HARMONIC MODELING FOR
POLYPHONIC MUSIC RETRIEVAL

A Dissertation Presented
by
JEREMY PICKENS
Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY
May 2004
Computer Science

Committee will be listed as:
W. Bruce Croft, Chair
James Allan, Member
Christopher Raphael, Member
Edwina Rissland, Member
Donald Byrd, Member
Department Chair will be listed as:
W. Bruce Croft, Department Chair

ACKNOWLEDGMENTS

It is incredible to me to realize that writing my doctoral dissertation is nearing an end. I arrived at graduate school not quite knowing what to expect from the entire research process. I am leaving with a profound understanding of how enjoyable that process is.

As I began my transition into graduate work, I was supported by the generous assistance of many fellow students currently in the program whom I thank, especially Warren Greiff and Lisa Ballesteros. As my work progressed, so did my collaborations and discussions. Essential among these have been evaluation methodology discussions with Dawn Lawrie and probabilistic modeling discussions with Victor Lavrenko.

In 1999 the Center for Intelligent Information Retrieval at UMass received an NSF Digital Libraries Phase II grant to begin work on music information retrieval systems. Donald Byrd invited me to be a part of this project, which led to this dissertation. I am grateful to him for extending this opportunity as well as for our numerous discussions and constructive arguments related to both text and music information retrieval matters. He is in many ways directly responsible for many of the directions this work took. Furthermore, all figures in this work that depict music in conventional notation format were generated by his Nightingale program; however, I assume full responsibility for any errors in the application of that notation.

The research team (OMRAS) formed in part by our grant included collaborators in the United Kingdom. From that team, Tim Crawford has been an invaluable support, co-formulating many of the ideas in this dissertation and helping fill the numerous gaps in my music education. In particular, the original idea for the harmonic description used as part of the harmonic modeling process was an idea that we both struck upon at the same time, but Tim was instrumental in fleshing out most of the important details. Matthew Dovey has also been a helpful sounding board and was instrumental in obtaining permission from Naxos to use portions of their audio collection as queries. Juan Pablo Bello, Giuliano Monti, Samer Abdallah, and Mark Sandler provided aid not only in terms of audio transcription, but in helping identify the problems we were trying to solve.

I thank my committee members for their many helpful comments, corrections, and suggestions, encouraging and pushing me to explore directions in which I otherwise might not have gone. Without the data from the Center for Computer Assisted Research in the Humanities, I would not have had a substantial portion of the current test collections, and my evaluation would have suffered. Therefore, I would like to thank Eleanor Selfridge-Field, David Huron, Bret Aarden, and Craig Sapp for not only providing this data but for assisting in the various issues that arose during format parsing and translation. Many others throughout my graduate tenure have been valuable in many ways, such as Kate Moruzzi and Sharon Mallory, and I cannot begin to list everyone.

Most importantly, I would like to thank my family. My mother, Melinda, has always made education a priority, and my father, John, has been an encouragement through his example and advice. My grandmothers Janet Rasmussen and Jean Tidd have always been there to support me, which has made this entire process that much easier. I would like to acknowledge my siblings, Ben and Sue, and the rest of my extended family as well; their love and encouragement have always been felt in my life.

ABSTRACT

Degrees will be listed as:
B.Sc. *cum laude*, BRIGHAM YOUNG UNIVERSITY
M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST
Directed by: Professor W. Bruce Croft

The content-based retrieval of Western music has received increasing attention in recent years. While much of this research deals with monophonic music, polyphonic music is far more common and more interesting, encompassing a wide selection of classical to popular music. Polyphony is also far more complex, with multiple overlapping notes per time step, in comparison with monophonic music's one-dimensional sequence of notes. Many of the techniques developed for monophonic music retrieval either break down or are simply not applicable to polyphony.

The first problem one encounters is that of vocabulary, or feature selection. How does one extract useful features from a polyphonic piece of music? The second problem is one of similarity. What is an effective method for determining the similarity or relevance of a music piece to a music query using the features that we have chosen? In this work we develop two approaches to solve these problems. The first approach, hidden Markov modeling, integrates feature extraction and probabilistic modeling into a single, formally sound framework. However, we feel these models tend to overfit the music pieces on which they were trained and, while useful, are limited in their effectiveness. Therefore, we develop a second approach, harmonic modeling, which decouples the feature extraction from the probabilistic sequence modeling. This allows us more control over the observable data and the aspects of it that are used for sequential probability estimation.

Our systems—the first of their kind—are able to not only retrieve real-world polyphonic music variations using polyphonic queries, but also bridge the audio-symbolic divide by using imperfectly-transcribed audio queries to retrieve error-free symbolic pieces of music at an extremely high precision rate. In support of this work we offer a comprehensive evaluation of our systems.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the short fictional story "Tlön, Uqbar, Orbis Tertius", author Jorge Luis Borges describes the inhabitants of the imaginary planet Tlön. In so doing, he describes a conception of the universe vastly different from our own. This conception stems from the language of these imaginary denizens.

> For the people of Tlön, the world is not an amalgam of *objects* in space; it is a heterogeneous series of independent *acts*–the world is successive, temporal, but not spatial. There are no nouns in the conjugal *Ursprache* of Tlön, from which its "present-day" languages and dialects derive: there are impersonal verbs, modified by monosyllabic suffixes (or prefixes) functioning as adverbs. For example, there is no noun that corresponds to our word "moon", but there is a verb which in English would be "to moonate" or "to enmoon". "The moon rose above the river" is "hlör u fang axaxaxas mlö", or, as Xul Solar succinctly translates: *Upward, behind the onstreaming it mooned* [19].

In this dissertation we begin with an understanding that the language of music is like the language of Tlön. In its purest form, music is composed exclusively of acts, not objects. Music is a doing and not a being. Any static feature one may extract destroys the fluid nature of the medium. Borges continues: "Every mental state is irreducible: the simple act of giving it a name – i.e., of classifying it – introduces a distortion, a slant or bias." Substituting "musical state" for "mental state" yields insight into the problem with which we are dealing. Along these same lines, Dannenberg [37] observes that "music evolves with every new composition. There can be no 'true' representation just as there can be no closed definition of music."

It would appear that any attempt at information retrieval for music is doomed from the outset. However, when one realizes that the goal of retrieval is not to create static, objective descriptions of music but to find pieces that contain patterns similar to a query, the limitations do not seem as overwhelming. Any proposed feature set will introduce slant or bias. However, if the bias is consistent, then the relative similarity of various music pieces to a given query will not change, and the retrieval process will not be hindered. It is the challenge and source of interest to this work to find features that are consistent in their slant as well as retrieval models that make apt use of such features, thereby effectively distinguishing among different pieces of music.

1.1 Information Retrieval

The fundamental problem of information retrieval is as follows: The user of a system has an information need, some knowledge that the user lacks and desires. The user has access to a collection of (most likely unstructured) information or data from which this information need can presumably be satisfied. The goal of the information retrieval system is to find some way of matching the information need with the information in the collection and extracting the pieces of information that are relevant to that need. Beyond attempting to satisfy the user's information need, having some manner of measuring the level of that satisfaction is also useful.

The information needs in this work are music information needs (see Section 1.2) and the type of information that comprises our collections is musical information (see Section 1.3). However, we must emphasize that the focus is on information retrieval rather than on music. Some music-theoretic techniques will be introduced, and of course the collection and queries themselves are music information. The goal is not to accurately or precisely model music; the goal is to satisfy a

user's information need. The emphasis is therefore not on the models, but on their ability to satisfy information needs.

We acknowledge that traditionally, information retrieval has meant text information retrieval. As alluded to in the introduction to this chapter, differences exist between text information and music information. We will explore these in the upcoming sections. Nevertheless, the fundamental goal is still to satisfy a user's information need.

1.1.1 Comparison with Text Retrieval

The purpose of this work is to bring music data into the information retrieval realm. In text information retrieval, a common view is that a document is relevant to a query if it is about the same thing that the query is about. Text documents on which retrieval systems operate are assumed to represent objective phenomena. As most retrieval systems are developed using newspaper articles, government or corporate reports, or Web pages, this assumption often holds true; the terms in such documents are high in semantic content. There are no poetry text retrieval systems, where authors can take poetic license with the meaning and usage of words and there can be little correlation between the syntax of a word and its semantic meaning. In the prose of the Web and of newspaper documents, words more often that not mean what they are.

This is an advantage of text retrieval systems that music does not have. Musical notes are not semantic-content bearing. Listeners do not hear a piece of music with the note $C\sharp$ in it and say, "ah, yes, this music is about C♯". On the other hand, readers do look at a document with the word "swimming" in it and say "ah, yes, this document is about swimming" or at least has something to say about swimming. A music piece with a C♯ does not really have a lot to say about C♯. It is perhaps a bit unfair to compare musical notes with text words. Notes are more akin to letters than they are to entire words. However, it remains unclear exactly how one should extract musical "words" from a piece.

In addition to the issue of semantic content, there is the problem of vocabulary size. A larger vocabulary has more raw discrimination power than does a smaller vocabulary. Text vocabularies are large, usually starting in the range of 40,000 terms or more. Music vocabularies are small, with around 128 available notes (on the MIDI scale), around half of which are never used in any given collection. At a very low level, text documents also have a very small vocabulary: 26 letters plus assorted punctuation for English. Because there are such natural, easily understandable, automated methods for moving from characters to words, most retrieval systems do not operate at the character level. Through the use of simple regular expressions, text data are easily transformed from raw characters into words bearing semantic content. In summary, text information is characterized by the following three factors: (1) A **Large vocabulary** of (2) **Easily extractable** and (3) **Semantic-content bearing** features.

Text information essentially has a nice *units of meaning* property (a high correlation between syntax and semantics). This does not immediately solve the text retrieval problem, but it makes it much easier than if these units of meaning were not present. Music does not have these units of meaning. Notes are a small vocabulary that does not bear semantic content, and there is no clear way of easily extracting units that do bear content.

Nevertheless, there is information to be retrieved, user information needs to be satisfied. We cannot rely only on the helpful fundamental units of meaning available to designers of text systems. Music information retrieval is not a research field distinct from text information retrieval—there is just an additional layer of complexity that results from this lack of semantic content.

1.1.2 Comparison with Other Forms of Retrieval

Music is not the only information source that suffers from the lack of a clear, easily extractable content-bearing terms. Pixels, the raw data that make up images, have a large vocabulary of millions of different color subshades, which is not content bearing. The same is true of video, a sequence of pixel maps over time. Raw audio, both the music and nonmusic kind, also suffers from this problem.

Biological information is another area which lacks readily available semantic content. Researchers are interested in mining or retrieving DNA sequences. Like music, DNA has an extremely small vocabulary: C, G, A, and T (cytosine, guanine, adenine, and thymine). Like music, this vocabulary does not bear significant semantic content. Just knowing that a particular DNA sequence contains cytosine in it is no evidence that the sequence is about cytosine.

It is interesting that some of the terminology used to describe music is also used to describe DNA sequences. For example, scientists speak of DNA motifs. "A DNA motif is a nucleic acid or amino acid sequence pattern that has, or is conjectured to have, some biological significance. Normally, the pattern is fairly short and is known to recur in different genes or several times within a gene" [53]. Significant passages of music often recur a number of times within a piece of music or across related movements; very short passages of this sort are even called motifs. Repetition is as important a concept for music as it is for genetics.

Scientists also speak about the need to find related variations in a genetic sequence. As will be explained later, finding variations is one of the fundamental goals of music retrieval. We do not claim that the techniques developed in this dissertation will solve DNA retrieval nor that they will solve text, image, or video retrieval. We only claim that the fields are related by the notion that patterns of information in a source collection that are similar to patterns of information in a user's information need might be good indicators of relevance to that need. For further explanation we turn to the concept of *evocativeness* and to the *language modeling* approach to information retrieval.

1.1.3 Evocative Framework

We propose that a useful framework for thinking about music retrieval is one that seeks less to discover true objective descriptions or semantic content of music sources and of music queries and more to discover how well a music source evokes a music query. In other words, it is useful to think of music information needs as having less to do with how much two pieces are about the same objective topic, less to do with whether one piece is relevant to another piece, and more to do with how evocative one piece is of another. The difference between these two conceptions is illustrated in Figure 1.1.

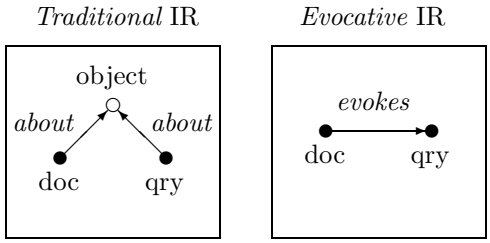


Figure 1.1. Distinction between traditional and evocative information retrieval (IR)

Evocativeness can no more be formally defined for music than *aboutness* can be for text. However, it is a useful concept to keep in mind when formulating feature selection techniques and retrieval models that incorporate those techniques, a concept that can guide and inspire the methods being researched.

