Examining Synthetic Databases in Melodic Retrieval Testing

Charles Parker
Department of Electrical Engineering and Computer Science, Oregon State University
parker@cs.orst.edu

Abstract

We investigate the practice of using probabilistically generated melodies to do large scale evaluations of query-by-humming systems by running a set of sung queries against both real and synthetic databases, using an already verified type of "matching function" to map sung queries to the appropriate target. We find that the accuracy of the generative process can be improved by introducing a first-order Markov assumption into the model, though neither method of melodic generation is found to be a statistically consistant approximation of an actual database under our experimental conditions.

1 Introduction

The statement of the so-called "query-by-humming" problem is straight-forward: Given a sung melodic query and a database of songs, match the query to the correct song. Even casual examination of this problem reveals that it contains many non-trivial subproblems. A significant portion of these problems stem from the fact in most approaches to the problem an *event-based* representation for both the sung query and the target database is used. An event-based representation in this context is one that parses a standard audio file (in WAV format or the like) into discrete events, segmenting either by amplitude or by change in pitch.

For a sung query, this is quite difficult. We must divide the audio into frames and then use a pitch detection process on each frame. After this phase is complete, we must group frames of like pitch into notes. Both of these processes are inexact even with the best of singers. Worse yet, poor singers may render the melody incorrectly. The resulting sequence of events, then, has little chance of being exactly the same as the sequence composing the proper target song.

For a target song, the process of parsing discrete events is much harder still. Because target songs (again, in WAV/MP3 format) usually have more than one instrument playing at once, we have the problem of *polyphonic transcription*, which has a body of literature in its own right. Even if we were able to correctly parse the many events in a given polyphonic target, we would still have the issue of *thematic extraction*. This, again, is a problem that is often studied on its own.

In an attempt to separate the first process (query transcription) from the second (target transcription), a vast majority of the current query-by-humming approaches make the simplifying assumption that all targets in the database are in monophonic MIDI format. This allows us to evaluate our matching process separate from the process of transcribing the target database. There have been recent calls for a more unified approach (Dannenberg, Birmingham, Tzanetakis, Meek, Hu, and Pardo 2003), but everyone agrees that there is room for improvement in the simplified version of the problem, and that a better solution would hasten the emergence of a better solution to the more general problem.

Unfortunately, the simplifying assumption we have made raises a new issue in terms of testing: Whereas we have an abundance of target data in WAV/MP3 format (in the form of CD audio and MP3 music files), there is a comparative lack of data in monophonic MIDI form. When testing a query-by-humming system, many implementations limit their tests to databases of MIDI files that are small enough to be hand-constructed (Rao and Raju 2002; Meek and Birmingham 2002) - on the order of hundreds of songs - but most concede that any realistic system will need to deal with a number of songs in the tens of thousands.

To this end, recent work (Shifrin and Birmingham 2003; Dannenberg et al. 2003) has used *probabilistic melody generation* in order to generate large test databases of MIDI data. The implicit assertion here is that the retrieval characteristics of a probabilistically generated database will be the same as those of a real database of the same size. We will attempt to provide a rough empirical evaluation of that assumption, and then suggest a possible method for bringing such an assumption closer to reality.

2 Song Representation

Each event, or note, in a song has two chief components: A pitch and a duration. For query-by-humming purposes, most implementations make the assumption that only *relative* pitch and duration are significant. That is, we will not place a restriction that a query start on a particular pitch or that it will be sung at a particular tempo. We make only the assumption that, once started, all relative changes in pitch and duration

will match relative changes of the correct target.

Hence, following most accurate representations (Shifrin and Birmingham 2003; Pardo et al. 2004), we will represent both sung queries and target songs as a series of duples, $\{(p_1, d_1), (p_2, d_2), \ldots\}$, where p_i represents the *difference* in pitches between notes i and $i-1$, and d_i represents the *ratio* of the duration of note i to $i-1$ More formally, if e_i is a note with frequency f_{e_i} and duration t_{e_i}, then in an n-note sequence:

$$p_i = f_{e_i} - f_{e_{i-1}}$$

$$d_i = \frac{t_{e_i}}{t_{e_{i-1}}}$$

Of course, frequency here is specified in the log domain so that we are actually measuring difference in semitones on the western even-tempered scale.

