+1}$.

2.2 Targets

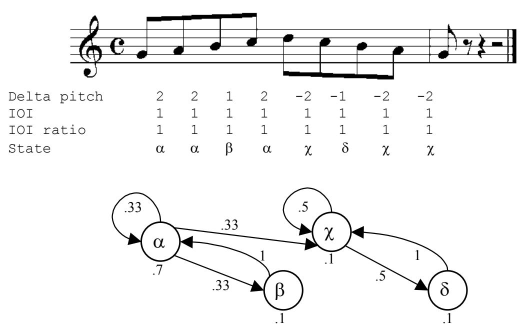
The targets in our database are themes, and are represented by Markov Models (MM). A MM is a weighted automaton that consists of:

- A set of states, $S = \{s_1, s_2, s_3, \ldots, s_n\}$.
- A set of transition probabilities, *T*, where each $t_{i,j}$ in *T* represents the probability of a transition from s_i to s_j.

In this model, the probability of transitioning from a given state to another state is assumed to depend only on the current state. This is known as the Markov property.

Each state*, s*, in the Markov model consists of the duple <*deltaPitch*, *IOIratio*>. Figure 3 shows a theme as a MM.

MMsare too simple for our application, as they require that each state in the model correspond to a fully observable state in the world. In QBH applications, this is almost never the case, as there are many errors introduced by the singer, the transcriber, and the segmenter (Mongeau and Sankoff 1990; Raphael 1999; Meek and Birmingham 2002). For example, a user (singer) may not be able to generate intervals greater than a perfect 5th, so that, from the system's perspective, it is impossible to distinguish between a minor 6th and a minor 7th. Thus, we need to account for these various types of errors by extending the MM to a hidden Markov Model, or HMM. The HMM allows us a probabilistic map of observed states to states internal to the model (hidden states).



Delta pitch	2	2	1	2	-2	-1	-2	-2
IOI	1	1	1	1	1	1	1	1
IOI ratio	1	1	1	1	1	1	1	1
State	α	α	β	α	χ	δ	χ	χ

Figure 3: Markov model for a scalar passage (adapted from Shifrin, et al. (Shifrin 2002))

A model that explicitly maintains a probability distribution over the set of possible observations for each state is called a hidden Markov model (HMM). More formally, an HMM requires two properties in addition to the properties required for a standard Markov model:

- A set of possible observations, $O=\{o_1, o_2, o_3, \ldots, o_n\}$.
- A probability distribution over the set of observations for each state in *S*.

In our approach, a query is a sequence of observations. Each observation is a note-transition duple, <*deltaPitch, IOIratio*>. Musical themes are represented as hidden Markov models whose states also corresponds to note-transition duples.

Thus P(*O*|*S*) is the P(*note transition*$_O$ | *note transition*$_S$*)*. This probability requires that both the observed deltaPitch and IOIratio must be computed in terms of the note transition probability. deltaPitch and IOIratio are assumed to be conditionally independent.

So, P(*note transition*$_O$ | *note transition*$_s$*)* = *P(deltaPitch*$_O$*|deltaPitch*$_S$*)*P(IOIratio*$_O$*|deltaPitch*$_S$*)*.

If one assumes pitch quantized at the half step and that a singer will jump by no more than an octave between notes, there are 25 possible deltaPitch values. Our P(*deltaPitch*$_O$*|deltaPitch*$_S$*)* table is based on the data from a study (Pardo and Birmingham 2003) where singers tried to match a pitch interval by singing it after hearing it. The P(*IOIratioO|IOIratioS)* is based on the ratio of two IOIratios. Where, the ratio of IOIratios is the *IOIratio*$_O$*/IOIratio*$_S$. This provides a ratio as to how close the observation is to the (hidden) state. Mapped in log space into 27 buckets in a normal distribution it provides our table for *P(IOIratioO|IOIratioS)*.

2.3 Matcher

Using the Forward algorithm, our matcher returns the probability that the HMM generated the observation sequence. (Rabiner and Juang 1993; Durbin, Eddy et al. 1998) The Forward algorithm sums up the probability of traversing all paths through the HMM.

3 Database Generation

The synthetic database that we generated follows the qualities of our existing theme database, which is based on a collection of Beatles songs. To generate the synthetic database, we analyzed the properties of a Beatles theme database that we have experimented with for a year. The Beatles database is a MIDI collection of 260 Beatles MIDI songs, which were processed into a database of 284 themes.