1.1.4 The Language Modeling Approach

In recent years the language modeling approach to information retrieval has become quite popular [60, 92]. This novel framework uses techniques adapted from the speech recognition community:

> [A language model is] a probability distribution over strings in a finite alphabet [page 9]... The advantage of using language models is that observable information, i.e., the collection statistics, can be used in a principled way to estimate these models and do not

> have to be used in a heuristic fashion to estimate the probability of a process that nobody fully understands [page 10]...When the task is stated this way, the view of retrieval is that a model can capture the statistical regularities of text without inferring anything about the semantic content [page 15]." [92]

We adopt this approach for music. We assume that a piece of music d is generated by a model $p(d|M_D)$. The unknown parameter of this model is M_D. In Chapter 4 we use hidden Markov models to estimate $\widehat{M}_D$ from d, and in Chapter 5 we use smoothed partial observation vectors over d to estimate a visible or standard Markov model $\widehat{M}_D$ from d. In the latter approach, the smoothing is indeed heuristic, but it is done in a manner that makes principled use of the existing regularities. The statistics of the resulting smoothed vectors are still used to estimate the probabilities of a model without ever assuming anything about the semantic content of that music. We hope that by showing that these modeling approaches are applicable to music, we may bring music into the larger domain of information retrieval.

We mentioned in the previous section that evocativeness, like aboutness, is not defineable; however, we have a few possible interpretations for it. The first is query likelihood. A music document is said to evoke a music query if it is likely for the estimated model of that document to have generated that query. We take this approach in Chapter 4 and it was also taken by Ponte [92]. Another interpretation is model approximation, in the form of conditional relative entropy. A music document is said to evoke a music query if the model of that document closely approximates the model of that query. This approach was taken by Zhai [121]. In either case, crucial to the notion of evocativeness is the fact that we do not try to estimate aboutness or relevance, directly. Rather, probabilistic language models are developed that let the statistical regularities of music speak for itself.

1.2 Music Information Needs

For music information retrieval systems to be discussed and developed, a stable groundwork needs to be laid. An understanding of the nature of music information needs can guide the creation of feature sets and retrieval functions. This section explores what it means for a music piece to be relevant to a music query. These are not the actual queries we will use in our systems, especially as some of them are monophonic and we are trying to solve the more difficult polyphonic case. They are examples of the types of information needs users might have.

1.2.1 Known Item, or "Name That Tune"

The following email was posted to an online forum. It contains an example of a real world music information need [10]:

> Hi, music librarians! On another listserv (the Ampex pro audio one), a query has been circulating about locating the original attribution for the "snake charmer" melody–but to no avail. I would guess some sort of "oriental" Russian piece from the late nineteenth century, but can't quite put my finger on it. I can't imagine that Raymond Scott wrote it himself. Here's the original query:
>
> ID wanted: Snake dance, Snake Charmer, Hoochie Koochie, Hula-Hula Dance etc. There have been apparently many names for this piece over the years. Everyone has probably heard it in Warner Brothers or other cartoons, and on various old radio shows as a "gag" piece, but nobody has been able to identify it positively or suggest a composer. Names like Snake Dance, Snake Charmer, Hoochie Koochie, and Hula-Hula Dance have been suggested, but nothing can be found on these. It is possible that it is one of those "traditional" or "public domain" pieces that have been lost in time? [The notes are] D E F E D, D E F A E F D, F G A A Bb A G E, F G G A G F, D E F E D, D E F A E F D

The responses from other members of the list are as interesting as the query itself. One list member wrote "I know it as, 'I'm a persian cat. I'm a little persian cat.' " Another wrote "Wasn't

that tune used for the intro on Steve Martin's 'King Tut'?" Two more people remembered a slightly more risque version of the song: "They wear no pants in the Southern part of France". In these four responses, only one person actually remembered the title of a song in which the query was found, Steve Martin's "King Tut". The other three had no recollection of any title, but instead remembered the melodic content of the song itself in the form of various lyrics which accompanied the piece. Thus, one real world music information need is "name that tune". One would like to find a music piece solely from the content of that piece, rather than from the metadata.

Another example of a "name that tune" information need was posted to the Google Answers online forum [59]. In the post, the user asks:

> Where does the musical motif come from that is played by so many bell towers around the world, and why is it so widespread? E-c-d-g...g-d-e-c (where g is the lowest note and c, d, e represent the fourth, fifth, and sixth above.)

Another user answered this post with a short history of this tune, the "Westminster chimes". In this case, the user's information need was met by another user. A content-based music information retrieval system would have allowed the user to input their query as actual music (either through humming, keyboard playing, or conventional music notation). That query could then be used to find Web pages in which relevant music content (such as a MIDI or audio file) was embedded. Such Web pages would likely contain the information the user was seeking. Thus, the user's information need can be met through a search based on musical content.

1.2.2 Variations, or "Find Different Arrangements"

Imagine for a moment a parent driving a teenager to soccer practice, forced to listen to this teenager's favorite radio station. A song comes on, an awful remake of some classic from the parent's own youth. The parent gets frustrated because he or she cannot remember the name of the artist who originally performed or wrote the song. The parent would like to use the current radio version of the song as a query for finding the original version.

This information need is one in which the user is not looking for an exact known tune but for different versions or arrangements (variations) on that tune. Many remakes of old songs have the same overall feel of the original but may contain wildly varying notes and rhythms, almost none of which are found in the original. Improvisational jazz is an extreme example of this phenomenon, although it occurs in popular and classical music as well.

1.2.3 Influenced Item, or "Find Quotations and Allusions"

Common in music is the practice of quoting or referencing passages, patterns, and styles of other composers. For example, the 15th symphony by Shostakovich contains numerous allusions to Rossini's famous line from the William Tell Overture, the familiar Lone Ranger melody: "Bah dah dum, Bah dah dum, Bah dah dum dum dahm". Musicologists use many of the same terms as those who study literature: quotation, reference, allusion, paraphrase, and parody [20]. Users may be interested in finding pieces that contain allusions to, references to, or quotations from their query. A piece that contains allusions to a query should normally be judged relevant to that query.

1.2.4 Working Definition of Relevance

In the previous sections we gave some real-world examples of different types of music information needs. In this section we make explicit the meaning of relevance within the context of this work. All the above information need statements contained a common thread: Relevance is determined through patterns of pitch. If the focus of this work were monophonic music, we might name this "melodic similarity". However, as melodies are typically not polyphonic, "thematic similarity" might be more appropriate. Whatever we wish to call it, relevance is primarily defined through pitch rather than through other types of features such as rhythm or timbre. Stated in terms of evocativeness

(see Section 1.1.3), we are only interested in whether one piece evokes the same melody as another piece, rather than whether one piece evokes the same rhythm or the same timbral feeling.

To test this notion, we create two different types of query sets. The first type is a known item set. We have amassed a number of music pieces in parallel audio and symbolic formats (see Section 1.3). We want to be able to use a query provided in the audio format to retrieve the same piece of music in its symbolic format. Because we wish to work with pitch data, this involves transcribing the audio piece, which will certainly introduce a number of errors; such is the state of the art for polyphonic transcription. We will determine whether the imperfect transcription can still retrieve the known item symbolic piece. The symbolic piece is judged as relevant to its corresponding audio transcription.

The second type of query set builds on the variations, or "finding different arrangements" information need. In support of this, we have collected a number of different real-world composed variations of a few pieces of music. In one case a handful of composers interpreted a certain piece of music in 26 different arrangements. In another case we have 75 real variations on one particular polyphonic piece of music. If any one of these variations were to be used as a query, we would hope and expect that a good retrieval system should be able to find all of the other variations. All variations on a particular theme are judged as relevant to any one variation.

Taking this a step further, one can even think of an imperfect audio transcription as a variation on a piece of music. We have also created parallel audio and symbolic versions of all of our variations pieces. Thus, with an imperfect transcription of one variation as a query, all other variations on that particular piece are judged as relevant.

Though we mentioned it in the previous section, this work does not treat the problem of finding quotations or allusions. The level at which we are working with piece of music is on the order of the whole song, piece, or movement. (For example, symphonies are broken down into their various movements at the natural/composed boundaries. While the resulting pieces are smaller than the original full symphony, it is still not a passage-level representation.) An entire piece/movement of music from the source collection is judged either relevant or not relevant to an entire piece of music used as a query. Future work may address the issue of passage-level retrieval, and thus passage-level relevance.

1.3 Music Representation

1.3.1 Part I - Notation

Music representation lies along a spectrum. At the heart of the matter is the desire for the composer to get across to the listener those ideas that the composer is trying to share. As some sort of performance is necessary to communicate these ideas, the question arises as to how best to represent this performance. On one end of the spectrum music is represented as symbolic or score-level instructions on what and how to play. On the other end of the spectrum, music is represented as a digitized audio recording of actual sound waves.

1.3.1.1 Definitions

Audio is a complete expression of composer intention. What is meant by the composer (at least as interpreted by another human, a conductor or a performer) is unmistakable, as one can hear the actual performance. However, there is no explicit structure to this representation. Rhythmic forms, phrasal structures, key structures, tonal centers, and other information that might be useful for retrieval are not explicitly given and must be inferred. Even the pitch values and durations of the actual notes played are not explicitly given. Figure 1.2 is an example waveform from a digitized recording.

The other end of of the spectrum is conventional music notation, or CMN [21, 115]. The most familiar implementation of this representation is sheet music. Notes, rests, key signatures, times signatures, sharps and flats, ties, slurs, rhythmic information (tuplets, note durations), and many

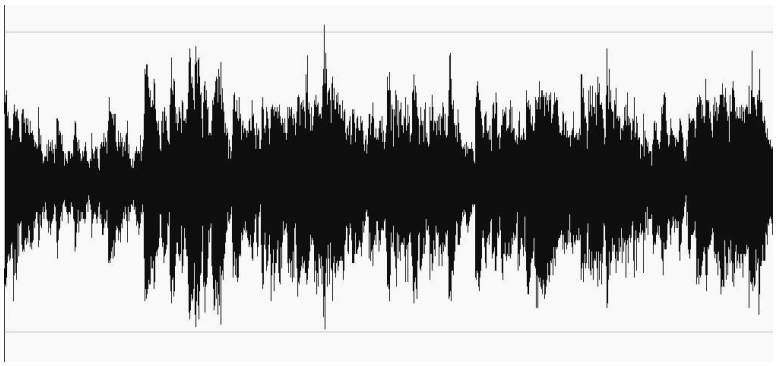


Figure 1.2. Bach Fugue #10, raw audio

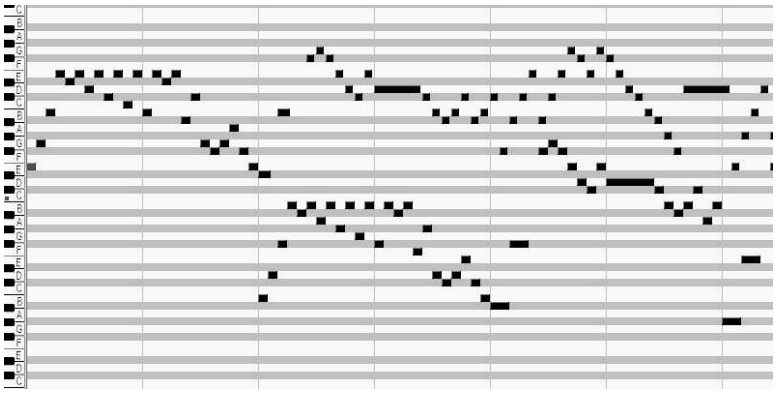


Figure 1.3. Bach Fugue #10, MIDI (event level)

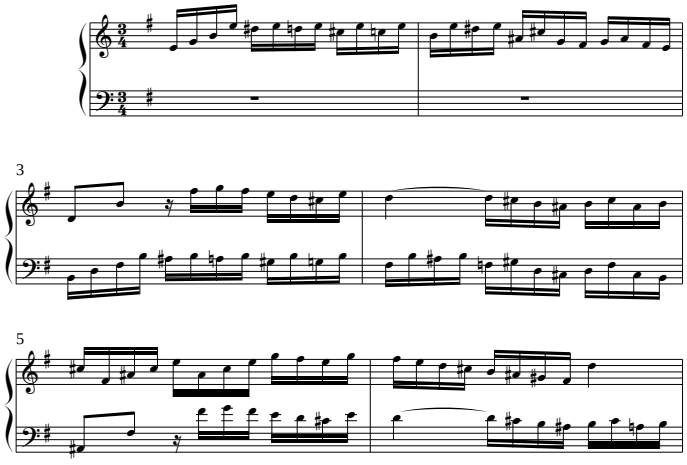


Figure 1.4. Bach Fugue #10, conventional music notation

more details are explicitly coded in files created by CMN notation software programs [3]. Figure 1.4 is an example of CMN.

