Johnny Can't Sing: A Comprehensive Error Model for Sung Music Queries

Colin Meek and William Birmingham
University of Michigan
Advanced Technologies Laboratory
1101 Beal Avenue
1-734-763-1561
meek@umich.edu

ABSTRACT

We propose a model for errors in sung queries, a variant of the Hidden Markov Model (HMM). This is related to the problem of identifying the degree of similarity between a *query* and a potential *target* in a database of musical works, in the music retrieval framework. The model comprehensively expresses the types of *error* or variation between target and query: cumulative and non-cumulative local errors, transposition, tempo and tempo changes, insertions, deletions and modulation. Results of experiments demonstrating the robustness of the model are presented.

1. INTRODUCTION

Various approaches have been proposed for the identification of viable targets for a query in a music database. We are interested here in queries posed in the most natural format for untrained users, the voice. Our goal is to demonstrate a unifying model, expressive enough to account for the complete range of modifications observed in the performance and transcription of sung musical queries. Our model is capable of expressing the following transformations, relative to a stored musical piece, which we call a *target*:

- Transposition: the query may be sung in a different *key* or *register* than the target, or both.
- Modulation: over the course of a query, the singer may change transposition.
- Tempo: the query may be slower or faster than the target.
- Tempo change: over the course of a query, the singer may speed up or slow down.
- Non-cumulative local error: the singer might sing a note off-pitch or with poor rhythm.
- Cumulative local error: local errors altering or effecting subsequent events.
- Insertions and deletions: adding or removing notes from the target, respectively.

While various representations and models can effectively represent some of these elements, to our knowledge no existing model explicitly accounts for *all* of these elements.

An important contribution of this work is an in depth exploration of the nature of note insertions and deletions. We assert that traditional string-edit operations [9][10] must be extended in the musical context and to this end introduce the corollary operations: elaborations and joins. Experiments have demonstrated that naive string edit approaches do not provide sufficient precision for music in the retrieval context [16].

Existing work using HMMs for query-by-humming consider a subset of the error-classes identified above. Shifrin et al,[15] use a state representation that is flexible in terms of transposition and tempo, but otherwise is capable of expressing only *cumulative* local error. Durey [6] does not allow for either transposition or tempo scaling.

This expressiveness has a computational cost. Conceptually our model considers states for each permutation of pitch relationship, rhythm relationship and string-edit operation. We make the assumption of conditional independence among these elements. For instance, the probability that a singer will skip a note is assumed to be independent of how out of tune they sang the previous note in our model. These assumptions help control the computational cost, but also reduce the parameterization of the model, essential for training [14]. Shifrin et al, [15] make similar assumptions about the independence of local pitch and rhythm errors, thus considerably reducing the amount of data needed to train the model. We recommend various approaches to *parameter tying* [1] throughout this paper (effectively reusing various parameters in different models and/or parts of models) but stress that this is an open area of research in this field.

2. PROBLEM FORMULATION AND NOTATION

An assumption of our work is that *pitch* and *inter-onset interval* (IOI) adequately represent both the target and the query. This limits our approach to monophonic lines, or sequences of note *events*. An event consists of a $\langle Pitch, IOI \rangle$ pair. The IOI is the time difference between the onsets of successive notes, and the pitch is the MIDI note number[1].

A crucial observation is that we are dealing with a *note-level* abstraction of music. Other systems act on a lower-level representation of the query [6][11], a frame-based frequency representation. Various methods for the translation of frequency and amplitude data into note abstraction exist [13][15]. Our group currently uses a transcriber based on the Praat f_0-extractor [5], designed to analyze voice pitch contour. A sample Praat analysis is shown in Figure 8. Note that these processes are not perfect, and it is likely that error will be introduced in the transcription of the query.

[1] Musical Instrument Digital Interface (MIDI) has become a standard electronic transmission and storage protocol/format for music. MIDI note numbers essentially correspond to the keys of piano, where 'middle *C*' corresponds to the integer value 60.

Restricting ourselves to this *event* description of target and query ignores several elements of musical style, including dynamics, articulation and timbre, among others. Objectively and consistently characterizing these features is quite difficult, and as such we have little confidence they can be usefully exploited for music retrieval at this point. We acknowledge, however, the importance of such elements in music query/retrieval systems in general. They will likely prove essential in *refining* and/or *filtering* the search space [4][7].

We further simplify the representation using IOI quantization, and by representing pitch in terms of *pitch class*. IOI is quantized to a logarithmic scale, using q=29 quantization levels, within the range 30 msec. to 3840 msec., chosen such that there are precisely four gradations between an eighth note and sixteenth note (or quarter note and sixteenth note, and so forth.) This representation mirrors conventional notation in Western music, in which the alphabet of rhythmic symbols (eighth, quarter, half, etc.) corresponds to a logarithmic scale on duration (see Figure 1.)

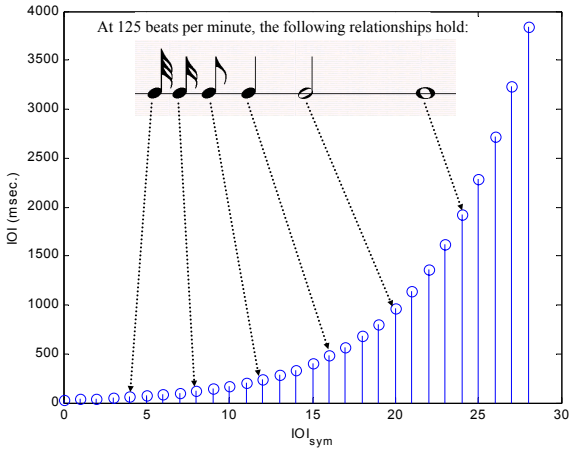

Figure 1: IOI quantization

We treat pitch in terms of *pitch class*, where all notes are folded into a single octave, and are considered in the context of the 12-tone, well-tempered scale. For instance, the frequency 453 Hz is "binned" into MIDI note number 70. The corresponding pitch-class is $70 \bmod_{12} = 10$. This addresses two issues: octave errors are quite common in some transcriber systems, and pitch class is an effective, if imperfect, musical [12] and perceptual [3] abstraction. In addition, this has the advantage of substantially reducing the model's "alphabet" size.