Most of the Beatles songs were modeled with one theme that encapsulated all hooks or riffs. However, a few of the songs had two or three distinct themes.

We analyzed three properties of the database, *deltaPitch*, *IOIratio* and length (of the themes). For deltaPitch, we examined the distribution of the deltaPitch's in the database. Any interval jump greater than +12 semitones or less than –12 semitones was quantized to +12 and –12 semitones, respectively. Thus, there were 25 total possible deltaPitch values. In the Beatles database, there were 12,379 pitch intervals. Figure 4 represents the deltaPitch distribution.

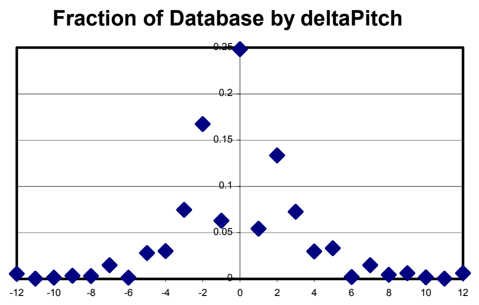


Figure 4: *deltaPitch* distribution in Beatles database

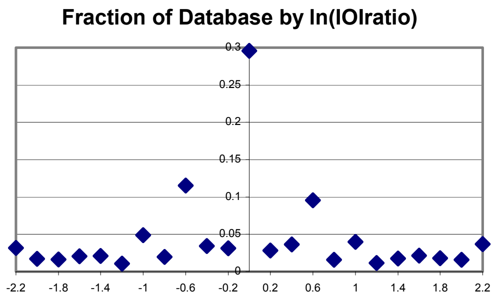


Figure 5: *IOIratio* distribution in Beatles database

Similarly, the Beatles database contained 12,379 IOIratios. We quantized the IOIratios into 23 bins distributed on a logarithmic scale. We chose logarithmic scale for a uniform distribution. The smallest IOIratio bin is ln(-2.2) and the largest IOIratio bin is ln(2.2), with bins spaced at every ln(.2). Figure 5 represents the IOIratio distribution.

Finally, we created a theme-length distribution based on the number of notes contained by the themes in the Beatles database The median theme length was 40 notes, with a standard deviation of 19.91 notes.

We created the database by using our observed distributions for deltaPitch, IOIratio and theme length to randomly generate each synthetic theme. Our resulting synthetic database contained 50,000 themes. This synthetic theme database corresponds to a database of roughly 22,000 songs assuming an average of 2.27 themes per song (which is typical for our databases).

4 Synthetic Query Generation

We generated 5100 queries from different targets in the database for each set. These queries were evenly distributed by length. One hundred queries were created ranging in length from five notes to 55 notes. This range represents bounds for the longest and shortest queries we observed in experiments with VocalSearch.

The synthetic queries consist of a sequence of notes. Each note has a pitch, an onset time and duration. The synthetic queries are equivalent to sung queries that have been pitch tracked and segmented. Queries were varied for length to determine if the matcher's performance is related to the query's length. If a query contains too few notes, it may not have enough information to distinguish its target from the other targets in the database; if the query were sufficiently long, it might not match our themes, which may be shorter than the query.

4.1 Perfect Queries

A perfect query is an exact excerpt from a target in the database. The purpose of presenting perfect queries to the matchers is to create a baseline statistic for testing our system. If our matcher cannot consistently rank perfect queries of length *l* first overall, then it cannot be expected to rank an imperfect query of length *l* first.

4.2 Imperfect Queries

An imperfect query simulates the type of query we expect to be input into our system. The typical user makes a substantial number of errors when singing, as reflected in *deltaPitch* and *IOIratio* ratios. In addition, pitch trackers and segmenters introduce errors.(McNab, Smith et al. 1996; Madden, Smith et al. 2001; Meek and Birmingham 2002)

An assumption we make is that we have a perfect probabilistic model of singer and transcription error. What we mean is that, given our statistics about singer error, we will only generate queries consistent with those statistics. We note that it is entirely possible that these statistics are deficient, which would mean that our synthetic queries are not representative of real-world queries.

Thus, our imperfect query experiments represent *a best-case* simulation. If we cannot successfully perform retrieval on a matcher that uses an exact observation probability distribution on a large database, then that matcher could not be expected to successfully retrieve on large databases. If, however, a matcher can successfully perform retrieval with our query approximation, it indicates that the matcher is capable of performing retrieval, provided it has an accurate observation probability distribution.