Other representations lie somewhere between audio and CMN. Time-stamped MIDI is one of many event-level descriptors that holds the onset times and durations (in milliseconds) of all the notes in a piece of music. MIDI contains more structure than audio because the exact pitch and duration of every note is known. It contains less structure than CMN, however, because one cannot distinguish between an F♯ and a G♭; both have the same MIDI note number on a piano; both are the same pitch. It also cannot distinguish between a half note and two tied quarter notes. MIDI has often been compared with piano roll notation from player pianos of a century ago. Figure 1.3 is an example of event-level representation.

MIDI-like representations may further be broken down into two levels: score based and performance based. MIDI must be created somehow. The two most common ways are from a CMN score and conversion from a performance (either through a MIDI-enabled instrument or some form of audio note recognition). The difference between these two methods is subtle but important. A CMN-based MIDI piece is likely to have note durations that are perfect multiples of each other. For example, some notes might last for exactly 480 milliseconds, others for 240 milliseconds, and others for 960 milliseconds. One could therefore infer that certain notes were twice as long or half as long as other notes and use that knowledge for retrieval. However, if a MIDI file is created from a performance, notes might last for 483 milliseconds, or 272 milliseconds. This makes it difficult to tell, for example, whether the performer has played a half note, or a half note tied to a sixteenth note.

In summary, Figures 1.2 through 1.4 depict the gradual shift along the spectrum, from what the audience hears (audio) to what the performers do (MIDI) to instructions to the performers (conventional music notation). A helpful analogy, which likens music to language, is given by Byrd and Crawford [23]: audio music is like speech, event-level music is like unformatted text (such as straight ASCII), and CMN is like HTML- or XML-annotated text documents.

1.3.1.2 Conversion between formats

Conversion between representations for monophonic music (defined in Section 1.3.2) is a fairly well understood and solved problem. Conversion between representations for polyphonic music can be easy or extremely difficult depending on the direction of the conversion [23].

CMN to MIDI is accomplished by replacing symbolic pitch and duration information with number and time-based information. For example, a middle C quarter note could be replaced by MIDI note number 60 lasting for 480 milliseconds (depending on the tempo). Conversion from MIDI to audio is equally simple; a computer with a sound card can turn MIDI note number 60 on for 480 milliseconds and create an actual audio performance. There are even acoustic pianos that may be controlled via MIDI sequences. Such pianos will play all the notes in the piece (similar to a player piano) creating a true analog audio performance. Such a performance might not be the most emotive or expressive performance, but it is a true performance nonetheless.

Conversions in the opposite direction, from audio to MIDI or MIDI to CMN, are a much more difficult task. Audio music recognition— transformation from the performed score to a MIDI representation—is an unsolved open problem. Transformation from MIDI to CMN is considerably more manageable but still not an easy task [25, 97]. As mentioned above, MIDI cannot distinguish between an F♯ and *G♭* or between a half-note and two tied quarter notes. It cannot even tell whether a given note is a freestanding half note, quarter note, eighth note, or a member of some sort of tuplet. Conversions in the direction of audio toward CMN involve creating or deducing explicit structure where none is given in the source, and one can never be certain of the accuracy of this deduction.

1.3.2 Part II - Complexity

In addition to notation, another factor is important in describing or categorizing music representation: the number and type of simultaneous events that occur. This is referred to by musicians as *texture*. These are listed here in increasing order of complexity:

1. Monophonic
2. Homophonic
3. Voiced polyphonic
4. Unvoiced polyphonic

The following examples are presented with an excerpt from the J.S. Bach Chorale #49, *Ein' feste Burg ist Unser Gott*. The example in Figure 1.7 is Bach's original composition. The remaining examples are adapted from the original to illustrate the differences between the various textures.

1.3.2.1 Definitions

As seen in Figure 1.5, *monophonic* music has only one note sounding at any given time. No new note may begin until the current note finishes. With *homophonic* music, multiple simultaneous notes are allowed. However, all notes that begin at the same time must also end at the same time, and all notes that end at the same time must have also begun at the same time. Figure 1.6 shows that the number of each notes in each concurrent note onset may vary but that no notes in any set overlap with the notes in the next set.

Polyphonic music relaxes the strict requirement of homophonic music, allowing note sets to overlap. A note may begin before or concurrently with another note and end before, at the same time, or after that other note finishes sounding. There is no limit to the number or types of overlappings that may occur. However, a distinction needs to be drawn between voiced and unvoiced polyphonic music. In *voiced polyphonic* music, the music source is split into a number (two or more) voices. Each voice by itself is a monophonic (or sometimes homophonic) strand. Voices may be on the same instrument (on a piano, for example) or they may be played by different instruments (one voice played by the guitar, once voice played by the glockenspiel).

Unvoiced polyphonic music also contains multiple overlapping monophonic strands; however, they are unlabeled. It is inherently unclear which notes belong to which voice. Figure 1.7 shows a fully voiced excerpt from the Bach Chorale #49, while Figure 1.8 contains exactly the same information— the same note pitches and durations— with the voicing information removed or obscured.

1.3.2.2 Conversion between complexity levels

Conversion between monophony, homophony, and voiced and unvoiced polyphony is not as common as conversions between score and audio formats. In fact, conversion from lower complexity (monophony) to higher complexity (polyphony) is generally not perceived as an information retrieval task. Research does exist in the area, as shown by the HARMONET project [1], which attempts to create automatic, Bach chorale-style (homophonic) harmonizations of a monophonic sequence. We know of no information retrieval application of conversion to higher complexities.

Conversion from more complex to less complex music is an important and useful research area. Whether it is recovery of voicing information (conversion of unvoiced to voiced polyphony) or automatic melody extraction (conversion of polyphony or homophony to monophony), the reduction of more complex to less complex music has a solid place in information retrieval. Indeed, such conversions can be thought of as feature extraction techniques, and they will be explored in greater detail in Chapter 2.

1.3.3 Working Definition of Representation

The focus of this research is unvoiced polyphonic music in event-level form. The reason for this is threefold: polyphony is interesting, the vast majority of music is polyphonic, and most music available in event-level form cannot be guaranteed to be voiced. Sometimes it is fully voiced, sometimes it is partially voiced, but just as often it is completely unvoiced. Audio music recognition, or transcription, which is the process of transforming audio signals to MIDI or CMN, also produces music that is not voiced or is unreliably voiced at best.

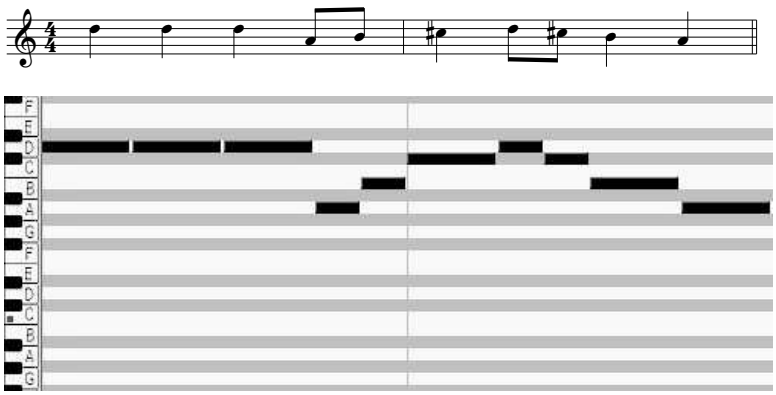


Figure 1.5. Bach Chorale #49, monophonic excerpt

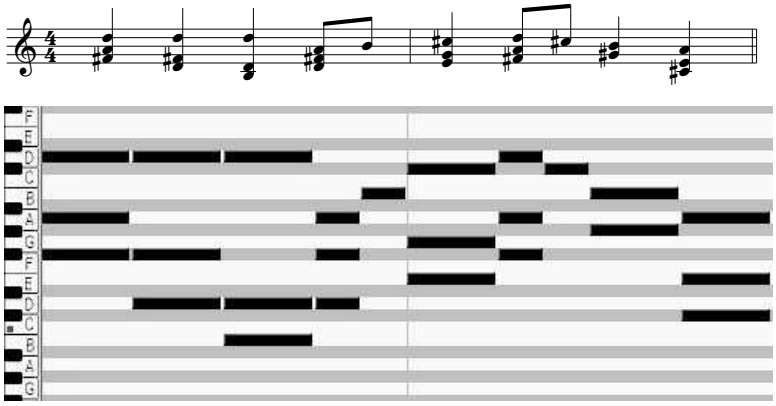


Figure 1.6. Bach Chorale #49, homophonic excerpt

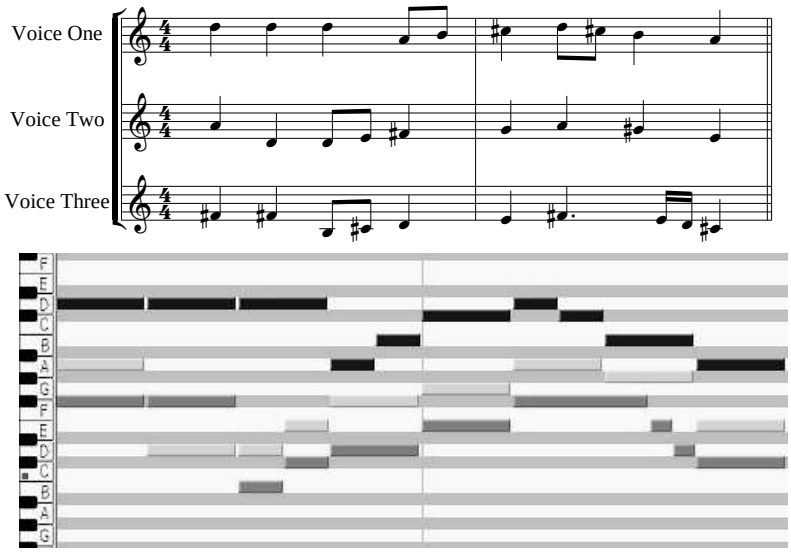


Figure 1.7. Bach Chorale #49, voiced polyphonic excerpt

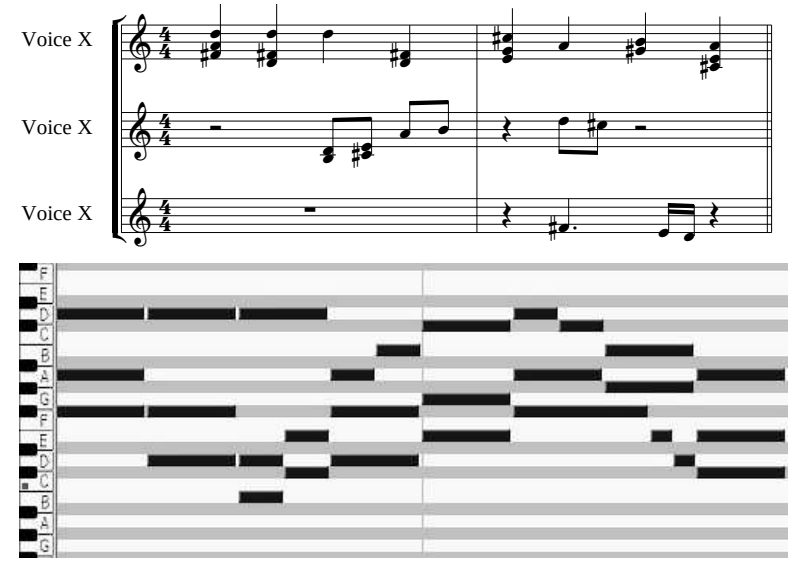


Figure 1.8. Bach Chorale #49, unvoiced polyphonic excerpt

Thus, it is important to develop techniques that work for unvoiced music, the lowest common denominator. If voicing information is available, that information may be used to further refine retrieval models or search results. Additional thoughts on the representation issues discussed here can be found in Byrd and Crawford [23].

1.4 Evaluation Framework

Evaluation of our music information retrieval systems will proceed much as does evaluation of other ad hoc text information retrieval systems. There are certainly many other important music information retrieval-related tasks, such as automated audio transcription, automatic clustering and heirarchy creation for user browsing, and so on. However, the focus of this work is on the ad hoc task, defined as new queries on a static (or nearly static) collection of documents. The collection is known a priori but the query that will be given is not. The Cranfield model is the standard evaluation paradigm for this sort of task and was outlined in the 1960s by Cleverdon et al. [31]. Along with many others in the music information retrieval community, we support this model for music information retrieval evaluation.