We choose discrete sets of symbols to represent pitch and duration since, as will be seen, a continuous representation would necessitate an unbounded number of states in our model. This second event representation is denoted:

$$\langle PC, IOI_{sym} \rangle, PC \in \{0,1,...,11\}, IOI_{sym} \in \{0,1,...,q-1=28\},$$

where PC is pitch-class and IOI_{sym} is the quantization symbol. For clarity, we will alternate between the representations describe above in this paper. The values are calculated as follows given a $\langle Pitch, IOI \rangle$ pair (where 30 and 3840 are the IOI values associated with the centers of the shortest and longest bins):

$$PC = Pitch \bmod_{12}, IOI_{sym} = round\left(\frac{\log IOI - \log 30}{\log 3840 - \log 30} \cdot (q-1)\right)$$

The goal of this paper is to present a model for query errors within the scope of this simple event representation. We will first outline the relevant error classes, and then present an extended Hidden Markov Model accounting for these errors. Taking advantage of certain assumptions about the data, we can then efficiently calculate the probability of a target model generating a query.

3. ERROR CLASSES

3.1 Insertions and Deletions

Insertions and deletions in music tend to influence surrounding events. For instance, when an insertion is made, the inserted event and its neighbor tend to occupy the temporal space of the original note: if an insertion is made and the duration of the neighbors is not modified, the underlying rhythmic structure (the beat) is changed. We denote this type of insertion a "warping" insertion. For instance, notice the alignment of notes after the warping insertion in Figure 2, indicated by the dotted arrows. The inserted notes are circled. For the non-warping insertion, the length of the second note is shortened to accommodate the new note.

With respect to pitch, insertions and deletions do not generally influence the surrounding events. However, previous work assumes this kind of effect: noting that intervallic contour tends to be the strongest component in our memory of pitch, one researcher has proposed that insertions and deletions could have a "modulating" effect [10], where the edit introduces a pitch offset, so that pitch intervals rather than the pitches themselves are maintained. We argue that *relative* pitch, with respect to the query as a whole, should be preserved. Consider the examples in Figure 3. The first row of numbers below the staff indicates MIDI note numbers, the second row indicates the intervals in semitones ('u' = up, 'd' = down.) Notice that the intervallic representation is preserved in the modulating insertion, while the overall "profile" (and key) of the line is maintained in the non-modulating insertion.

The effects of these various kinds of insertions and deletions are now formalized, with respect to a target $\{\langle Pitch_a, IOI_a \rangle, \langle Pitch_b, IOI_b \rangle\}$ and a query $\{\langle Pitch_c, IOI_c \rangle, \langle Pitch_{insert}, IOI_{insert} \rangle, \langle Pitch_d, IOI_d \rangle\}$, where $\langle Pitch_{insert}, IOI_{insert} \rangle$ is the inserted event. Note that deletion is simply the symmetric operation, so we will show examples of insertions only (Figure 2 and Figure 3):

- Effects of a warping insertion on IOI: $IOI_c = IOI_a, IOI_d = IOI_b$
- Effects of a non-warping insertion on IOI: $IOI_c = IOI_a - IOI_{insert}, IOI_d = IOI_b$
- Effects of a modulating insertion on pitch: $Pitch_c = Pitch_a, Pitch_d = Pitch_{insert} + \underbrace{Pitch_b - Pitch_a}_{\text{pitch contour}}$
- Effects of a non-modulating insertion on pitch: $Pitch_c = Pitch_a, Pitch_d = Pitch_b$

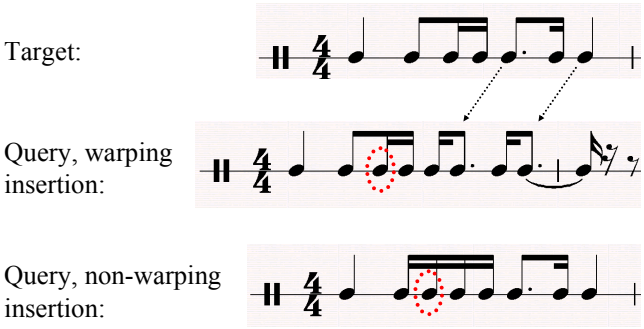

Figure 2: Warping and non-warping insertions

In our current model, non-modulating and non-warping insertions and deletions are handled explicitly. The other types of edit are represented *in combination with* other error classes. For instance, a modulating insertion is simply an insertion combined with a modulation. Consider for instance insertions or deletions introduced by an imperfect transcriber. In this case, we clearly would not expect the onset times or pitches of surrounding events to be influenced. The relationships amongst successive events must be modified to avoid warping and modulation! Reflecting this bias, we use the terms "join" and "elaboration" to refer to deletions and insertions respectively. Mongeau and Sankoff [20] use a similar notion of insertion and deletion, described as "fragmentation" and "consolidation" respectively.

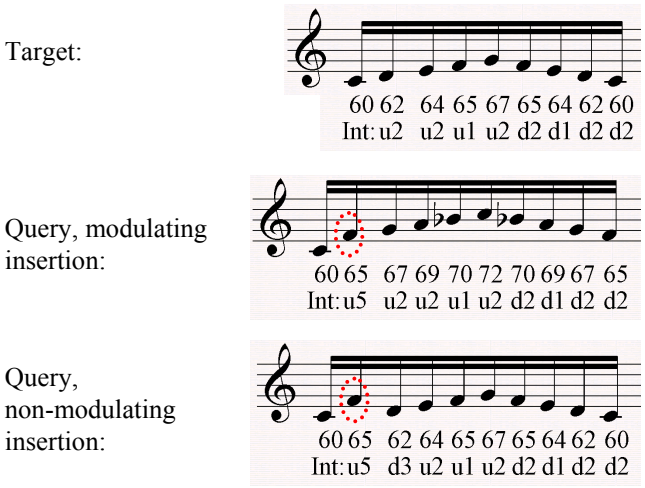

Figure 3: Modulating and non-modulating insertions

3.2 Transpositions and Tempo

We account for the phenomenon of persons reproducing the same "tune" at different speeds and in different registers or keys. Few people have the ability to remember and reproduce exact pitches [17], an ability known as "absolute" or "perfect" pitch. As such, transpositional invariance is a desirable feature of any query/retrieval model. The effect of transposition is simply to add a certain value to all pitches. Consider for example the transposition illustrated in Figure 4(a) of *Trans* = +4.