In order to generate a perfect query, we first select a length *l* subsequence of a target in the database. Then, we transform the selected portion of the target into a sequence, *S,* of <*deltaPitch*, *IOIratio*> duples.

An imperfect query is created as follows. For each state s_i in *S,* we generate an observation o_i based on the same probability tables used to calculate P($O|S$) in the matcher.

As an example of how the queries were generated, say that $Sm = \alpha$, and that $P(\beta|\alpha) = 0.7$, $P(\chi|\alpha) = 0.3$, the observation generated *Om,* would have a probability of 0.7 of being β and a probability of 0.3 of being χ.

This set of imperfect queries assumes users introduce only cumulative error. Cumulative error means that the singers shift their pitch and rhythm baseline as they go. For example, consider a target with *deltaPitch* <+2,-2> and an *IOIratio* of <1,1>. If the first observed *deltaPitch* in the query is +4, using a cumulative error distribution, there is the same likelihood that the second observed *deltaPitch* would be -2 as it would have been if the first observed *deltaPitch* had been a +2.

Some research (Meek and Birmingham 2002; Pollastri 2002) suggests that local errors are more common than cumulative errors. For a local error, if the first observed *deltaPitch* was a +4, the singer would be more likely to query a *deltaPitch* of -4 for the second observation to keep their initial baseline. Figure 6 demonstrates the difference between local errors and cumulative errors. Currently, the MIR community is divided on whether cumulative or local errors are more prevalent.

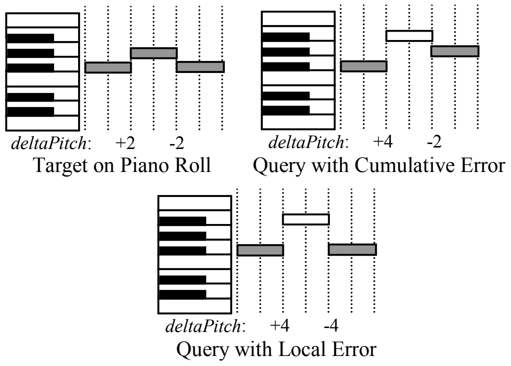


Figure 6: Example of cumulative and local errors

The HMM-based retrieval system used in the experiments presented in this paper was designed under the assumption that singers shift their baseline as they go, introducing cumulative errors. Thus, the imperfect queries were created under the assumption of only cumulative errors.

4.3 Query Insertions and Deletions

One common aspect of real-world queries is insertions and deletions. Persons and pitch trackers/segmenters commonly insert and delete notes in queries, relative to the targets in our database. According to experiments done by Meek on a set of 80 folk-tune queries with lyrics, the P("no edit") = .81, where no edit means that there was no insertion or deletion. P("insertion") = 0.06, and P("deletion") = 0.13. Thus, we created a query set modeled after the imperfect queries, except it also had insertions and deletions.

For note insertion, a note *n* with duration *d* and pitch *p*, is divided into notes n_1 and n_2. Both n_1 and n_2 have duration *d/2* and pitch *p*. Thus, a note is added, but the rhythm is preserved.

For a deletion, consider the note n_2, which is preceded by note n_1 with note onset time o_1 and pitch p_1, and followed by note n_3 with note onset time o_3. If note n_2 is deleted, note n_1 still has pitch p_1 and has duration o_3-o_1. So, the preceding note is lengthened so it "covers" the duration of the skipped note. It is worth noting that since our representation is dependant only on the *interOnsetInterval* it makes no difference whether we extend note *n1* until the onset of *n3* or simply delete note *n3*. Figure 7 gives an example of an insertion and a deletion.

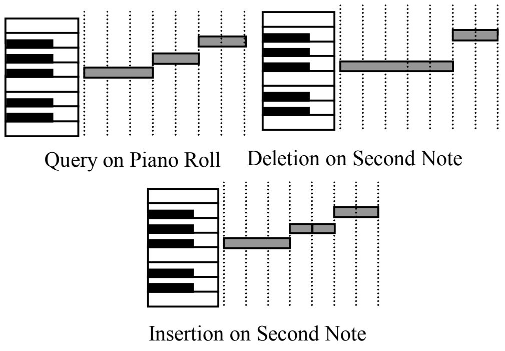


Figure 7: Example of Insertions and Deletions

While the set of imperfect queries is intended to show whether retrieval is possible on a large database if the matcher's topology reflects the properties of the queries, the dataset of insertions and deletions is meant to reflect real-queries. This dataset allows us to examine our system's performance on queries that do not directly conform to the HMM topology (i.e., the HMM does not directly model note deletions and insertions). Queries with insertions and deletions allows us to determine the sensitivity to the types of errors we expect to see in real queries.