We undertake five basic steps to evaluate our systems: We (1) assemble collections of music pieces; (2) create queries on those collections; (3) make relevance judgements between queries and the pieces; (4) run retrieval experiments, using our models to created a ranked list of pieces; and (5) evaluate the effectiveness of each retrieval system by the qualtity of the ranked list it produces.

The first phase, assembling collections, is marked by a number of subtasks. Primary among these is defining a research format. Not all music notation formats are created equally, and various amounts of structure and information are found among the formats. Our research format, MEF (music event format), contains the bare minimum: only onset time, pitch, and millisecond duration of every note is known. The collections we will assemble are polyphonic. Voicing information may or may not be known, but it will be assumed to be unknown, and probabilistic modeling of documents will occur at that level. Our main source is the classical scores from the CCARH Musedata repository [54].

The second phase is assembling queries. It is assumed that the query will be given, translated, or transcribed into the same format as the collections: MEF. The onset time and pitch of every note in the query will be known, though the quality or accuracy of these note is not guaranteed and may vary depending on the source. Furthermore, the queries will be polyphonic, as are the documents in the collection. Queries are assembled by manually finding multiple versions or arrangements of a single piece of music. The third phase, creating relevance judgements, then becomes simple: when any one variation is used as a query, all variations on that piece are judged relevant and the remainder of the collection is judged nonrelevant. The last two phases, retrieval experiments and ranked list evaluation, can only be performed after retrieval systems have been built, which is the subject of the latter chapters of this work.

1.5 Significance of this Work

This work makes a number of contributions to the field of music information retrieval. First, this is the first fully polyphonic music retrieval system, meaning that both the query and the collection piece being sought are polyphonic. Second, and equally important, it is the first music retrieval system to bridge the audio/symbolic divide within the polyphonic realm. We will show that it is possible to use imperfect transcriptions of raw polyphonic audio to retrieve perfect transcriptions (original scores in symbolic notation) of that same piece. In addition to this song-identification application, we will also show that our methods are able to retrieve real world, composed variations on a piece of music. Our evaluation of our music retrieval systems is among the most comprehensive in the field to date.

1.6 Dissertation Outline

Chapter 2 contains an overview of the features and retrieval systems currently in use for music information retrieval. Chapter 3 contains a description of the features we have chosen to use, the intuitions behind choosing these features, and the data preparation necessary to be able to extract these features. Chapters 4 and 5 develop two retrieval systems based on our features. The former chapter covers a hidden Markov model approach while the latter chapter covers a decoupled two-stage Markov modeling approach. In Chapter 6 we comprehensively evaluate these systems, and in Chapter 7 we summarize the contributions of this work.

CHAPTER 2

RELATED WORK

The content-based retrieval of Western music has received increasing attention recently. Much of this research deals with monophonic music. Polyphonic music is far more common, almost to the point of ubiquity, but also more complex. Feature selection becomes a difficult task. Yet music information retrieval systems must extract viable features before they can define similarity measures. It is important to be aware that, throughout this dissertation, we deal exclusively with Western music, with its 12 pitches, octaves, and so on.

We wish to distinguish between feature selection techniques and full retrieval models. For text retrieval, a feature selection algorithm is often simply the regular expression rules used to convert a sequence of ASCII characters into a set of alphanumeric word tokens. A retrieval algorithm may be the different weights and matching functions used to pair query word tokens with document word tokens. With music, we also distinguish between features and the retrieval systems built using those features.

We emphasize the difference between feature extraction algorithms and retrieval algorithms for two important reasons. The first is that the number of viable feature extraction techniques is much larger for music than it is for text. Feature extraction is well-enough understood for text that it is almost considered a solved problem; most text researchers no longer even mention their word tokenization rules when describing their retrieval experiments. For music, on the other hand, features are still an open research area. The types of features extracted have great influence on the nature of the retrieval models built upon them.

The second reason for emphasizing the distinction is that in music retrieval, a single algorithm may have multiple distinct uses. An algorithm used for feature extraction by one set of researchers can be used by another set of researchers as an entire retrieval model. For example, Iliopoulos et al [57] uses string matching techniques to extract musical "words" from a music document, and researchers such as Downie [45] use these words as the basic features for a vector space retrieval model. On the other hand, Lemström [69] uses string matching as the entire basis for a retrieval algorithm; the strings being matched are the query strings. In both cases, string matching is being used, but in the first case it is to extract a set of features, and in the second case it is to find a query. We must be careful to distinguish between the tasks to which an algorithm is applied.

2.1 Feature Extraction

In this section, we summarize and categorize features that have been used for monophonic, homophonic, voiced polyphonic, and unvoiced polyphonic music. In all cases, some form of event-level representation is available to the feature extraction algorithms. As voiced polyphonic music is not always available (for example, in the case of raw audio), a common approach has been to reduce complex sources to simpler forms, then further extract viable features from these simpler forms. For example, Uitdenbogerd constructs what is assumed to be the most salient monophonic strand from a polyphonic piece, and then runs retrieval experiments on this monophonic strand [118, 119]. So while the focus of this work is unvoiced polyphony, a complete understanding of the features which may be extracted from less complex forms is necessary.

2.1.1 Monophonic Features

2.1.1.1 Absolute vs. Relative Measures

Most monophonic approaches to feature extraction use pitch and ignore duration; a few use duration and ignore pitch. Arguments may be made for the importance of absolute pitch or duration, but many music information retrieval researchers favor relative measures because a change in tempo (for duration features) or transposition for pitch features) does not significantly alter the music information expressed [44, 81, 49, 70, 16, 62, 107, 68], unless the transposition or the tempo change is very large.

Relative pitch is typically broken down into three levels: exact interval, rough contour, and simple contour. *Exact interval* is the signed magnitude between two contiguous pitches. *Simple contour* keeps the sign and discards the magnitude. *Rough contour* keeps the sign and groups the magnitude into a number of equivalence classes. For example, the intervals 1-3, 4-7, and 8-above become the classes "a little", "a fair amount", and "a lot". Relative duration has three similar standards: *exact ratio*, *rough contour*, and *simple contour*. The primary difference between pitch and duration is that duration invariance is obtained through proportion, rather than interval. Contours assume values of "faster" or "slower" rather than "higher" or "lower". In all above-mentioned relative features, intervals of "0" and ratios of "1" indicate no change from previous to current note. In information retrieval terms, using exact intervals and ratios aid precision, while contour aids recall. Rough contours or equivalence classes attempt to balance the two, gaining some flexibility in recall without sacrificing too much precision.

There are exceptions to the trend to treat pitch and duration as independent features [68, 28, 40]. In these approaches, pitch and duration (or pitch interval and duration ratio) are combined into a single value. By so doing, precision is increased; pitch combined with duration more clearly and uniquely identifies every tune in a collection. However, a great deal of flexibility, and thus recall, is sacrificed. When pitch and duration are combined into a single value, it is no longer possible to search on either feature separately, as might be desirable when a user is looking for different rhythmic interpretations of a single tune.

It is our feeling that pitch and duration should be extracted independently and then combined at the retrieval stage. While pitch and duration are generally not statistically independent, treating them as such in an information retrieval setting makes sense.

2.1.1.2 N-grams

Collectively, the features in section 2.1.1.1 are known as *unigrams*. A single pitch, pitch interval, duration, or duration ratio, is extracted. Some retrieval methods, such as string matching, require unigrams in order to function. But other approaches require larger basic features. Longer sequences, or *n-grams*, are constructed from an initial sequence of pitch, duration, pitch interval or duration ratio unigrams.

One of the simpler approaches to n-gram extraction is with sliding windows [45, 18, 119]. The sequence of notes within a length n window is converted to an n-gram. The n-gram may be of any type discussed in Section 2.1.1.1: absolute or relative values, exact intervals, rough contour intervals, or simple contour intervals. Numerous authors suggest a tradeoff between n-gram type and n-gram size. When absolute values or exact intervals are used, n-grams remain shorter, perhaps to avoid sacrificing recall. When rough or simple contour is used, n-grams become longer, perhaps to avoid sacrificing precision.

A more sophisticated approach to n-gram extraction is the detection of repeating patterns [52, 116, 71, 5]. Implicit in these approaches is the assumption that frequency or repetition plays a large role in characterizing a piece of music. The n-grams which are extracted are ones which appear two or more times in a piece of music. Sequences which do not repeat are ignored.

Another alternative segments a melody into musically relevant passages, or "musical surfaces" [78]. Weights are assigned to every potential boundary location, expressed in terms of relationships among pitch intervals, duration ratios, and explicit rests (where they exist). The weights are then

evaluated, and automatic decisions are made about where to place boundary markers using local maxima. The sequence of notes between markers becomes the n-gram window.

One last approach uses string matching techniques to detect and extract n-grams [6, 57]. Notions such as insertion, deletion, and subsitution are used to automatically detect n-grams. These n-grams, unlike those from other techniques, may be composed of notes which are *not always contiguous* within the original source; this is useful because the technique of *ornamentation*, common in almost all types of music, adds less important notes – often several at a time – between existing note pairs [4].

2.1.1.3 Shallow Structural Features

Features which are extracted using techniques which range from lightweight computational to lightweight music-theoretic analyses are given the name *shallow structural*. An example of such a feature for text information retrieval is a part-of-speech tagger [120], which identified words as nouns, verbs, adjectives, and so on. While music does not have parts of speech, it has somewhat analogous shallow structural concepts such as key or chord. A sequence of pitches is thus recast as a sequence of keys, tone centers, or chords. There are a growing number of techniques which examine a monophonic sequence of note pitches to do probabilistic best fit into a known key or chord [113, 112, 66].

Similar shallow structural techniques may be defined for duration as well as pitch. Shmulevich [113] describes techniques for defining the temporal pattern complexity of a sequence of durations. These methods may be applied to an entire piece, or to subsequences within a piece. A monophonic sequence of durations could be restructured as a monophonic sequence of rhythm complexity values.

2.1.1.4 Statistical Features

Statistical features may also be used to aid the monophonic music retrieval process. We distinguish between a pitch interval as a feature, and the statistical measure of pitch intervals. Extraction of the latter depends on the identification of the former, while retrieval systems which use the former do not necessarily use the latter. Schaffrath [105] creates an "interval repertoire", which includes the *relative frequencies* of various pitch unigrams, length of the source, and "tendency" of the melody (i.e.: 3% descending or 6% ascending). Mentioned, but not described, is a "duration repertoire" similar to the interval repertoire, giving counts and relative frequencies of duration ratios and contours. Other researchers do statistical analyses of sequential features [45]. It is clearly possible to subject most if not all of the features described in the preceeding sections to statistical analysis.

2.1.2 Homophonic Features

As with monophonic music, features most researchers select from homophonic music tend to ignore duration and extract pitch, or ignore pitch and extract duration. In Chapter 1 we characterized homophony as two-dimensional. This is only true for pitch features, however. The onset and duration sequence of a homophonic piece is one-dimensional. All of the notes in a given simultaneity, in a given time step, have the same duration. So there is a clear rhythmic or durational sequence, and monophonic rhythm feature selection techniques may be used for homophonic duration.

The pitch sequence, on the other hand, is more complicated. Rather than a sequence of pitches, homophonic music is a sequence of variable-sized pitch sets. Lemström et al [69] proposes a number of features based on these pitch sets. One approach uses octave equivalence to reduce the size of the pitch set from 128 (a full range of notes) to 12. Another approach attempts to mimic the relative measures discussed in Section 2.1.1.1, creating transposition invariance by transforming the sequence of pitch sets ($S = S_1 S_2 \ldots S_n$) into a sequence of pitch interval sets ($D = D_1 D_2 \ldots D_{n-1}$):

1 **for** $i := 2$ *to* n **do**
2 **for each** $a \in S_{i-1}$ *and* $b \in S_i$ **do**
3 $D_{i-1} := D_{i-1} \bigcup \{b - a\}$

We also note that harmonic analysis may be performed on homophonic music, but the techniques used are going to be practically identical to those used for polyphonic music. Therefore, we reserve discussion of harmonic analysis and harmonic descriptions for Section 2.1.4.4.

2.1.3 Voiced Polyphonic Features

Voiced polyphony presents a unique challenge to feature extraction. One must make the *a priori* assumption that the salient or relevant musical patterns, on which one believes a user will query, occur either in completely independent voices, or else jump from voice to voice. In other words, one must decide whether queries will cross voices or not.

If one believes that queries will not cross voices, then each voice can be treated independently, and existing monophonic techniques can be used to dissect each voice. It is still up to a retrieval model to decide how to treat the multiple voices, i.e., whether all voices are weighted equally and, if not, how to weight them. [118, 119]. However, this is not a problem that needs to be solved at the feature extraction stage.