We will also make this representation a discrete-valued one so that we can estimate the probabilities with ease later on. The discretization for the p portion of the above duple is obvious: We will use the number of half-tones (the interval) that the difference in pitch represents, within one octave on either side of the current pitch, and reserving an "out of bounds" value for either end, resulting in 26 separate values.

The discretization of the d component of the duple is more subtle. Work has been done to show that we can simply bin the real values of duration in order to obtain discrete values without too much of a sacrifice in performance, if we change the values to be in the log domain, and use at least six bins (Pardo and Birmingham 2002). We will select this representation as it is the most well-studied to date.

3 Melodic Generation

3.1 Probabilistic Generation

Now that we have discrete values for pitch and duration, the obvious way to generate a random melody is to first generate a single random "reference" event, then to generate a series of duples which represent the relative motion of the rest of the events in the melody. We will make the common assumption that duration and pitch are generated independent of one another. The probability of generating a given interval p^x can be estimated by simply counting the total number of occurrences of p^x and dividing by the total number of intervals in the database, with a similar process for estimating probabilities over possible durations. Generating a melody is a simple process of generating random numbers to obtain a series of these duples.

At this point, anyone remotely acquainted with musical composition will balk. Clearly, the process of composition cannot be be reduced to (or even readily approximated by) a model that generates random notes based on the probability of their occurrence in other melodies. There are two questions here: First, can we improve this process so that it reflects a notion of composition that is even somewhat closer to reality than the formulation above? Second, does this matter at all to our overall goal of creating an artificial database of targets with which we can model query-by-humming performance on a database of real targets? We will deal with the first question in the next section.

3.2 First-Order Markov Generation

To augment this process of probabilistic melody generation, we will introduce a *first-order Markov assumption*: Instead of assuming that all intervals and durations are generated independent of one another, we will now assume that the every generated duple (p_i, d_i) is probabilistically conditioned on the previously generated duple (p_{i-1}, d_{i-1}). We can express this conditioning in the form of a very simple *two time-step dynamic Bayesian network* as shown in figure 1.

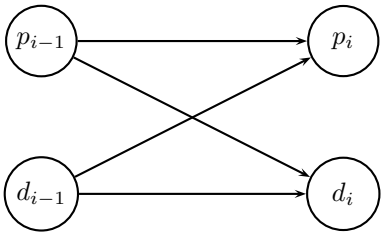

Figure 1: A simple probabilistic model for generating melodies.

Training the network is much same as was estimating the probabilities in the simpler case. The difference here is that we are estimating instead the *conditional* probability $P(p_i = p^x | p_{i-1} = p^y, d_{i-1} = d^z)$. That is, we are estimating that the current interval will be p^x given that the last duple generated was (p^y, d^z) where this is a specific interval, duration value pair. Thus, we need to estimate the *joint distribution* $P(p_i, p_{i-1}, d_{i-1})$. The training process, however, is still a matter of simple counting. For a more in-depth look at dynamic Bayesian networks, Markov models, and Bayesian learning and inference, please consult Russell and Norvig (2003).

Do we believe that our simple addition to the melodic generation process will have our system generating melodies that are correct by any human standard of composition? Certainly not. But we do believe that learning this more specific distribution will help us produce songs that are more realistic than the more simplistic case. Let us turn then, to evaluating these systems empirically.

Event	Probability
$P(p_i = +5)$	0.051
$P(p_i = +5 \mid p_{i-1} =$null ($p_i$ is first))	0.256
$P(p_i = +3)$	0.065
$P(p_i = +3 \mid p_{i-1} = +4)$	0.284

Table 1: Frequencies of some events that change drastically based on the previous event

4 Experimental Setup

To evaluate these methods, we construct a database of 150 sung queries over 40 singers and 12 songs. We then use a common database of about 2000 songs to learn the probabilities associated with each of our melody generators above. The experiments are run by evaluating how many of our queries give the highest score to the correct target using a well known method of query-target matching described below.