5 Experimental Results

In this section, we present the results of our experiments. For the experiment sets, we used three different types of HMM matchers. A matcher that only considers the observed pitch, a matcher that only considers the observed duration, and a matcher that considers both the observed pitch and duration.

Each matcher was tested against three sets of 5100 queries. The query sets were perfect queries, imperfect queries, and imperfect queries with insertions and deletions. For each query, the ranking of the target that generated it in our database was recorded. Each query could rank between one and 50,000.

During each set of experiments with a matcher, we recorded both the *Mean Reciprocal Rank (MRR)*and the median rank of all queries in a set of length *l*. The formula for the MRR of *N* queries with rank r_i is: $1/N\sum_{i=1}^{N}1/r_i$. Since each set of queries contained 100 queries of length *l*, with *l* ranging from five to 55, each experiment with a query set returned 51 MRR values.

5.1 Results of Pitch and Duration Matcher

The MRR and median results by length of the matcher that considers pitch and duration for all three query experiments are given in Figure 8. The median results are on a logarithmic scale.

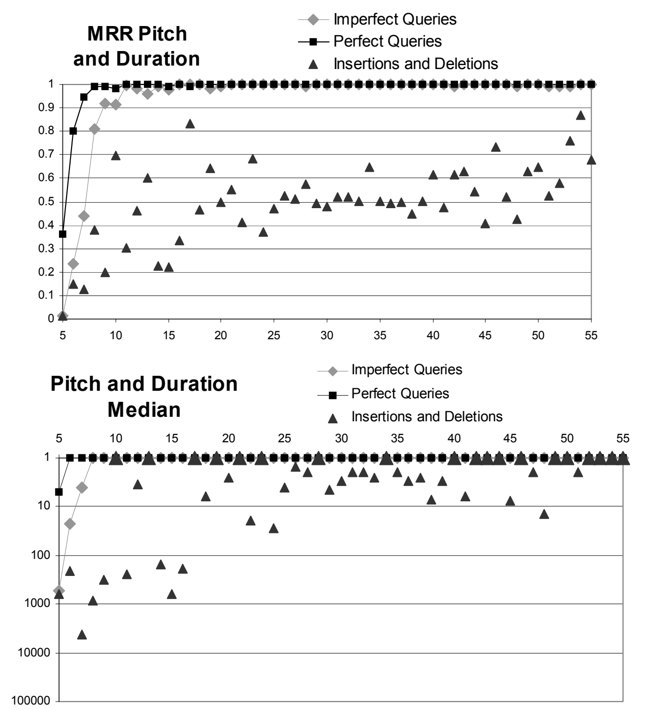


Figure 8: MRR value vs. query length for all three query-types on the pitch and duration matcher

In the two experiments without insertions and deletions, the matcher clearly benefits from longer queries. While these effects are less pronounced on the query set containing insertions and deletions, the results still suggest this trend. Longer queries provide the matcher more information, increasing the likelihood of distinguishing the intended target from other targets.

Longer insertion and deletion queries do not necessarily improve retrieval performance. Due to the topology of our model, an insertion or deletion causes the path traversed by the Forward algorithm to be split into two paths through the target. These disjoint sub-paths each will have a lower probability then a "full-length" path. The overall effect of the disjoint paths is to degrade a target's score, and thereby decrease the information benefit of longer queries.

5.2 Results of Matcher Using Pitch Only

The MRR and well as the median results by length for the matcher that only considers pitch are displayed in Figure 9. The median results are displayed on a logarithmic scale.