If one believes that queries will cross voices, then some sort of feature which marks the relationship between voices at every time step needs to be created. We feel that, at the current time, the easiest way (though perhaps not the best way) to do this is simply to throw away voicing information in the music source and treat it as unvoiced polyphonic music. It is difficult to know, *a priori*, at which points and in which voices a user query might cross voices. As far as we know, no researchers have developed feature extraction techniques specifically designed for voiced polyphony, though Byrd and Crawford do discuss the cross-voice matching issue at length [23]. Voiced polyphonic music has either been treated as separate monophonic strands, or has been converted to unvoiced polyphonic music and subjected to the corresponding feature extraction techniques.

2.1.4 Unvoiced Polyphonic Features

Unvoiced polyphony is a large step in complexity beyond monophony and homophony. With monophony, there is sequentiality of both pitch and duration. Homophony has sequentiality of duration. With unvoiced polyphony, it is difficult to speak of the "next" note in a sequence; there is no clear one-dimensional sequence. Features such as pitch interval and duration contour are no longer viable. Most researchers avoid this complexity altogether by reducing unvoiced polyphonic music to simpler forms, then extracting additional features from those forms. This reduction destroys much of the information in a piece of music. Nevertheless, it is assumed that effective retrieval may still be done.

2.1.4.1 Reduction to Monophony

Perhaps the oldest approach to polyphonic feature selection is what we call *monophonic reduction*. A monophonic sequence is constructed from an unvoiced polyphonic source by selecting at most one note at every (non-overlapping) time step. The monophonic sequences that most researchers try to extract is the melody, or theme. Whether this monophonic sequence is useful for retrieval is tied to how well a technique extracts the "correct" melody, how well any monophonic sequence can actually represent a polyphonic source, and whether a user querying a music collection has the melody in mind.

The first thematic catalogues of this kind come from the 18th century, but the short sequences in Barlow and Morgenstern [7, 8] are probably the best-known use of monophonic reduction. They construct a short, "word-length" monophonic sequence of note pitches from a polyphonic source. (To be precise, there are a few instances where the extracted sequence is polyphonic; however, these are rare. For more discussion on these books, see Byrd [22].) The monophonic selection is done manually. Clearly, this becomes impractical as music collections grow large.

Automated methods become necessary. There exist algorithms which can search polyphonic sources for straight or evolutionary monophonic strings [69, 56]. There also exist feature extraction algorithms which automatically select salient monophonic patterns from monophonic sources using clues such as repetition or evolution (see section 2.1.1.2). Recently, researchers such as Meredith, Lemström and Wiggins [38] and Lavrenko and Pickens [64] combine the two, automatically selecting short, salient "word" strings from polyphonic sources.

One might not trust the intuition that repetition and evolution yield salient, short monophonic sequences that would be useful for retrieval. The alternative is to pull out an entire monophonic

note sequence equal to the length of the polyphonic source. Once this sequence is obtained, it may be further dissected and searched using available techniques from section 2.1.1. A naive approach is described in which the note with the highest pitch at any given time step is extracted [118, 119, 93]. An equally naive approach suggests using the note with the lowest pitch [16]. Other approaches use voice or channel information (when available), average pitch, and entropy measures to wind their way through a source [118]. Interestingly, the simple, highest pitch approach yields better results than the others.

2.1.4.2 Reduction to Homophony

While monophonic reduction is done by taking at most one note per time step, *homophonic reduction* is done by taking at most one *set* of notes per time step. Many different names have been given to sets created in this manner: simultaneities, windows, syncs, and chunks.

Homophonic sets differ slightly in the manner of their construction. Some approaches use only notes with simultaneous attack time, i.e.: if note X is still playing at the time that note Y begins, only Y belongs to the set [43]. Other approaches use all notes currently sounding, i.e.: if note X is still playing at the time that note Y begins, both X and Y belong to the set [69]. Yet other approaches use larger, time or rhythm based windows in which all the notes within that window belong to the set [93, 30]. In any case, once the unvoiced polyphonic source is reduced to a homophonic sequence of note sets, the feature extraction methods described in Section 2.1.2 are then applied. These include, among others, pitch interval sets and harmonic analysis.

2.1.4.3 Reduction to Voiced Polyphony

Some feature extraction techniques do not attempt to reduce unvoiced polyphony to either a single monophonic melodic line or a homophonic note set sequence. Instead, they split the unvoiced source into a number of monophonic sequences [75, 27]. This resulting set of monophonic sequences is equivalent to voiced polyphonic music, and may be treated as such. Whether any or all of the monophonic sequences created in this manner correspond to the "correct" voicing information (if any) is not as important as whether these voices are useful for retrieval. Currently, we know of no retrieval experiments which actually test features extracted in this manner.

2.1.4.4 Shallow Structural Features

As with monophonic music, features which are extracted using techniques which range from lightweight computational to lightweight music-theoretic analyses are given the name *shallow structural*. While it might be argued that harmony itself is not a shallow feature, as music theorists have been working on developing precise and intricate rules for harmonic analysis for hundreds of years, we wish to distinguish between the full use versus the superficial application of those rules. For example, a part-of-speech tagger for text does not need to do a full grammatical parse of an entire document (deep structure) in order to figure out whether a particular word is a noun or a verb. Instead, lightweight techniques (shallow structure) can be used to do this. By analogy, the same is possible for music.

There are undoubtedly dozens of papers and works on the harmonic analysis and harmonic description problem. In this section we mention just a few of those that are known to us and that are most germane to this dissertation. For example, Prather [93] segments a polyphonic sequence into windows based on a primary beat pattern (obtained using time signature and measure information). The pitches in these windows are made octave equivalent (mod 12), then further tempered by placing them into an atomic harmonic class, or *chord*. These harmonic classes are comprised of triads (major, minor, augmented, and diminished) and seventh chords (major, minor, dominant, and diminished minor) for every scale tone. The pitches in a set often fit more than one class, so neighboring sets are used to disambiguate potential candidates, leaving only a single chord per window.

Chou [29] also tempers pitch sets by their harmonicity. Sets are constructed by dividing a piece into measures and adding to each set all the notes present in a measure. A *chord decision algorithm* is then used to extract the most "salient" chord in that measure, and this chord is used for retrieval.

Five principles guide the selection of this chord, including a preference of chords with high frequency of root notes, fifths, and thirds. In other words, the frequency of consonant notes in the set contribute to the selection of a single most-salient chord.

Other researchers have focused on the chord extraction process as well. Barthelemy [9] starts by merging neighboring simultaneities which are highly similar, then assigns a single lexical chord label to each resulting merged simultaneity by mapping it to the nearest chord. Pardo [85] reverses the process: Instead of fitting simultaneities to lexical chords, the lexical chord set is used to dynamically shape the size of the simultaneities, so that partitioned areas are created in positions where a single (harmonically significant) lexical chord dominates. Pardo [86] also tackles the difficult problem of simultaneous segmentation and labeling. For example, if a triad is arpeggiated, then there should not be three separate windows for each of the note onsets. Those three onsets should be grouped into a single window, and labeled with the proper chord name. Similarly, other locations with richer (or non-arpeggiated) chordal textures would require smaller windows. Most other work in this area, ours included, has not specifically addressed the segmentation problem.

All the techniques listed above produce a one-dimensional sequence of atomic chord units. In other words, the goal is to do a reduction to the best chord label for each window. Other authors, ourselves included, have taken the approach that more than one chord may describe a window or chunk of music data [94, 113]. Purwins in particular uses Krumhansl distance metrics to assist in the scoring. In fact, the idea of multiple descriptors for a chunk of music was a fundamental aspect of Krumhansl's work; she mentions that her "...present algorithm produces a vector of quantitative values...thus, the algorithm produces a result that is consistent with the idea that at any point in time a listener may entertain multiple key hypotheses. (pages 77-78) [63]" It is with this same basis or understanding that we construct our own harmonic description algorithm in Chapter 5.

Recently, a few authors have taken a more principled, statistical approach to the problem of entertaining multiple key hypotheses. Ponsford uses a mixture of rules and Markov models to learn harmonic movement [91]. Raphael and Sheh use hidden Markov models and their associated learning algorithms to automatically induce from observable data the most likely chord sequences [100, 109]. Hidden Markov models are also a framework where the segmentation problem is given a principled probabilistic foundation. In Chapter 4 we also take the hidden Markov model approach to harmonic analysis.

2.1.4.5 Statistical Features

Blackburn [17] proposes a number of statistical features appropriate for polyphonic music: the number of notes per second, the number of chords per second, the pitch of notes (lowest, highest, mean average), the number of pitch classes used, pitch class entropy, the duration of notes (lowest, highest, mean average), number of semitones between notes (lowest, highest, mean average), how polyphonic a source is, and how repetitive a source is.

Many of these features are applicable to homophonic music as well. Using the average pitch in each time step might provide a decent measure of pitch contour (determined by looking at the difference between contigous average pitches). Using the average duration in each time step might do the same for duration contour. Using the number of notes per time step could yield a "busy-ness contour". Existing work is just beginning to enumerate the possibilities.

2.1.5 Deep Structural Features

A *deep* structural feature is the name we give more complex music-theoretic, artificial intelligent, or other form of symbolic cognitive techniques for feature extraction. Such research constructs its features with the goal of conceptually "understanding" or explaining music phenomena. For information retrieval, we are not interested in explanation so much as we are in comparison or similarity. Any technique which produces features that aid the retrieval process is useful. Unfortunately, most deep structural techniques are not fully automated; the theories presented must inspire rather than solve our feature extraction problems. These include Schenkerian analysis [106], AI techniques [26],

Chomskian grammars [102], and other structural representations [84], to name very few. Deeper structural features are beyond the scope of this work.

2.2 Retrieval Systems and Techniques

It was necessary to complete a review of existing feature extraction techniques before turning our attention to the retrieval systems which make use of the various features. Not every retrieval model is suited to every type of feature, and the type of feature used influences the nature of the retrieval model which may be constructed. For example, a string-matching retrieval approach would not work well when n-grams are the atomic unit, becaues string matching requires unigrams.

Though the focus of this work is polyphonic music, we again intersperse our discussion with references to monophonic approaches. Not all techniques developed for monophony are "scalable" to homophony or polyphony, but any discussion of music information retrieval should include both. At the time this work was begun, there were not that many systems which used polyphonic queries to search polyphonic source collections [41, 40, 79]. One of the contributions of this work is to add a stable foundation to the growing body of polyphonic symbol-based music retrieval research.

2.2.1 String Matching

The earliest example of a string matching retrieval algorithm comes from the Barlow and Morgenstern [7] melody index. An excerpt from the book is found in Figure 2.1.

G C D E G E G	L64
G C D E G F E D C	B1663
G C D E G F E D E	S1377
G C D E G G F	O8
G C D E G G G E	D7
G C D E G G G F	M63
G C D E♭ A♭ B♭	R273
G C D E♭ B C A♭	H346
G C D E♭ B C D	B46
G C D E♭ C A♭	B1454
G C D E♭ C B	B1524
G C D E♭ C D B♭	M194
G C D E♭ C D E♭	S474
G C D E♭ C G A♭	H153
G C D E♭ C G C D E♭ C	D369
G C D E♭ C G C D E♭ C	V80
G C D E♭ C G E♭	S601
G C D E♭ D C B A	V22

Figure 2.1. Excerpt from the Barlow and Morgenstern Notation Index

Retrieval is done in the following manner: A user formulates a query inside his own head, transposes that query into the key of C, and then selects a chunk or snippet (a "theme") to use for searching. With that theme in hand, the user opens the notation index. This has been sorted by sequential note letter name, as in a radix sort. By progressively scanning the list until the first letter in the sequence matches, then the second letter, then the third letter, and so on, the user may quickly find the desired piece of music.

For example, suppose the query is [G C D E♭ C B]. A user would sequentially search the index in Figure 2.1 until a G was found in position 1. This would match the first item in the index. Next, the user would sequentially search from that position until a C was found in position 2, and then a D in position 3; this is still the first item in the index. Next, an E♭ in position 4 is sought, which

drops the user down to the seventh item in the index. This would continue until a match was found, at which point the index B1524 indicates where to find the piece which corresponds to the theme.

Some of the first works on music retrieval by computer take a similar approach. Mongeau and Sankoff [81] match strings, but allow for insertion, deletions, and substitutions. Differences between two strings are weighted; for example, a consonant insertion is judged "closer" to the original than an insertion which is more dissonant. Ghias [49] uses a $k-$mismatch string matching algorithm which adds allowance for transpositions and duplications in addition to insertions and deletions.

Many other researchers have taken the string matching approach [77, 16, 35, 103, 36]. Some of this work uses simple edit distances to compute similarity, other works take a more musically "intelligent" approach, giving different weights to insertions and deletions of salient versus non-salient notes. In all above cases, both the query strings and document strings are monophonic sequences. The original source may have been monophonic or polyphonic, but it was necessarily reduced monophonically in order for these retrieval algorithms to function.