We start the database containing only the 12 correct target songs, and then grow it in three different ways: Using each of the two methods of melody generation above, and also by gradually adding songs from the aforementioned 2000 song "training" database. We will do each of the methods of random melodic generation 10 times in order to minimize random effects and also to estimate the statistical significance of any differences in performance that we observe.

We expect, of course, that the number of queries matching the correct target will decrease as we grow the database, inserting new, possibly similar targets. If it decreases faster for the "real" database than for the two generated ones, then it means that the songs we are generating are not realistic enough, and our matching process can tell the difference.

4.1 Target Database

The target database we will use is the Digital Tradition database of American folk songs, which is often used in the literature. The database contains mostly well known American/European folk tunes and is used many places in the literature (McNab et al. 2000). The subset of the database that we will use is 2280 songs with an average length of about 50 notes per song.

A preliminary investigation of some probabilities seems promising, as noted in table 1. We see that the interval of a perfect fourth upward (+5 semitones) has a *prior probability* of about 5%. However, when we look at only duples that are the first of a song, this probability is raised five-fold. This corresponds to the common practice of the dominant pickup in western music, used in *Auld Lang Syne*, Wagner's *Wedding March*, and the theme from the last movement of Brahms' *Symphony no. 1*, to name just a few songs.

The other probability shown is the probability of the minor third occurring in a song, which is 6.5% in general, but four times that when preceeded by a major third. This corresponds, of course, to the arpeggiation of a major chord. It is these types of dependencies that will be detected by our new model.

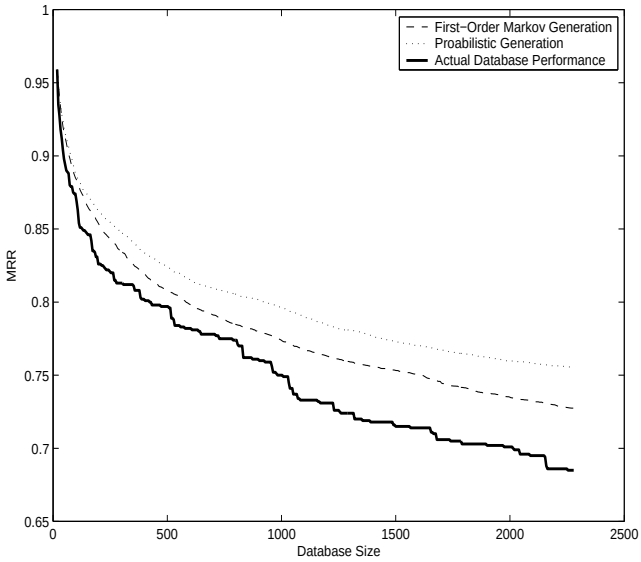

Figure 2: Mean reciprocal rank of all queries with increasing database size.

4.2 Query Database and Matching Scheme

The 150 queries we will use for evaluation are part of a massive body of collected query data which consists of about 1200 queries. The subjects in these queries were chosen as potential users of a query-by-humming system. They are typically competant singers but with no formal vocal training. In these queries, the melody is correctly rendered, and would be readily identifiable by a human listener.

The "matching function" we will use is the simple, event-based one outlined by Dannenberg et al. (2003), which, according to that work and others (Pardo et al. 2004), is the fastest algorithm that performs at state of the art levels.

5 Results

The results are plotted in figures 2 and 3. Figure 2 plots the *mean reciprocal rank* or MRR. This is a standard benchmark in TREC tests, and is again used often in the literature (Meek and Birmingham 2002; Pardo, Birmingham, and Shifrin 2004). We also plot, in figure 3 the percentage of queries which the correct target is ranked first out of all targets in the database - the ideal result.