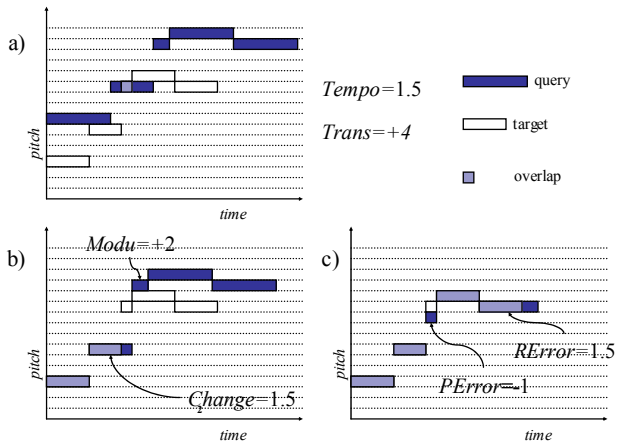

Figure 4: error class examples, opening notes of Brahms' "Cello Sonata in e-minor"

Tempo in this context is simply the translation of rhythm, which describes duration relationships, into actual time durations. Again, it is difficult to remember and reproduce an exact tempo. Moreover, it is very unlikely that two persons would choose the same metronome marking, much less unconstrained beat timing, for any piece of music. The effect of a tempo scaling is simply to multiply all IOI values by some amount. Thus, if the query is 50% faster than the target, we have a scaling value of *Tempo*=1.5, as shown in Figure 4(a).

In practice, we use quantized tempo scaling and duration values. Note that addition in the logarithmic scale is equivalent to multiplication, yielding a substantial computational advantage: this replaces floating point multiplication with integer addition. For instance, given our quantization bins, a doubling of tempo always corresponds to an addition of 4: $Tempo = 2 \leftrightarrow Tempo_{sym} = +4$.

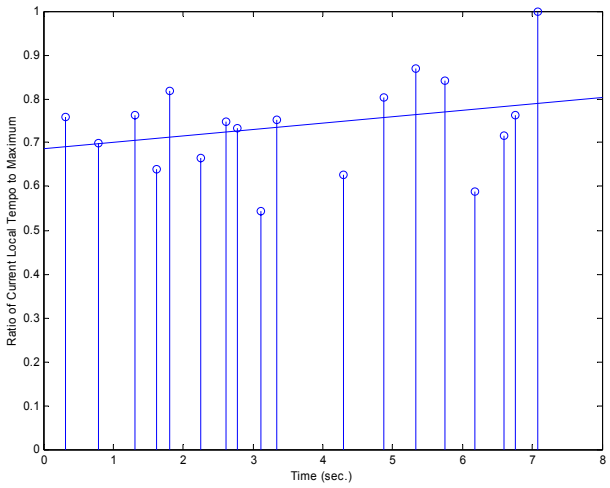

Figure 5: Tempo increase

3.3 Modulations and Tempo Changes

Throughout a query, the degree of transposition or tempo scaling can change, referred to as *modulations* and *tempo changes* respectively. Consider for a moment a query beginning with the identity transposition *Trans*=0 and identity tempo scaling *Tempo*=1, as in Figure 4(b). When a modulation or tempo change is introduced, it is always with respect to the previous transposition and tempo. For instance, on the third note of the example, a modulation of *Modu*=+2 occurs. For the remainder of the query, the transposition is then equal to 0+2=+2, since 0 is the starting reference transposition. Similarly, the tempo change of *Change*=1.5 on the second note means that all subsequent events occur at a tempo scaling of 1*1.5=1.5.

Consider Figure 5, which plots the apparent tempo scaling in a rendition of "Row, Row, Row your Boat" on a note-by-note basis. While our model considers several interpretations of such a rendition, one approach would be to consider a constantly increasing tempo, represented by the least-square deviation regression line, with local rhythmic errors (see Section 3.4), represented by the note-wise deviations from that line.

3.4 Local Pitch and IOI Errors

In addition to the "gross" errors we have discussed thus far, there are frequently local errors in pitch and rhythm. These errors are relative to the modifications described above. A local pitch error of *PError* simply adds some value to the "ideal" pitch. A local IOI error of *RError* has a scalar effect (or again, additive in the quantized domain.) Figure 4(c) shows examples of each. Note that these errors do *not* propagate to subsequent events, and as such as termed non-cumulative errors. We model cumulative errors as transpositions and tempo changes.

In some cases, there are multiple interpretations for the source of error in a query. Consider for instance Figure 9, which shows a

specific interpretation of three disagreements between a target and query. The second note in the query is treated as a local pitch error of -1. The final two notes, which are a semi-tone sharp, are handled as a modulation. The error model, described below, considers all possible interpretations simultaneously, for instance considering the possibility that the error in the second note is accounted for by two modulations (before and after), and the final two errors by a pair of local errors. Depending on our *expectation* that such errors will occur, one or the other interpretation might appear more likely.

4. ERROR MODEL

Hidden Markov models are the basis for our approach. For an excellent tutorial on these structures, please see Rabiner [14].

We account for edit errors in the query (insertions and deletions) in the "hidden" portion of the model. Using the notion of state "clusters," we account for transposition, modulation, tempo and tempo changes. Fine pitch and rhythm errors are accounted for in the observation distribution function.

4.1 Join and Elaborations

For the sake of notational clarity, we do not enumerate the states in the hidden model, but define them in terms of symbols that *indirectly refer* to events in the target sequence. There are three types of symbol:

- $Same_i$: refers to the correspondence between the i^{th} note in the target and an event in the query.
- $Join_i^l$: refers to a "join" of l notes, starting from the i^{th} note in the target sequence. In other words, a single note in the query replaces l notes in the target.
- $Elab_{i,j}^m$: refers to the j^{th} query note elaborating the i^{th} target note. In other words, a single note in the target is replaced by m notes in the query.

Notice that $Same_i = Join_i^1 = Elab_{i,1}^1$, each referring to a one-to-one correspondence between target and query. In our implementation, $Join_i^1$ plays all three roles. We generate a set of states S for a given target consisting of, for each target event:

- A *Same* state.
- Join states for $2 \le l \le L$, where L is some arbitrary limit on the number of events that can be joined.
- Elaboration states for $2 \le m \le M$ and $1 \le j \le m$, where M is some arbitrary limit on the length of elaborations.

Why do we have so many states to describe each event in the target? We wish to establish a one-to-one correspondence between hidden states and query events, to simplify the implementation, which is why we introduce multiple states for each elaboration. We choose not to implement joins by "skips" through a reduced set of states, since as discussed, joins influence not only which target events we consider, but how we interpret them.