2.2.2 Pattern Matching

When the source collection or query is homophonic or polyphonic, string matching runs into trouble. The sequence is no longer one-dimensional. More generalized pattern matching becomes necessary. Recall that homophonic music can be characterized by a sequence of *sets* of pitches or pitch intervals. If each of those sets is treated as an atomic object, then we have a one-dimensional sequence, a string. But if each set is not treated atomically, if the members of the set may be searched individually then a whole new range of pattern matching approaches must be used.

For example, Iliopoulos [56] can find overlapping monophonic query strings within a homophonic source. "Overlapping" means that one monophonic instance of the query may begin before a previous instance has ended. This is useful with fugues, for example. The monophonic sequences found may also be evolutionary. Suppose that instance X of a query is found, which instance is no more than $k-$distant from the query by means of insertions, deletions, and substitutions. Then instance Y may also be found, which instance is no more than $k-$distant from instance X. But had instance X not existed, then instance Y would never have been retrieved, because it is too different from the original query. Thus, query matches within the source are allowed to slowly evolve.

Lemström [69] also find monophonic query sequences within a homophonic source. This is an adapted bit-parallel algorithm which, despite the homophony, detects both transposed and transposition invariant matches in $O(n)$ time. Dovey [41, 42] takes the notion of string matching for music information retrieval one step further. In his dynamic programming-based algorithm, polyphonic query and polyphonic source document can be matched, complete with insertions and deletions.

2.2.3 Standard Text Information Retrieval Approaches

Whereas the most appropriate feature type for the systems described above in Section 2.2.1 is the unigram, the retrieval models in this section presuppose the use of longer n-grams. An n-gram is similar to an alphanumeric text string, a word. While n-grams can be used for both text and music, the main difference is that in text, words may be easily extracted and bear significant semantic content, while in music, there is no such guarantee with n-grams (see Section 1.1.1 for additional discussion). Yet the probabilistic models which have been developed for text information retrieval are well-enough understood that application to music is a desireable endeavor.

The two most common probabilistic text approaches are the Bayesian Inference Network model [24] and the Vector Space Model [104]. Doraisamy uses the cosine similarity metric from the Vector Space model [40] on non-voiced n-grams extracted from polyphonic sources. Other researchers such as Downie and Melucci also successfully apply the Vector Space model to their longer n-grams [45, 78]. Pickens [88] uses inference networks with bigrams to arrive at probabilistic estimates of whether a user's information need was met. Uitdenbogerd [119] does a maximum likelihood n-gram frequency count, similar to the term frequency approaches of many text systems.

Though each of these researchers used features of their own choosing, it should be observed that any monophonic n-gram from Section 2.1.1.2 may be used in these probabilistic text retrieval

systems, whether pitches, pitch intervals, durations, duration ratios, atomic chord units, or the like. The use of these retrieval models also does not require any specific feature selection technique. As long as monophonic n-grams are present, and created in the same manner for both query and collection, it does not matter what the n-grams are "made of".

2.2.4 Suffix Trees

Standard string matching algorithms have a lower bound time complexity of $\Omega(n)$, where n is the size of the document. When one is searching a single music document for a string, this is not a problem. However, when one wants to search an entire collection, a linear scan through every document in the collection becomes impractical. A specialized approach to string matching comes in the form of suffix trees.

Standard suffix trees may be built in $O(n)$ time, where n is the length of the entire collection. They may be searched in $O(m)$ time, where m is the length of the query. The time complexity is desirable, but the space complexity is $O(n^2)$. A number of researchers have used suffix trees for monophonic music retrieval using monophonic queries [29, 67, 65, 28]. The trees have been adapted to handle music-specific issues such as approximate matches and multiples indices (pitch and duration, for example). Monophonic features of all kinds are used: pitch, duration, Lemström's *tdr* (see Section 2.1.1.1), and even chords (see Section 2.1.4.4).

2.2.5 Dynamic Time Warping

Dynamic time warping is a dynamic programming technique that has been used for a number of decades for a number of various tasks, including speech recognition, image recognition, score tracking, beat or rhythm induction, among others. This process aligns two sequences of features in a manner such that the optimal path between all possible alignments of the sequences is found; one sequence is warped (expanded and/or contracted) until the best possible fit with the second sequence is found. This optimal path is expressed in terms of the features or similarities being sought. A distance metric between features is created, and alignments of the two sequences that minimize the cost introduced by this distance metric are preferred. Dynamic programming is used so that the exponentially-many set of possible alignments does not need to be fully enumerated.

For example, in work by Paulus and Klapuri [87], the goal is to measure the rhythmic similarity between two pieces of music. The two most important features for beat tracking were determined to be perceived loudness and brightness. Loudness was measured by mean square energy of a signal within a frame. Brightness was measured by the spectral centroid of that signal. The feature vectors are related to these measures. Thus, frames in the sequence with high loudness and high brightness are brought closer together by the dynamic time warping algorithm, as are frames with low loudness and low brightness.

Though this technique has been applied to rhythmic similarity (as mentioned above) as well as general spectral similarity [46], we are not aware of any uses of dynamic time warping on chordal features. This is a direction this dissertation could have taken, and we are sure that at some point in the future this technique will be tried. We chose not to use it, however, because we felt it was too limited by its sequential, linear nature. For example, suppose a certain piece of music were broken up into three major sections, ABC. Suppose furthermore that a variation on that piece had made some changes to section B: AB′C. Then dynamic programming would work well by giving a higher alignment score from ABC to AB′C, and a lower score to some other piece DCCFA.

However, dynamic time warping would not work very well if certain sections were repeated or shuffled. Suppose for example that ABC became AABBCC. You often find this kind of repetition in music. The time warping algorithm would find an alignment, but the score might be low, depending on whether the algorithm was able to align section A with the repeated sections AA, without bleeding any of the A alignment into the B section. It gets even more complicated when multiple sections are repeated: ABABC, or ABCBC, or ABCABC. As is the nature of music, entire sections might even be switched in order: ACB. In these more complicated cases, it is our intuition that dynamic time warping is going to be problematic. For this reason, we chose not to focus on it and instead on a

modeling technique that makes only localized decisions about sequentiality. These are the Markov approaches mentioned in the next section.

2.2.6 Markov and Hidden Markov Models

Recently, researchers have begun to realize the value of sequential probabilistic models. After all, music is sequential in nature. Birmingham uses hidden Markov models for retrieval, creating 1^{st}-order models from monophonic pitch and duration sequences [15] These sequences are first obtained by reducing a polyphonic source to a monophonic sequence. Shifrin [110, 111] also uses hidden Markov models, ranking polyphonic models of music by their likelihood of generating a monophonic user query. Finally, Shalev-Shwartz [108] uses tempo as well as sequential spectral features to create hidden Markov models of raw polyphonic audio and ranks raw monophonic audio queries by the likelihood of the model generating that query. In Chapter 4 we also take the hidden Markov modeling approach to music information retrieval, using chords as our hidden-state features.

Rand [96] and Hoos [51] both apply 1^{st}-order Markov modeling to monophonic pitch sequences. Birmingham extends the modeling to the polyphonic domain, using both 0^{th} and 1^{st}-order Markov models of raw pitch *simultaneities* to represent scores [14]. Pickens [90, 89] recasts raw polyphonic pitch simultaneities as vectors of partial chord observations, and uses 0^{th} through 3^{rd}-order Markov models to record the probabilities of chord sequences. The latter work also builds transposition invariance into the model, taking into account the possibility that a variation might exist in another key. Purwins [94] has devised a method of estimating the similarity between two polyphonic audio music pieces by fitting the audio signals to a vector of key signatures using real-valued scores, averaging the score for each key fit across the entire piece, and then comparing the averages between two documents. This can be thought of as a 0^{th} order Markov model. In Chapter 5 we take the Markov modeling approach to music information retrieval, and continue to flesh out earlier related work [90, 89].

2.2.7 Other Work

There are undoubtedly many more systems and retrieval models for both monophonic and polyphonic music which we have not mentioned here. The past year or two has seen a tremendous explosion in the number of papers, as well as variety of venues, at which music information retrieval work has been published. Also important to note is that we have not covered any of the audio-only or metadata music retrieval work, those that function by determining similarity of genre, mood, or timbre. Although we do bridge the gap from audio to symbolic representations, as will be explained in the next chapter, our focus is on symbolic-based thematic ("melodic") similarity (the Shalev-Shwartz citation was a notable exception, as it operates on raw audio, but we included it anyway because it bore similarities to our work in many other ways). We therefore focused primarily on similar works in this literature review.

CHAPTER 3

CHORDS AS FEATURES

In the words of Blackburn, "Feature extraction can be thought of as representation conversion, taking low-level representation and identifying higher level features" [18]. Features at one level may build upon features at a lower level. Techniques employed for feature extraction range from string-matching algorithms familiar to a computer scientist to "deep structure" approaches more familiar to a music theorist. Our goal, however, is not to develop a better theory of music, or even to analyze music. The goal is retrieval. Computational and music-theoretic analyses of music might aid that goal, but we consider them important only insofar as they aid the retrieval effort. The purpose of this chapter is to define and describe the pre-processing steps for the basic features that will be used in our retrieval models of Chapters 4 and 5.

3.1 Data Preparation

It is possible that the pieces of music which will be searched or which will be used as queries exist in a format not immediately useful for our systems. Therefore, the first stage is to translate from that data format to one which we understand. For example, much of our collection (approximately 3000 pieces from the CCARH [54]) existed in the Kern/Humdrum format [55]. Some of our data also came in the Nightingale format [3]. And some of our data existed as MIDI files [80]. In each of these cases, we had to build parsers which could read and understand each format. Though some of these formats are much more complex than others (Kern, for example, is a conventional music notation format, while MIDI is a time-stamped event format), all of the data contains symbolic representations of pitch and duration.

However, some of our music queries came from the Naxos collection, in the form of raw, uncompressed audio [2]. Extracting pitches from this data is a much tougher problem. Therefore, techniques external to this work were used, as will be explained in section 3.1.1. These techniques are not perfect, and not only are many incorrect pitches introduced and many correct pitches missed, but occasionally entire onsets of pitches are missed. Nevertheless, once the data is extracted or translated, from whatever source, we convert that data into *simultaneities*.

3.1.1 Step 0: (Optional) Polyphonic Audio Transcription

As explained in Chapter 1, while the musical data to which we apply our algorithm necessitates that pitch information is available, the raw data that we start with might be in some other format, such as audio. If this is the case, then we need to begin our data preparation with a transcription step.

Automatic music transcription is the process of transforming a recorded audio signal into the symbolic values for the actual pitches, durations, and onset times of the notes which constitute the piece. Monophonic transcription is a difficult problem, but the task becomes increasingly complicated when dealing with polyphonic music because of the multiplicity of pitches, varied durations, and rich timbres. Most monophonic transcription techniques are therefore not applicable. In fact, despite several methods being proposed with varying degrees of success [98, 39, 61, 74, 76], automatic transcription of polyphonic music remains an unsolved problem. We have therefore restricted ourself to polyphonic, monotimbral audio transcription: the notes are polyphonic, but no more than a single instrument (in our work, always piano) is playing. We use two outside algorithms for the transcription procedure, the first by Monti [82] and the second by Bello [11, 12]. Additional details on each of these algorithms can be found in Pickens [90].

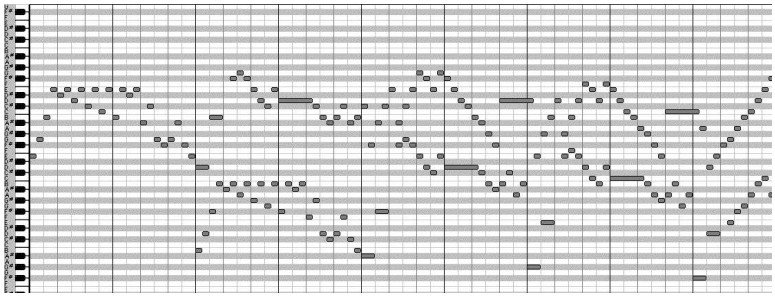


Figure 3.1. Bach Fugue #10, original score

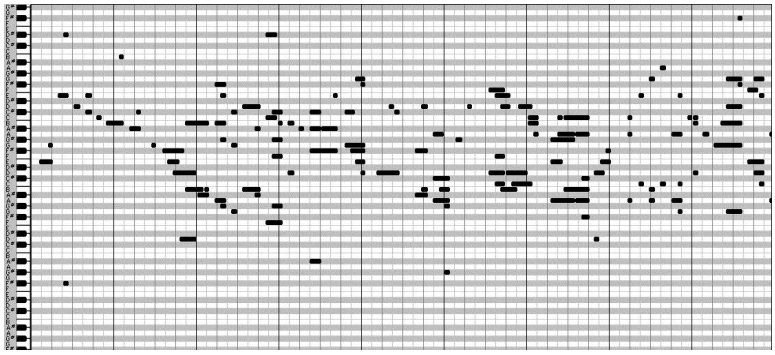


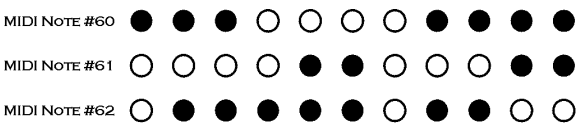
Figure 3.2. Bach Fugue #10, Bello polyphonic transcription algorithm

We offer two figures as an example of this transcription procedure. Figure 3.1 is the original score of Bach's Fugue #10 from Book I of the Well-tempered Clavier, presented here in piano-roll notation. Figure 3.2 is the transcription from one of the transcription algorithms we use. With this quite imperfect transcription we can still achieve excellent retrieval results, as will be demonstrated in Chapter 6.