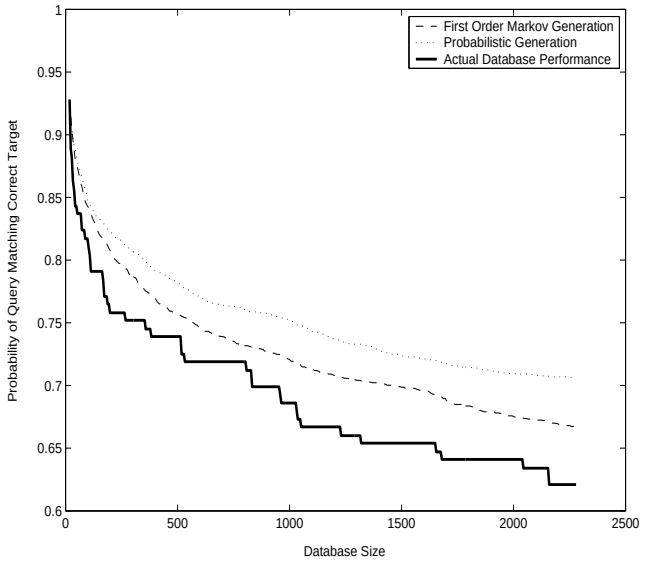

Figure 3: Probability of a query returning the correct target ranked first versus increasing database size.

We can see that, as we add songs in the database, the performance of the queries decreases with all three methods of melodic generation. However, we also see that the performance against the real database decreases faster than with either of the two generated databases. Furthermore, we see that the first order Markov assumption is indeed able to eliminate some of this gap between random generation and real composition.

Type	Mean	St. Dev.
Probabilistic Generation	0.755	0.010
First-Order Model Generation	0.728	0.012
Actual Database	0.685	—

Table 2: Statistics for the MRR at the largest database size

How significant are these differences? Table 2 sums up the results for MRR at the largest database size considered. We can use these statistics to perform the venerable *t-test* for statistical significance. With the above statistics and a sample size of 10 for each probabilistic method, we find that the t-test yields $P < 0.0001$. Performing the one-value t-test for each method against the actual database performance yields a similarly small P value. This tells us that we can be over 99.99% certain that the differences here are statistically significant.

6 Conclusions

We set out to test the assumption that probabilistically generated melodies accurate enough to simulate the retrieval characteristics of a real database. We have chosen a common retrieval method and shown that, in this particular case, this assumption is at least somewhat in error. However, introducing a simple first-order Markov assumption has helped us to capture something sufficiently interesting to make it statistically more accurate than the simpler model, again using our particular method of retrieval.

We must stress that these results are not by any means comprehensive across different methods of retrieval. What we have shown (and have intended all along to show) is that it is distinctly possible that the use of probabilistically generated melodies for testing may or may not give results that will not fully generalize to real databases of songs. This is a facet of the work that merits consideration, especially in cases where the validity of the work rests squarely on this assumption of generality.

Finally, we may entertain the notion that because a first-order Markov assumption appears to give us an increase in accuracy, that a second-order assumption will do better, and a third-order assumption better still. This would difficult to test, given that the size of the probability table grows exponentially with the order of the assumption. However, the use of a regression trees or some other type of approximation may be all that is needed.

References

Dannenberg, R. B., W. P. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo (2003). The musart testbed for query-by-humming evaluation. In *Proc. 4th International Symposium on Music Information Retrieval*.

McNab, R. J., L. A. Smith, I. H. Witten, and C. L. Henderson (2000). Tune retrieval in the multimedia library. *Multimedia Tools and Applications 10*(2/3), 113–132.

Meek, C. and W. Birmingham (2002). Johnny can't sing: A comprehensive error model for sung music queries. In *Proc. 3rd International Symposium on Music Information Retrieval*.

Pardo, B. and W. Birmingham (2002). Encoding timing information for musical query matching. In *Proc. 3rd International Symposium on Music Information Retrieval*.

Pardo, B., W. Birmingham, and J. Shifrin (2004). Name that tune: A pilot study in finding a melody from a sung query. *Journal of the American Society for Information Science and Technology 55*(4).

Rao, P. and M. A. Raju (2002). Building a melody retrieval system. In *National Conference on Communications*, Bombay, India.

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach* (second ed.)., pp. 492–583. Prentice Hall.

Shifrin, J. and W. P. Birmingham (2003). Effectiveness of HMM-based retrieval on large databases. In *Proc. 4th International Symposium on Music Information Retrieval*.