4.2 Transition Matrix

We now describe the transition matrix A, which maps from $S \times S \to \Re$. Where q_t is the state at time t (as defined by position in the query note sequence), a_{xy} represents the probability $P(q_{t+1}=y \mid q_t=x)$, or in other words, the chances we will proceed from state x to state y.

Most of the transitions have zero probability, as suggested by the state descriptions. For instance, $Same_i$ states can only precede states pointing to index i+1 in the target. Elaboration states are even more restrictive, as they form deterministic chains of the form: $Elab_{i,1}^m \to Elab_{i,2}^m \to \ldots \to Elab_{i,m}^m$. This last state can then proceed, like $Same_i$, to the i+1 states. Similarly, $Join_i^l$ states can only proceed to $i+l$ states. A sample model topology is shown in Figure 6, for $M=L=2$. Note that this is a *left-right* model, in which transitions impose a partial ordering on states.

Based on properties of the target, we can generate these transition probabilities. We define $P_{Join}(i,l)$ as the probability that the i^{th} note in the target will be modified by an order l join. $P_{Elab}(i,m)$ is the probability that the i^{th} note in the target will be modified by an order m elaboration. $P_{Same}(i)$ has the expected meaning. Since every state has non-zero transitions to all states with a particular state, we must insure that:

$$\forall i, P_{Same}(i) + \sum_{m=2}^{M} P_{Elab}(i,m) + \sum_{l=2}^{L} P_{Join}(i,l) = 1$$

This also means that along non-zero transitions, the probability is entirely determined by the second state. For example, the probability of the transition $Join_3^2 \to Elab_{5,1}^2$ is the same as for $Same_4 \to Elab_{5,1}^2$.

4.2.1 Defining P_{Same}, P_{Elab} and P_{Join}

We intentionally leave P_{Same}, P_{Elab} and P_{Join} undefined. With reference to broadly observed trends in queries and their transcription, we suggest these alternatives:

1. The simplest and easiest solution is simply to build up tables indicating the chances that, *in general*, a note will be elaborated or joined. Thus, the probabilities are independent of the particular event in the target. For instance, our current test implementation uses this approach with $M=2$ and $L=2$, with $\forall i, P_{Same}(i) = 0.95$, $P_{Join}(i,2) = 0.03$, and $P_{Elab}(i,2) = 0.02$.
2. Transcribers are more likely to "miss" shorter notes, as are singers (consider for instance Figure 8, in which the second and third note are joined.) As such, we believe it will be possible to take advantage of contextual information (durations of surrounding events) to determine the likelihood of joins and elaborations at each point in the target sequence.

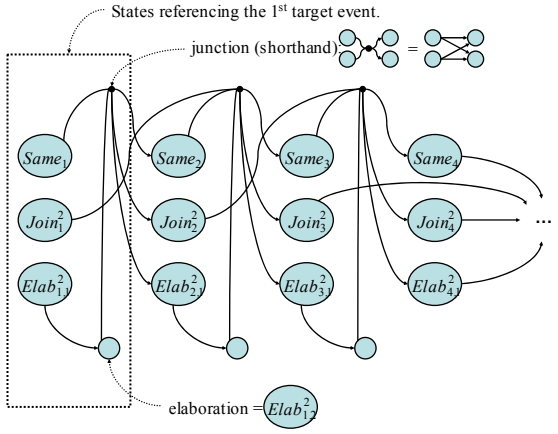

Figure 6: Hidden model topology

4.2.2 Distribution of initial states π_s.

We associate the initial state distribution in the hidden model with a single target event. As such, a separate model for each possible sequence starting point must be built. Note however that we can actually reference a single larger model, and generate different initial state distributions for each separate starting-point model,

addressing any concerns about the memory and time costs of building the models. Essentially, these various "derived" models correspond to various alignments of the query with the target sequence.

Our initial state distribution, for an alignment starting with the i^{th} event in the target, is therefore over the states $Same_i$, $Elab_{i,1}^m$ and $Join_i^l$, with probabilities determined by P_{Same}, P_{Elab} and P_{Join} respectively. For example, $\pi_s(Same_i) = P_{Same}(i)$ where $\pi_s(x) = P(q_1 = x)$, the probability of beginning in state x.

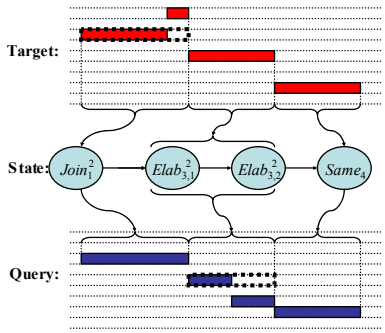

Figure 7: Relationship between states and events

4.2.3 Translating from State to Event

The hidden-layer states represent note events (in the case of joins) or sequences of notes (in the case of elaborations.) As mentioned, we treat only non-modulating and non-warping insertions and deletions. As such, when comparing target and query events with respect to a join, we generate a longer target note, with the sum duration of the relevant target events, and the pitch of the first. Similarly, for an elaboration, we consider a longer query event. Figure 7 shows a portion of the hidden state graph relating a target and query through a sequence of hidden states, where the dotted notes are examples of each modification.

Where $\langle Pitch_{query[i]}, IOI_{query[i]} \rangle$ is the i^{th} query note, and $\langle Pitch_{target[i]}, IOI_{target[i]} \rangle$ the i^{th} target note, we have if the following relationships between target and query, indicated by →:

$$\begin{cases} \langle Pitch_{target[i]}, IOI_{target[i]} \rangle \rightarrow \langle Pitch_{query[t]}, IOI_{query[t]} \rangle, & \text{if } q_t = Same_i \\ \langle Pitch_{target[i]}, \sum_{j=i}^{i+l-1} IOI_{target[j]} \rangle \rightarrow \langle Pitch_{query[t]}, IOI_{query[t]} \rangle, & \text{if } q_t = Join_i^l \\ \langle Pitch_{target[i]}, IOI_{target[i]} \rangle \rightarrow \langle Pitch_{query[t]}, \sum_{j=t}^{t+m-1} IOI_{query[j]} \rangle, & \text{if } q_{\{t,t+1,\ldots,t+m-1\}} = Elab_{i,\{1,2,\ldots,m\}}^m \end{cases}$$

4.3 Transposition and Tempo

In order to account for the various ways in which target and query could be related through transposition and tempo, we must refine our state definition. We use the notation $s' = \langle s, Trans, Tempo_{sym} \rangle$ to refer to a state with "type" s, and "cluster" $\langle Trans, Tempo_{sym} \rangle$.