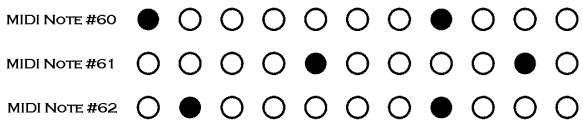
3.1.2 Step 1: Simultaneity Creation

We define *simultaneity* as an octave-invariant (mod 12) pitch set. We use the name simultaneity because these entities are created from polyphonic music by extracting *at every point in time* either all notes which *start* at that point in time [41], or all notes which are *sounding* at that point in time [69]. For the purpose of this work, we have chosen to create simultaneities in the former manner, ignoring durational information and adding to each simultaneity all pitches of notes which start at the same time.

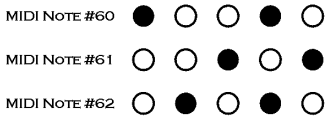
We may think of polyphonic music as a two-dimensional graph, with time along the x-axis, and pitch number (1 to 128) along the y-axis. At any point along the y-axis, notes turn "on", remain "on" for a particular duration, and then turn back "off" again. As an example, see the figures below. Black circles represent notes being "on". White circles represent notes being "off".



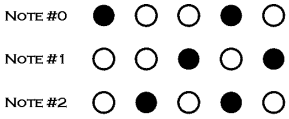
We begin simultaneity creation by selecting only the onset times of each new pitch in the sequence, and ignoring the duration of the note. This is a homophonic reduction, described in Section 2.1.4.2. The example above thus transforms into:



Next, we get rid of all onset times which contain no pitches. We are throwing away not only the duration of the notes themselves, but the duration between notes. We feel this is necessary for a first-stage modeling attempt. Future models might contain more complexity. All those onset times which do contain pitches, however, we give the specialized name "simultaneity".



Finally, we reduce the 128-note y-axis to a 12-note octave-equivalent pitch set. We do this simply by taking the mod-12 value of every pitch number. The example above thus becomes:



So we are left with a sequence of 12-element bit vectors; there is either a 1 or a 0 in each spot, depending on whether a note of that (mod 12) pitch had an onset in that particular simultaneity. The steps to create these vectors may be summarized as follows:

1. At every point in time at which a new note begins, a simultaneity is created
2. All notes that start at that time are added to the simultaneity (notes that are still sounding, but began at a previous point in time, are not added)
3. Duration of the notes is ignored. Duration between simultaneities is ignored
4. The MIDI pitch value of all the notes in each simultaneity is subjected to a mod 12 operation, to collapse the pitches to a single octave

3.2 Chord Lexicon

As the primary features we will be using in this work are chords, we need to define a dictionary, or lexicon, of allowable chord terms. We define a *lexical chord* as a pitch template. Of the 12 octave-equivalent (mod 12) pitches in the Western canon, we repeatedly select some n-sized subset of those, call the subset a *chord*, give that chord a name, and add it to the lexicon. Not all possible chords belong in a lexicon; with $\binom{12}{n}$ possible lexical chords of size n, and 12 different choices for n, we must restrict ourselves to a musically-sensible subset.

3.2.1 Chord Lexicon Definition

The chord lexicon used in this work is the set of 24 major and minor triads, one each for all 12 members of the chromatic scale: C Major, c minor, C♯ Major, c♯ minor ... B♭ Major, b♭ minor, B Major, b minor. Assuming octave-invariance, the three members of a major triad have the relative semitone values n, $n+4$ and $n+7$; those of a minor triad n, $n+3$ and $n+7$. No distinction is made between enharmonic equivalents (C♯/D♭, A♯/B♭, E♯/F, and so on). Thus our chord lexicon consists of the values found in Table 3.1.

	C	C♯	D	E♭	E	F
Major	0 4 7	1 5 8	2 6 9	3 7 10	4 8 11	5 9 0
Minor	0 3 7	1 4 8	2 5 9	3 6 10	4 7 11	5 8 0
	F♯	G	A♭	A	B♭	B
Major	6 10 1	7 11 2	8 0 3	9 1 4	10 2 5	11 3 6
Minor	6 9 1	7 10 2	8 11 3	9 0 4	10 1 5	11 2 6

Table 3.1. Chord lexicon

3.2.2 Intuitive Underpinnings

There are two intuituions we need to explain. The first is why we chose chords as our features, and the second is why we chose to limit our lexicon to the 24 major and minor triads. The two intuitions are not unrelated.

Instead of choosing chords as features, it would have been perfectly reasonable to simply use the notes themselves. Notes can be searched, they can be stochastically modeled, and so on. The problem we are trying to solve, however, is to develop a credible method for determining music similarity, where similarity is defined to inclue both variations on a theme and as degraded, audio-transcribed known items (see Chapter 1). As such, it is common for notes that do not "belong to" the prevailing theme to occur, and for notes that do belong to the prevailing theme not to occur. Variations, in other words, are characterized by numerous or almost constant note insertions and deletions.

If we were doing straight matches or models of the notes themselves, we would not have any notion about which notes are "good" insertions or deletions, and which notes are "bad" insertions or deletions. In other words, it is less harmful if certain notes are added, and more harmful if certain others are added. It is also less harmful if some notes are missing, but not others. Add or delete enough of the "wrong" notes and the piece of music turns into an entirely different piece. But add or delete the same number of "right" notes, and it is still the same piece of music, the same theme. It is not the number of notes that matters; it is which notes.

We are guided by the assumption that thematic similarities are going to share harmonic similarities as well. Thus, the intuition to use chords comes from the need to have a guide for which notes are good or bad insertions and deletions. By developing models in which we infer likely sequences of chords we gain that guidance. Even if a good note is missing, or a bad note is inserted, as long as it does not affect the prevailing harmony it should have little effect. By the same token, if the addition or deletion of a certain note does affect the prevailing harmony, that note is critical in understanding how similar one piece of music is to another. Chords as features are the guide by which the consequence or significance of individual notes note can be determined. Stated in another manner, we feel that chords are a *robust* feature for the type of music similarity retrieval system we are constructing.

The second intuition deals with our particular lexicon. We have chosen a rather narrow space of chord features: 12 major and 12 minor triads. We did not include dyads or note singletons. We did not include more complex chords such as 7^{th}, 9^{th}, 11^{th} or 13^{th} chords. We did not include other chords such as jazz chords, mystic chords, augmented triads, diminished triads, augmented 6^{ths}, and so on. Neither did we include other dissonant chords such as a [C, C♯, F♯] chord.

Our intuition is that by including too many chords, both complex and simple, we run the risk of "overfitting" our chord-based models to a particular piece of music. As a quick thought experiment, imagine if the set of chords were simply the entire $\sum_{n=1..12} \binom{12}{n} = 2^{12} - 1$ possible combinations of 12 octave-invariant notes. Then the extracted chord features would simply be the raw simultaneities, and we would not gain any discrimination power over which notes are "good" or "bad" insertions and deletions. This is an extreme example, but it illustrates the intuition that the richer the lexical chord set becomes, the more our feature selection algorithms might overfit one piece of music, and not account well for future, unseen variations. Furthermore, Tim Crawford, a musicologist with whom we had many discussions in the early stages of this work, shares this intuition:

> I am not sure you will need to include higher-order chords given the proposed probability-distribution model. They can be 'decomposed' into overlapping triads in general, and the distributions will account for that. Or at least I think so. It will be interesting to see. The problem is where to stop in elaborating the 'lexicon' of chords to use in the description. Intuitively I feel that it should be as simple as possible.[33]

In this work we do not test our choice of chord lexicon directly by comparing it against other chord lexicons on the same collection, or with the same chord lexicon on other collections (on a jazz collection rather than a classical collection, for example). So at this point, our choice of the chord lexicon remains a simplifying assumption, something that may not be completely accurate but which is necessary as a first-stage feature extraction attempt.

While it is clear that the harmony of only the crudest music can be reduced to a mere succession of major and minor triads, as this choice of lexicon might be thought to assume, we believe that this is a sound basis for a probabilistic or partial observation approach to feature extraction. As our goal is not the selection of a single, most salient lexical chord, but a distribution or partial observation over possible harmonic chords, we feel that the set of triads is large enough to distinguish between harmonic patterns, but small enough to robustly accomodate harmonic invariance.

3.3 Chord Selection

Now that we have prepared the data and selected a chord lexicon, the final stage of our feature extraction is to fit the simultaneities to our lexical chord set. The exact details are found in Chapters 4 and 5. However, we wish to make clear the notion that we want some sort of multiple chord selection for each simultaneity. This is a different mindset from those trying to do a more theory-based harmonic analysis or chord reduction.

In Section 2.1.4.4, unvoiced polyphonic music is reduced in a one-dimensional sequence of atomic chord objects. At each step in time, one and only one chord is selected as representative of the polyphonic source. Of course, due to the nature of polyphonic music, it is quite conceivable that more than one chord exists as a potential candidate at any given time step. The question is how to select the "correct" candidate.

Prather [93] overcomes the ambiguity by examining neighboring time windows. For example, imagine the following chord candidates at neighboring time steps. The chord selected as representative of timestep $n+1$ will be the "A minor" triad, because it is found in both neighboring windows.

Timestep	Chord Candidates
n	C Major, A minor
$n+1$	A minor, F minor
$n+2$	C Major, A minor, A minor

Chou [29] also overcomes the ambiguity and selects only a single chord as representative of each time step by using heuristic clues such as frequency and consonance. From the example above, the "C major" triad is selected at timestep n, because a major triad is more consonant than a minor triad. However, at timestep $n+2$, the "A minor" triad occurs the most frequently, so it is selected over the more consonant "C major" triad.

There are problems with both of these approaches. For example, at timestep $n+1$, both the "A minor" and the "F minor" are equally frequent and equally consonant. Which should be selected? It is not clear. Furthermore, even though timestep $n+1$ contains no "C major" triad, as it is surrounded by timesteps with "C major" triads, this chord could be a viable candidate.

These problems could be corrected with better heuristics, but there is an even more fundamental problem, one which cannot be solved by more intelligent chord selection. This is the notion that, in music, composers like to play around with chords, and make more than one chord salient in a given time step. Sometimes, a given timestep is best described by both a "C major" and an "A minor" triad. This can be true if the simultaneity consists of the notes [C-E], or if the simultaneity

consists of the notes [A-C-E-G]. No single chord effectively represents the music. This is especially true because our chord lexicon is limited. The problem is not just solved by adding a major 3^{rd} dyad on C and an A minor $7th$ chord. Short of adding the full set of raw simultaneities to the lexicon, there will never be perfect fits between the raw data and the chord lexicon. Any method which attempts to extract only a single chord from that timestep, no matter how intelligently, will capture an incorrect representation of the music source.

The alternative is simply not to limit chord extraction to a single item. One possibility is instead of eliminating unused candidate chords, we place all candidates into a set. An unvoiced polyphonic source is thus recast as a sequence of *chord sets*. Each chord is still an atomic unit, but there are multiple such units coexisting at every time step. These chord sets can then be searched in any manner in which homophonic note sets are searched.

A second option is to attach a weight to each of the candidate chords in the set. Then, using ideas gleaned from Chou [29] and Prather [93] such as frequency, consonance or dissonance, and other methods for smoothing across neighboring windows, we can *reshape* the chord distribution and gain a better estimate of the salient chords within the current window.

Thus, instead of a single chord at each time step, one has either a non-parametric distribution (Chapter 4) or a vector of partial chord observations (Chapter 5). Modeling and searching can then be done on these weighted chord sets. Either way, an incorrect selection of the one, most salient chord becomes less threatening to the retrieval process, as hypotheses for all candidates are continually entertained and no candidate is eliminated completely.