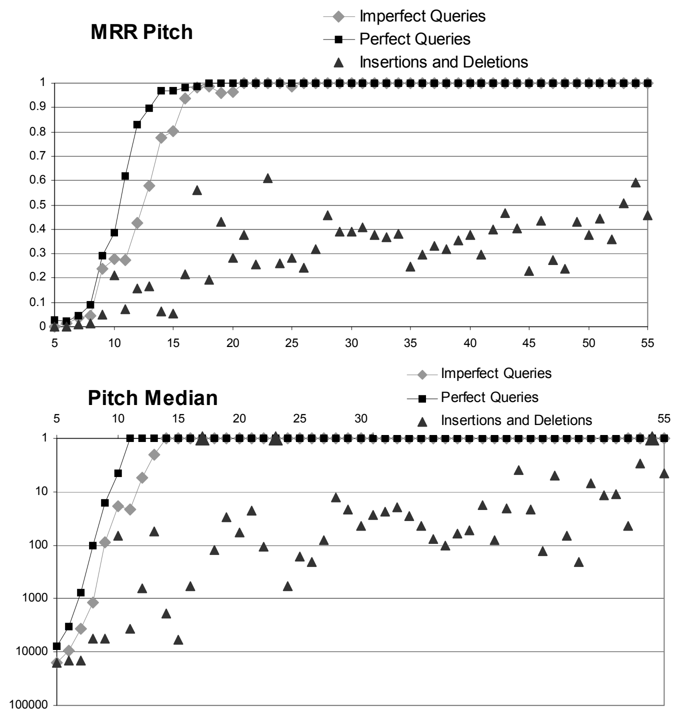


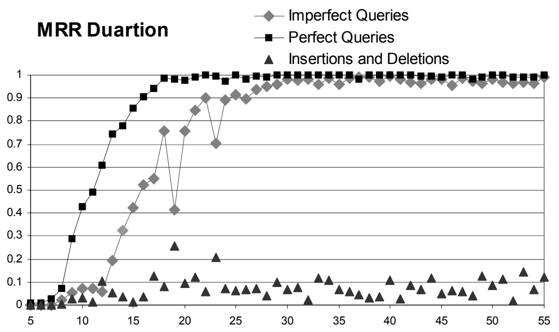
Figure 9: MRR and mean value vs. query length for all three query types on the pitch matcher

As was the case for pitch and duration experiments, the longer the queries, the better the results. The slightly worse performance on the imperfect and perfect query experiments on the matcher that uses only pitch, as opposed to the matcher that uses both pitch and duration shows the importance of the duration information.

There is even more significant degradation on the performance of the insertion and deletion query set on the matcher that only considers pitch, compared to the matcher that considers both pitch and duration. Since the matcher is only comparing the deltaPitchs of the observation and the targets, as insertions and deletions are introduced there is a greater likelihood that sub-path's of the other targets in the database will match the query then there is in the case of a matcher that compared both the pitch and duration.

5.3 Results of Matcher that Uses Duration Only

The MRR and median results by length for the matcher that only considers duration are given in Figure 10. The median results are displayed on a logarithmic scale.



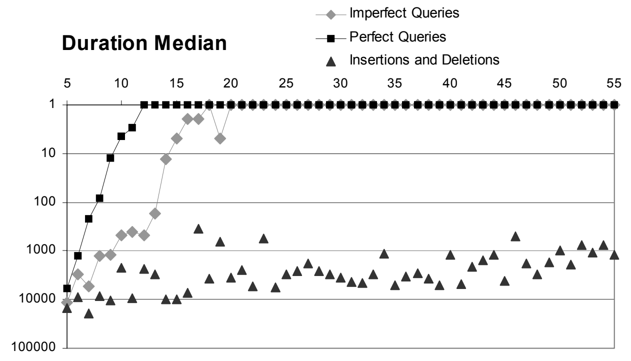


Figure 10: MRR and mean value vs. query length for all three query types on the duration matcher

The pitch-only matcher outperforms the duration-only matcher for all three query sets because of our implementation. Pitch contains more information about a song than duration. The pitch data is used to compare the actual deltaPitch of an observation and a target, while the duration data is used to compare the ratio of IOIratios The ratio of IOIratios is merely a measure of the similarity of durations, and it contains no real indication of rhythm: IOratio models tempo variation well. In cases where rhythm is a critical determining feature for a query, IOratio is not a good measure.

There is significant performance degradation on the query set with insertions and deletions for the duration-only matcher compared to the pitch-only matcher. Since the duration matcher considers less information, there is an even greater chance traversing the sub-paths of the different targets in the database with a high probability. Since the duration-only matcher traverses so many paths with a high probability when queries contain insertions and deletions, query length has a miniscule effect on performance.

6 Summary and Conclusions

There are several trends apparent from the experimental data. The first trend is that longer queries have a positive effect on retrieval performance, which is as we expected. As the size of a database grows, we expect that this effect will be even more pronounced. Moreover, we expect these results to generalize across retrieval algorithms; in other words, we believe that the "query-length" effect is not dependent on our particular HMM topology, but will extend to string matchers, different HMM topologies, and so forth.