The intuition here is that the type determines the gross relationship of the query with the target (see Figure 7, for instance) and the cluster determines *how* particular events are related between the target and the query.

Again, we establish limits on how far off target a query can be. Since we use a pitch-class representation, we can represent all possible transpositions in the range $-5 \le Trans \le +6$. We currently allow for tempi as slow as half speed, or as fast as double speed: $-4 \le Tempo_{sym} \le +4$, using the same quantization degree as for IOI_{sym}: $Tempo_{sym} = 4 \log_2 Tempo$. The initial distributions are defined as follows:

- $\pi_{Trans}(x) = P(q_1 = \langle ?, x, ? \rangle)$: the probability of beginning a query with $Trans = x$.
- $\pi_{Tempo}(x) = P(q_1 = \langle ?, ?, x \rangle)$: the probability of beginning a query with transposition $Tempo_{sym} = x$.

Note that we treat *Trans* and *Tempo* as conditionally independent in this model. Relaxing this assumption entails some additional computational complexity, and a substantial increase in the parameterization of the model, but is not *conceptually* a difficult modification.

We propose two approaches to the shaping of the initial distribution π_{Trans}:

1. Since the overwhelming majority of people do not have absolute pitch, we recommend a uniform initial distribution: $\pi_{Trans}(x) = \frac{1}{12}$.
2. The distribution could be tailored to individual users' abilities, thus the distributions might be quite different for a musician with absolute pitch and a typical user.

We propose a single tack for π_{Tempo}. We are able to remember roughly how fast a song "goes". As such, we currently apply a normal distribution[2] over initial tempo, with $\mu = 0, \sigma = 1.5$, again in the quantized tempo representation.

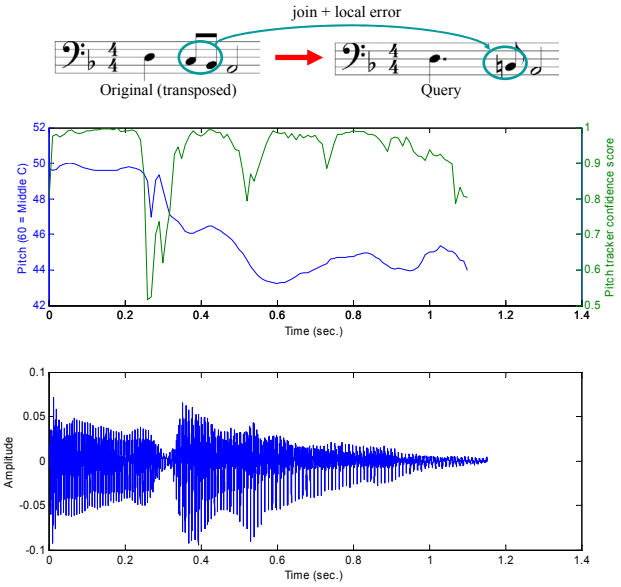

Figure 8: Portion of query on "Hey Jude", The Beatles[3]

[2] In our experiments, we frequently apply normal distributions over some probability function, using the normal density function: $y = \frac{e^{\frac{-(\mu - x)^2}{2\sigma^2}}}{\sigma\sqrt{\pi}}$, and then normalize to sum 1 over the function range.

[3] The Praat analysis shows two signals, one representing frequency, the other representing the *confidence* score in the analysis. Note segmentation was done manually for all examples in this paper and all experimental queries. The pitches indicated in the score are the weighted average (by confidence) of the frequencies observed over the course of each note. The notated rhythms are approximations, but timing was left unaltered for the purpose of the experiments.

The initial distribution over these refined states is thus: $\pi\left(\left\langle s, Trans, Tempo_{sym}\right\rangle\right) = \pi_s(s) \cdot \pi_{Trans}(Trans) \cdot \pi_{Tempo}(Tempo_{sym})$, the probability of beginning a query with state type s and cluster $\left\langle Trans, Tempo_{sym}\right\rangle$.

4.4 Modulation and Tempo Changes

Modulation and tempo changes are modelled as *transitions between clusters*. We denote the probability of modulating by x semitones on the i^{th} target event as $P_{Modu}(i, x)$ (again defined over the range $-5 \le x \le +6$). The probability of a tempo change of x quantization units is denoted $P_{Change}(i, x)$, allowing for a halving to doubling of tempo with $-4 \le x \le +4$.

Here are possible methods for the calculation of P_{Modu}:

1. In our current implementation, we simply a apply a normal distribution over P_{Modu} centred at $x=0$, assuming that it is most likely a singer will not modulate on every note. The distribution is fixed across all events.
2. We may wish to take advantage of some additional musical context. For instance, we have noted that singers are more likely to modulate for a large pitch interval.

We have observed no clear trend in tempo changes. Again, we simply define a normal distribution centred at $x=0$.

Given these definitions, we can now describe the transition table A' for the refined hidden states, incorporating cluster transition probabilities:

$$A': \left\langle S \times [-5,+6] \times [-4,+4]\right\rangle \times \left\langle S \times [-5,+6] \times [-4,+4]\right\rangle \to \Re$$

$$a'_{\left\langle x, Trans_x, Tempo_{sym_x}\right\rangle \left\langle y, Trans_y, Tempo_{sym_y}\right\rangle} =$$

$$a_{xy} \cdot P_{Modu}\left(i, \underbrace{Trans_y - Trans_x}_{\text{"Modulation" is the change in transposition}}\right) \cdot$$

$$P_{Change}\left(i, \underbrace{Tempo_{sym_y} - Tempo_{sym_x}}_{\text{Tempo change}}\right)$$

where i is the index in the target referenced by state y. Consider for instance the transition $\left\langle Join_1^2, +3, +1\right\rangle \to \left\langle Same_3, +5, +2\right\rangle$. This represents a whole-tone modulation of +2 and a tempo increase of +1, along with a transition to the $Same_3$ state type. The probability of the transition is therefore: $P_{Same}(3) \cdot P_{Modu}(3,+2) \cdot P_{Change}(3,+1)$.