The one caveat is that insertions and deletions mitigate the "query-length" effect. We believe that this mitigation is more an artifact of our particular retrieval method. We are looking at ways around this problem, such as using a different algorithm for ranking (e.g., Viterbi) and modifying our model topology.

One possible topological improvement would be to extend our model to include insertion and deletion states. This would allow the observation to match against a longer path in the target, and the subsequent matching of other target's sub-strings would not have as dramatic an effect. The HMM architecture is easily modeled to include insertions and deletions (Durbin, Eddy et al. 1998). Thus, we would no longer have problem of traversing disjoint sub-paths each time an insertion or deletion is encountered. A potential drawback of this approach is that search time will increase significantly when the topology admits more paths.

The second trend is that all the experiments show a saturation point, where retrieval performance reaches a maximum with fairly well-defined inflection point. We were expecting more gradual performance improvement without a saturation point. The shape of these curves gives us guidance about the accuracy of our retrieval system. For example, on relatively short queries (i.e., to the "left" of the inflection point), we expect that ranking will not be very accurate. We are looking at ways to model accurately and possibly report it as a confidence factor to the user.

The experiments suggest changing our current duration model, particularly to one that more accurately represents rhythm.

The most encouraging result from these experiments is the performance of the imperfect queries. Our system successfully retrieved using the set of imperfect queries on the large synthetic database. This shows both the importance of obtaining an accurate observation model and that retrieval is possible on large databases. In future work, we plan to have a training mode for our system to create an accurate observation model specific to an individual. Accurately trained and modeled QBH HMMs systems can accurately perform retrieval on large music databases, and have the potential to be used for real world applications.

In addition, we intend to experiment with large datasets of queries collected from singers. We plan to gather aggregate statistics of the same type as did for the research presented here. We also plan to follow the same experimental plan evaluating our system.

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation under grant IIS-0085945. The opinions in this paper are solely those of the authors and do not necessarily reflect the opinions of the funding agencies.

We would also like to thank Bryan Pardo for many detailed comments on this paper, and Colin Meek for many helpful suggestions.

References

Bainbridge, D., J.R. McPherson, S.J. Cunningham (2002). *Forming a corpus of voice queries for music information retrieval: a pilot study*. ISMIR-2002, Paris, France.

Birmingham, W., Bryan Pardo, Colin Meek, Jonah Shifrin, (2002). "The MusArt Music-Retrieval System: An Overview." *D-Lib Magazine*.

Birmingham, W., Colin Meek, Kevin O'Malley, Bryan Pardo, Jonah Shifrin (2003). "Managing a Personal Music Library." *Dr. Dobbs Journal* Vol 28(9), September, 2003..

Birmingham, W. P., R. B. Dannenberg, et al. (2001). *MUSART: Music Retrieval Via Aural Queries*. International Symposium on Music Information Retrieval, Bloomington, Indiana.

Downie, S. J., S.J. Cunningham (2002). *Toward a theory of music information retrieval queries: System design implications.* ISMIR-2002, Paris, France.

Durbin, R., S. Eddy, et al. (1998). *Biological sequence analysis*, Cambridge University Press.

Madden, T., R. B. Smith, et al. (2001). *Preparation for Interactive Live Computer Performance in Collaboration with a Symphony Orchestra*. Proceedings of the 2001 International Computer Music Conference, Havana, International Computer Music Association.

McNab, R. J., L. A. Smith, et al. (1996). *Towards the digital music library: Tune retrieval from acoustic input.* Proceedings of Digital Libraries '96, ACM.

Meek, C. and W. P. Birmingham (2002). *Johnny Can't Sing: A Comprehensive Error Model for Sung Music Queries.* ISMIR 2002, Paris, France.

Mongeau, M. and D. Sankoff (1990). Comparison of Musical Sequences. *Melodic Similarity Concepts, Procedures, and Applications.* W. Hewlett and E. Selfridge-Field. Cambridge, MIT Press. **11**.

Pardo, B. and W. P. Birmngham(2003). *Name that Tune: A Pilot Studying in Finding a Melody from a Sung Query To Appear in JASIST.*

Pollastri, E. (2002). *Some considerations about processing singing voice for music retrieval*. ISMIR-2002, Paris, France.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ, Prentice Hall Signal.

Raphael, C. (1999). "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models." *IEEE Transactions on PAMI* **21**(4): 360-370.

Shifrin, J., Bryan Pardo, Colin Meek, William Birmingham (2002). *HMM-Based Musical Query Retrieval*. ACM International Joint Conference on Digital Libraries (JCDL), ACM.