4.5 Local Pitch and Duration Errors

These errors are modelled in the probabilistic mapping from hidden state to observation (or query note.) We define $P_{PError}(i, x)$ as the probability of a pitch error x on the i^{th} target event, and $P_{RError}(i, x)$ as the similar concept for IOI error, in quantization units.

Here are possible methods for determining P_{PError}:

1. In our current implementation, we simply apply a normal distribution, centred at $x=0$.
2. Downey and Nelson [19] notes certain tendencies in pitch error depending on pitch interval. We can thus take advantage of intervallic context in building pitch error distributions.

Again, we have not as yet determined a clear trend in rhythmic error. In our current implementation we apply a normal distribution over P_{RError} centred at 0, though we believe an examination of the relationship between duration and error deviation might be fruitful.

4.6 Generation of Queries Using Model

For experimentation, we generate synthetic queries, while varying model parameters. The process is straightforward:

1. Choose an initial state type, tempo scaling and transposition from the distributions π_s, π_{Trans} and π_{Tempo} respectively.
2. Generate a sequence of hidden states using the Markov transition matrix model A', of some predetermined length. For each hidden state, generate a query event using the local error distributions, P_{PError} and P_{RError}, and given the states' tempo and transposition values.

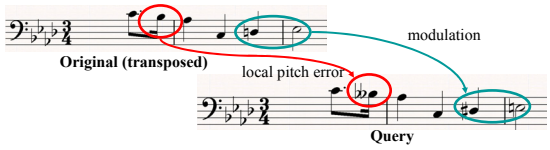

Figure 9: Portion of query from "American National Anthem", examples of modulation and local pitch error

4.7 Observation Probability

Combining all of these factors, we now describe the probability of an observation (query note) given a state $q_t = \left\langle s, Trans, Tempo_{sym}\right\rangle$.

Using the procedure described in Section 4.2.3, we generate a pair of events $\left\langle PC_{target}, IOIsym_{target}\right\rangle$ and $\left\langle PC_{query}, IOIsym_{query}\right\rangle$. Given an "ideal" mapping from state to query event, we would expect:

$$\left\langle PC_{ideal}, IOIsym_{ideal}\right\rangle = \left\langle PC_{target} + Trans, \underbrace{IOIsym_{target} + Tempo_{sym}}_{\text{These pairs are "compatible" because of the log scales used}}\right\rangle$$

Given these ideal values, we calculate the *local* error values:

$$PError = PC_{query} - PC_{ideal}, RError = IOIsym_{query} - IOIsym_{ideal}$$

The probability that q_t would generate the observation $\left\langle PC_{query}, IOIsym_{query}\right\rangle$ is then: $P_{PError}(i, PError) \cdot P_{RError}(i, RError)$.

We use the shorthand $b(q_t, o_t)$ to refer to the observation probability of the relevant query events o_t with respect to state q_t.

In the case of elaborations, a *deterministic* sequence of hidden states can refer to a sequence of query notes. Conceptually, we consider the observation probabilities of these sequences in their entirety. In practice, this means calculating the observation probability over a some segment.

5. PROBABILITY OF A QUERY

In music retrieval, we are primarily concerned with calculating the likelihood that a certain target would generate a query given the model. Using these likelihood values, we can then rank a series of potential database targets in terms of their relevance to the query.

Conceptually, the idea is to consider every possible path through the hidden model. Each path is represented by a sequence of states $Q=\{q_1, q_2, \ldots, q_T\}$, which has a probability equal to the product of the transition probabilities of each successive pair of states. In addition, there is a certain probability that each path will generate the observation sequence $O=\{o_1, o_2, \ldots, o_T\}$ (or query.) The probability of a query given the model (denoted λ) is:

$$P(O \mid \lambda) = \sum_{\text{all } Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

$$= \sum_{\text{all } \{q_1, q_2, \ldots, q_T\}} \left[\begin{array}{l} \pi(q_1) b(q_1, o_1) \cdot a'_{q_1 q_2} b(q_2, o_2) \cdot \\ a'_{q_2 q_3} b(q_3, o_3) \cdot \ldots \cdot a'_{q_{T-1} q_T} b(q_T, o_T) \end{array} \right]$$

Fortunately, there is considerable redundancy in the naïve computation of this value. Using the standard forward-variable

algorithm [14] provides a significant reduction in complexity. We define a forward variable:

$$\alpha_t(s') = P(o_1, o_2, \ldots, o_t \mid q_t = s', \lambda)$$

We initialize the forward variable using the initial state probabilities:

$$\alpha_1(s') = P(o_1 \mid q_1 = s', \lambda) = \pi(s') \cdot b(s', o_1)$$

By induction, we can then calculate successive values:

$$\alpha_{t+1}(s') = \sum_{x' \in S} \alpha_t(x') a'_{x's'} b(s', o_{t+1})$$

Finally, the total probability of the model generating the query is the sum of the probabilities of *ending* in each state:

$$P(O \mid \lambda) = \sum_{\text{all } s'} \alpha_T(s')$$

5.1 Time and Space Complexity

Based on the topology of the hidden model, and the above optimization, we can calculate the complexity of the forward-variable algorithm for this implementation. Since each state *type* has non-zero transition probabilities for at most $L+M$-1 other types, this defines a branching factor (b) for the forward algorithm. In addition, any model can have at most bn states, where n is the length of the target.

Updating the transposition and tempo probabilities between two state types (including all cluster permutations) requires $k = (9 \cdot 12)^2$ multiplications given the current tempo quantization, and the limits on tempo change. Notice that increasing either the allowable range for tempo fluctuation, or the resolution of the quantization, results in a super-linear increase in time requirements!

So, at each induction step (for t=1,2,…), we require at most knb^2 multiplications. As such, given a target of length n and a query of length T, the cost is $O(knb^2T)$. Clearly, controlling the branching factor (by limiting the degree of join and elaboration) is critical. k is a non-trivial scaling factor, so we recommend minimizing the number of quantization levels as far as possible without overly sacrificing retrieval performance.

6. SIMULATION RESULTS

This error model is intended to serve in the context of a music-information retrieval model. It is comprehensive in the sense that it expresses the full range of transformations observed in the pitch and IOI domains for queries. Its usefulness, however, lies in the ability to discriminate among various hypotheses about the source of a query. It has been shown that even a small number of errors can lead to (fatally) low discrimination between targets [16]. We contend that our subtler, probabilistic model of query errors can lead to greater precision in music searches, even when significant error is introduced. We used synthetically generated queries to demonstrate this claim, over a database of 100 classical/romantic themes taken from a musical thematic catalogue [2]. We set the model parameters as follows:

- We allow joins and elaborations up to order three, with fixed probabilities as described in Section 4.2.1.
- We apply normal distributions over each of the remaining parameters discussed in the paper, examining the effect of error variance for modulation, tempo change, local pitch error, and local IOI error.

Each parameter setting corresponds, roughly, to a level of singer ability. As we increase the λ-values, the distribution flattens, so that our synthetic singers become increasingly likely to introduce increasingly dramatic error to the query. For each of these "singers," we generated 30 queries according to the current model settings (and the procedure described in Section 4.6), based on randomly chosen database targets. The queries are limited to a length of 12 notes, to prevent "default" matches for longer queries: such queries might be feasible only against models with longer underlying targets. We then posed these queries to the model database, evaluating performance based on the likelihood rank of the correct model.

Each synthetic singer is associated with a particular cumulative-error profile, and a particular non-cumulative-error profile. The error distributions associated with these profiles are shown in Figure 10 and Figure 11, respectively. The probability of changing tempo is with respect to a "tempo change factor" where, for instance, 2.0 is a doubling of tempo between two note events. Similarly, rhythmic error is shown as a factor of the original IOI value.

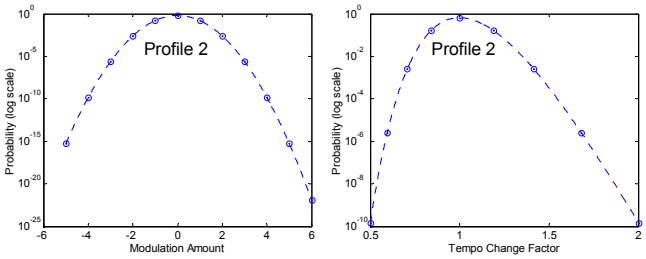

Figure 10: Cumulative Error Profiles (Profile 1 has no cumulative error)

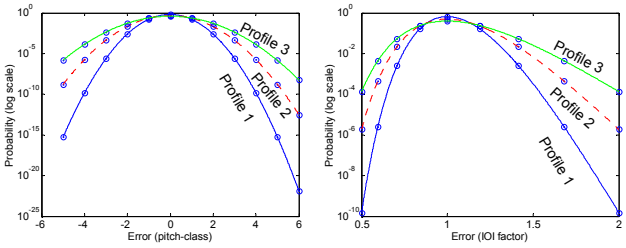

Figure 11: Non-Cumulative Error Profiles

Notice that the first cumulative-error profile allows for no error of this type, thus singers with this profile are assumed not to modulate or change tempo. For each singer profile, we indicate the number of queries for which the correct target is ranked first, ranked at least fifth, and ranked at least tenth. In addition, we indicate the Mean Reciprocal Rank (MRR), a standard measure used in the Text REtrieval Conference (TREC) benchmarks [21]. The "rank" in question is that of the highest rated relevant result. In these experiments, only one target is considered relevant to each query--the target generating the synthetic query--and, as a result, we simply take the reciprocal of that target's rank.

Cumulative Error Profile	Non-Cumu. Profile	# ranked first	# ranked <= 5th	# ranked <= 10th	MRR
1	1	30	30	30	1.0
1	2	30	30	30	1.0
1	3	29	30	30	0.983
2	1	29	30	30	0.983
2	2	29	30	30	0.975
2	3	27	28	30	0.920

Even when substantial error is introduced, discrimination remains robust. For instance, our final synthetic singer is more likely than not to introduce some local *and* cumulative error on every event of the query, but nonetheless the error model favours the correct target in 27 of 30 queries.

We define a database entry as "problematic" if it is either a false positive (ranked higher than the correct target) or a false negative (ranked lower than an incorrect target). In our experiments, we identified 20 such entries. In order to study the interactions among these entries in greater depth, we ran another experiment using only these problematic cases in our database. Using the most error-prone singer model (cumulative profile 2 and non-cumulative profile 3), we generated ten queries for each of these entries, and calculated the likelihood that each of the 20 problem models generated the query. The mean likelihoods of these query/target comparisons are shown in a confusion matrix (Figure 12). Probabilities are shown on a logarithmic scale, since there are orders of magnitude difference between values (an 'X' indicates that a particular comparison had the highest mean probability.) In addition, the probabilities for each query are normalized such that the highest ranked target has a probability of one.

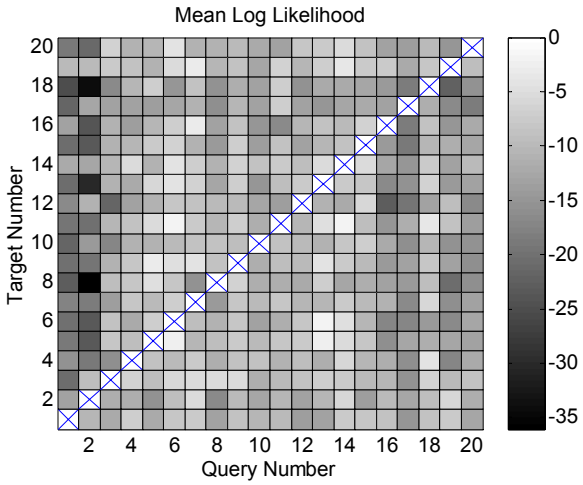

Figure 12: Confusion Matrix for Problematic Entries

The confusion matrix shows that, on average, the correct model is the most likely candidate for all queries, shown by the strong diagonal. This suggests that we need not expect uniformly poor discrimination for any particular target. More significantly, the model tends to favour correct targets over spurious matches by orders of magnitude (a factor of over 10^4 between probabilities on average). The MRR value for this set of queries was 0.938, though we must emphasize that this is across a smaller database.

7. EXPERIMENTAL RESULTS

To get anecdotal evidence on the ability of our model to characterize real singing tendencies, we ran some highly preliminary experiments on a small set of queries by five subjects, one a professional musician (subject A) and all others without any special musical training (subjects B-E). Each subject was asked to sing passages from four well-known songs, shown in Figure 13. Each passage was repeated twice from memory, and twice after hearing a piano rendition of the passage, for a total of 16 queries per subject. We augmented our thematic database with the four relevant targets for the purpose of these experiments. The sung queries were transcribed according to the process outlined in Figure 8, using manual note segmentation and automated pitch extraction.

In the absence of sufficient training data for the model, we simply used a liberal parameterization, using cumulative error profile 2 and non-cumulative error profile 3. We plan to extensively train our error model based on broad classifications of singer, to further improve performance. However, even without the benefit of training, we achieved solid performance on the test database for all subjects (MRR = 0.949), returning the correct target first for 75 of 80 queries.

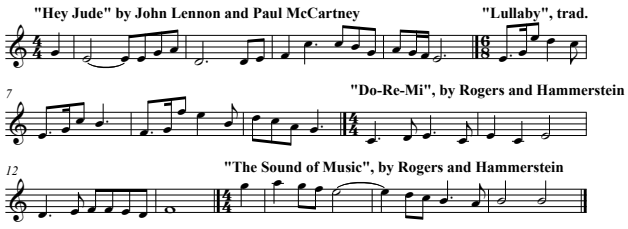

Figure 13: Queries

Table 1: Experimental Results

	Rank of Correct Target								
	Hey Jude		*Lullaby*		*Do-Re-Mi*		*Sound of…*		MRR
A	1	1	1	1	1	1	1	1	1.0
	1	1	1	1	1	1	1	1	
B	1	1	1	1	1	1	1	1	1.0
	1	1	1	1	1	1	1	1	
C	1	1	2	18	1	1	1	1	0.909
	1	1	1	1	1	1	1	1	
D	1	1	1	1	1	1	1	1	0.953
	1	1	1	1	1	1	4	1	
E	1	1	17	18	1	1	1	1	0.882
	1	1	1	1	1	1	1	1	
								Overall	0.949

1	1
1	1

← Before hearing passage

← After hearing passage

The "Lullaby" was problematic for Subjects C and E. Subject C was initially unable to remember the rhythm, though he had a rough recollection of the melodic contour. Subject E was unable to remember the melodic contour, though he convincingly reproduced the rhythm. After hearing the passage, both singers sang queries resulting in correct matches. Subject D's "Sound of Music" was occasion for the only other error in these experiments. Her third rendition was considerably slower than the version in our database.

With the real queries, we observe much lower probabilities on spurious matches, as well. In cases where we did have a correct match, the second most likely target was assigned a probability on average $1/10^{17}$ times the probability of the correct target. For this reason, we are optimistic about this model's ability to scale up to much larger databases.

8. FUTURE WORK

Even with the generalizations described in this model, a large number of parameters remain. We are currently gathering query data to train the model, as more in-depth evaluations of performance on non-synthetic queries will be essential. Various important questions remain to be answered, such as the following:

- What is the effect of query representation, for instance using a conventional note representation rather than pitch-class?
- How can we best tie parameters for training? For efficient training, how many equivalence classes can (or should) be established?
- HMMs are amenable to "frame-based" representations, which would allow us to bypass the note-segmentation stage of query transcription. Instead of modeling the query as a sequence of discrete note events, it is represented as a sequence of fixed-width time-frame analyses. Each state in the target model then has an associated distribution over duration – the probability of remaining in the state for some number of time-frames. We would like to explore the effectiveness of this

approach, particularly with regards to the tradeoffs between time and retrieval performance.

- We will shortly be integrating an automatic note-segmenter, currently being developed at the University of Michigan, which uses a simple neural-network to classify query analysis frames.

Finally, tests on much larger databases will be necessary. While we believe that meta-data in the query process (genre, era, instrumentation) will allow us to restrict searches to a subset of a database or library, it is reasonable to assume that a large number of targets will be relevant to many searches.

9. ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Foundation under grant IIS-0085945, and The University of Michigan College of Engineering seed grant to the MusEn project. 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 members of the MusEn research group for comments and advice. This group includes Greg Wakefield, Bryan Pardo, Norman Adams, and Mark Bartsch.

10. REFERENCES

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Journal of Pattern Analysis and Machine Intelligence*, 1983.

[2] H. Barlow and S. Morgenstern. *A Dictionary of Musical Themes*. Crown Publishers, 1948.

[3] M. Bartsch and G. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *WASPAA01*.

[4] W. Birmingham, B. Pardo, C. Meek, and J. Shifrin. The musart music-retrieval system. *D-Lib Magazine*, 2002.

[5] P. Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proceedings of the Institute of Phonetic Sciences.*

[6] A. Durey. Melody spotting using hidden Markov models. In *Proc. of International Symposium on MIR*, 2001.

[7] W. Birmingham et al. Musart: Music retrieval via aural queries. In *Proc. of International Symposium on MIR*, 2001.

[8] B. J. Feng. *The Structured Composite Source Representation.* PhD thesis, University of Michigan, November 2001.

[9] Stefan Kurtz. Foundations of sequence analysis. *citeseer.nj.nec.com/kurtz01foundations.html*, 2001.

[10] K. Lemstrom. String matching techniques for music retrieval. Technical report, University of Helsinki, 2000.

[11] D. Mazzoni. Melody matching directly from audio. In *Proc. of International Symposium on MIR*, 2001.

[12] B. Pardo and W. Birmingham. Automated partitioning of tonal music. In *Proc. of FLAIRS 2000.*

[13] E. Pollastri. An audio front end for query-by-humming systems. In *Proc. of International Symposium on MIR*, 2001.

[14] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of the IEEE*, 1992.

[15] J. Shifrin, B. Pardo, C. Meek and W. Birmingham. HMM-based musical query retrieval. In *Proc. of Joint Digital Libraries Conference*, 2002.

[16] T. Sorsa. Melodic resolution in music retrieval. In *Proc. of Symposium on MIR*, 2001.

[17] E. Terhardt and W. D. Ward. Recognition of musical key: Exploratory study. *J. Acoust. Soc. Am.*, 1982.

[18] E. M. Voorhees and D. Harman. Overview of the fifth text retrieval conference. In *The Fifth Text REtrieval Conference*, 1996.

[19] S. Downie and M. Nelson. Evaluation of a Simple and Effective Music Information Retrieval Method. *Proc. ACM-Sigir Conference, Athens, Greece.*

[20] Mongeau, M.,Sankoff,D., *Comparison of Musical Sequences*, Computers and the Humanities 24,Kluwer Academic Publishers 1990, 161-175.

[21] Text REtrieval Conference Web Site. *http://trec.nist.gov*.