CHAPTER 4

HIDDEN MARKOV MODELS

Now the we have defined chords as the primary feature or information in which we are interested, we need some way of making use of that feature for the purpose of retrieval. In other words, we need a framework, or system. Most of the existing music retrieval systems utilize string matching and other general pattern matching techniques. Not only are these approaches often limited by their inability to generalize from the monophonic to the polyphonic case, but they do not allow one to make use of statistical regularities which might be useful for music. Thus, rather than finding and extracting strings of notes, we propose building a probabilistic model of each piece of music in the collection, and then ranking those models by their probability of generating the query.

The models we use are capable of characterizing the harmony of a piece of music at a certain point as a *probability distribution* over chords, rather than as a single chord. Selecting a single chord is akin to inferring the semantic meaning of the piece of music at that point in time. While useful for some applications, we feel that for retrieval, this semantic information is not necessary, and it can be harmful if the incorrect chord is chosen. Rather, we let the statistical patterns of the music speak for themselves. We are thus adapting to music the "language" modeling approach to Information Retrieval. Hidden Markov models are the first manner in which we do this.

4.1 System Overview

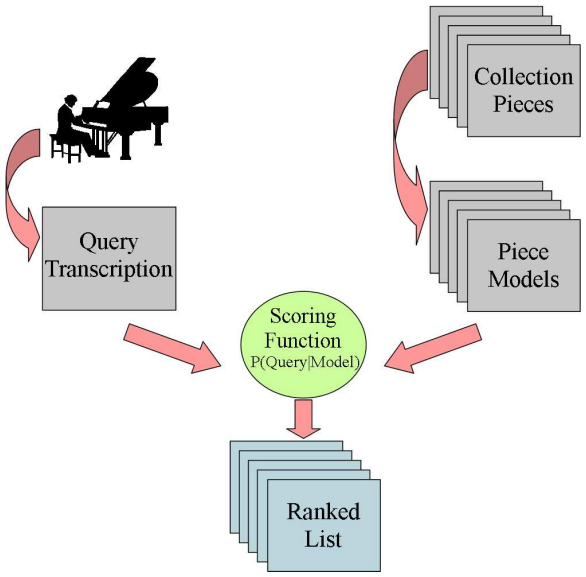


Figure 4.1. Overview of HMM-based retrieval system

Figure 4.1 contains an overview of a music information retrieval system based on hidden Markov models. In Chapter 3 we covered the process of (optionally) transcribing a piece of music from raw

audio and then (non-optionally) selecting a sequence of simultaneities from the symbolic representation. The query which is fed into the system is this sequence of simultaneities.

On the source collection side, however, a bit more processing needs to be done. We start by extracting simultaneity sequences from each piece of music in the collection. Next, a hidden Markov model is estimated for each piece, individually. The estimation is done by first initializing the parameters of the model in a musically sensible manner, and then using standard HMM estimation techniques to iteratively adjust the parameters so that the probability of producing the simultaneity sequence (the *observation*) given the model is maximized. Probability distributions over chord sequences are learned concurrently with the probability distributions over observations. Thus, feature extraction, as discussed in the previous chapter, is an integral part of the model.

With an HMM of every piece of music in the collection, and with a query simultaneity sequence (an *observation*) as well, we may then ask the question of each HMM: how likely is it that this HMM could have produced this query? Pieces are then ranked by this likelihood. The remainder of this chapter contains the details of the model estimation and query likelihood determination problems.

4.2 Description of Hidden Markov Models

Our usage of the hidden Markov model framework is standard. In this section we review the components of an HMM and explain how we adapt these components to our chord-based music modeling. For an excellent, in-depth tutorial on HMMs, we refer the reader to a paper by Rabiner [95]. A fully specified HMM, λ, contains the following components. First, the model contains a finite vocabulary of states and a finite vocabulary observation symbols:

$\{s_1, \ldots s_N\}$	the size N set of states
$\{k_1, \ldots k_M\}$	the size M set of observation symbols

Next, the following probability distributions involving these states and observations are needed:

π_i	the probability of starting a sequence in state s_i
$A_{i,j}$	the probability of transitioning from state s_i to state s_j
$B_{i,l}$	the probability of outputting the observation symbol k_l while in state s_i

Finally, we notate a particular sequence of states and observations as:

$X = \{x_1, \ldots, x_T\}$	the sequence of states, of length T
$O = \{o_1, \ldots, o_T\}$	the sequence of observation symbols, of length T

Now that these terms are defined, we need to know what values they assume for our models. Figure 4.2 is an example hidden Markov model, and we will use it as a reference. It represents an HMM for a single piece of music. The nodes along the top row are the sequence of states, X. The nodes along the bottom row as the sequence of observations. The length of the sequence, T, is specific to each piece of music. We set the length of the sequence equal to the number of points in time at which there are note onsets. In other words, T is equal to the number of simultaneities in the piece. Consequently, O is simply the (observable) sequence of these simultaneities, and X is the (hidden) sequence of states. In Figure 4.2, T is equal to 4, so that $O = \{o_1, o_2, o_3, o_4\}$ and $X = \{x_1, x_2, x_3, x_4\}$.

Next, N, the number of state values in one of our models, is 24. There is one state for each of the 12 major and 12 minor triads, as explained in the previous chapter. Thus, each state $x_1 \ldots x_4$ in our example can take on one of 24 different values, s_1 through s_{24}. Furthermore, M, the number of distinct observation symbols, is a discrete alphabet of size $2^{12} - 1 = 4095$; our observations are the note simultaneities. Recall from Chapter 3 the manner in which simultaneities are created. At every point in time throughout a piece of music, all notes which start (have their onset) at that time are selected and added to the simultaneity. The mod-12 (octave invariance) of the pitch values insures that there are no more than 12 different notes in the simultaneity. By definition, simultaneities are

extracted only when a new note onset occurs; therefore, there are never any noteless simultaneities. Thus, a simultaneity is a 12-bit vector with values ranging from 000000000001 to 111111111111. The 000000000000 simultaneity will never be observed, and so is excluded from the vocabulary. Again, this yields $2^{12} - 1 = 4095$ distinct possible observations. Each observation $o_1 \ldots o_4$ in Figure 4.2 takes on one value (m_1 through m_{4095}) from this vocabulary.

The initial state distribution, π, is a distribution over the starting state of the sequence. The state transition probability matrix, A, is a distribution over how likely it is that we transition into some next state, given the current state. We sometimes write this as $P(s_{i+1}|s_i)$. Finally, the observation symbol matrix, B, is a probability distribution over which observation symbols one might see, given the current state in the sequence. This can also be written as $P(o_i|s_i)$. In the next few sections, we will explain how we estimate the parameters of the π, A, and B distributions for a piece of music, and then how we use those values to implement a retrieval system.

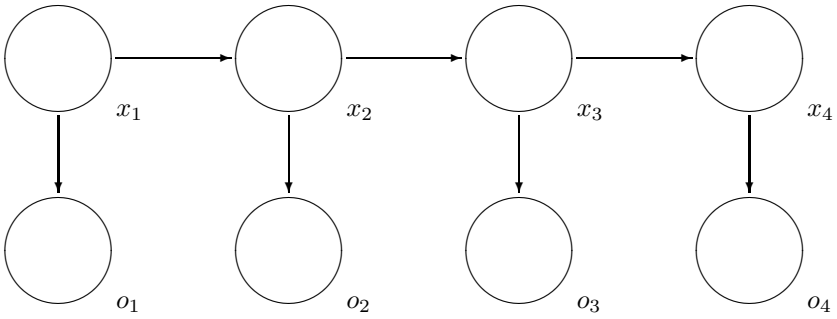


Figure 4.2. Example hidden Markov model sequence

4.3 Model Initialization

The parameter values for π, A, and B are not given to us and need to be determined on a per-piece-of-music basis. Fortunately, standard algorithms such as Baum-Welch exist for estimating these distributions in an unsupervised manner. However, these estimation algorithms suffer from the problem that, depending on the initial values of π, A, and B, re-estimation might get stuck on a local maximum. Therefore it is necessary to select initial estimates which put us in a region where we may find the global maximum. This section explains how we choose our initial distributions, which owe their basic form to discussions with Chris Raphael [99].

While perhaps unwise, random initialization of the parameters is an option. In Chapter 6 we will compare retrieval results on HMM systems with random initialization against HMM systems with the more intelligently selected initialization values we provide in this section. Furthermore, of the three distributions, π, A, and B, the observation symbol distribution is the most sensitive to model parameter reestimation algorithms. Rabiner mentions that "experience has shown that either random...or uniform initial estimates of the π and A parameters is adequate for giving useful reestimates of these parameters in almost all cases. However, for the B parameters, experience has shown that good initial estimates are helpful in the discrete symbol case, and are essential...in the continuous distribution case." [95] We are dealing with the discrete case; nevertheless, we offer two variations on initial estimates for B, which we call $Model_0$ and $Model_1$, in an attempt to find values which are more "helpful". These two models share the same initial values for π and A, and only differ on how B is constructed.

4.3.1 Initial State Probability [π] Initialization

Prior to encountering an actual piece of music, we have no reason to prefer any state over any other state. A C major triad is just as likely as an F$^\sharp$ major triad. We have no reason to believe, in other words, that our model will start out in any particular state. Therefore, in all of our models, we set $\pi_i = \frac{1}{N} = \frac{1}{24}$. Initially, we are equally likely to start off in any state. The more intelligently chosen A and B distributions will help in the reestimation of π.

4.3.2 State Transition Probability [A] Initialization

While we do not prefer any state over any other state for the initialization of π, we do have a priori preferences about which state might follow another state. This is because the music to which we restrict ourselves within this work stems from the Common Practice Era (European and U.S. music from 1600-1900). Music composed in this time is based on fairly standard theoretical foundations which let us make certain assumptions in our initialization procedures.

4.3.2.1 Assumptions

In particular, the common practice era notion of the *circle of fifths* is crucial. The circle of fifths essentially lays out the 12 major and 12 minor keys in a (clockwise) dominant, or (counter-clockwise) subdominant relationship. Essentially, keys nearer to each other on the circle are more consonant, more closely related, and keys further from each other are more dissonant, less closely related. We translate this notion of closely related keys into a notion of closely related triads (chords) which share their tonic and mode with the key of the same name. In other words, because the C major and G major keys are closely related, we assume that the C major and G major root triads are also closely related.

Though standard circle of fifths visualizations do not make the following distinction, we differentiate between the root triad of a major key and the root triad of that key's relative minor. Thus, we may view the 24 lexical chords (C major, c minor, C$\sharp$ major, c$\sharp$ minor ... B$\flat$ major, b$\flat$ minor, B major, b minor) as points on two overlapping 'circles of fifths', one for major triads, the other for minor triads. Each circle is constructed by placing chords adjacently whose root pitch is separated by the interval of a fifth (7 semitones); for example, G major or minor (root pitch-class 7) has immediate neighbours C (7 - 7 = 0) and D (7 + 7 = 14, i.e. octave-invariant pitch-class 2). Thus each major tonic chord (G major, say) stands in appropriately close proximity to its dominant (D major) and subdominant (C major) chords, i.e. those to which it is most closely related in music-theoretical terms. The two circles (major and minor) may be aligned by placing major triads close to their respective relative minor triads, as shown in Figure 4.3 (major triads are shown in upper case, minor triads in lower case).

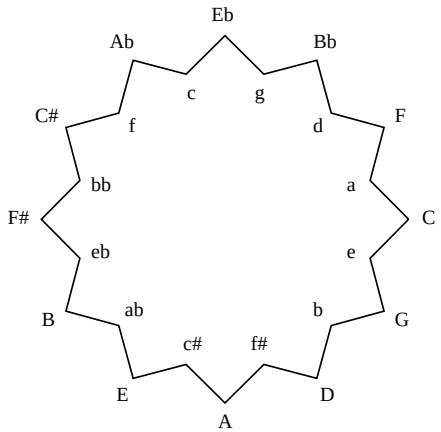


Figure 4.3. Lexical chords and their relative distances

Generally speaking, we are making the assumption that the minor triad on the root of the key which is the relative minor of some major key is more closely related to the major triad on the subdominant of that major key than to the major triad on the dominant of that major key, simply because they share more notes.

4.3.2.2 Initial Distribution Values

With these ideas about lexical chord relative distances, we have a basis on which we can create an initial state transition probability distribution: *triads that are more closely related, more consonant against each other, have a higher initial transition probability than triads which are less closely related, less consonant against each other.*

As this is the distribution initialization stage, it should not matter what the actual probabilities are. It only matters that, relative to each other, certain chord transitions are more likely than others. We begin by giving the transition from any given chord to itself the highest probability, as a chord is most consonant with itself: $p = \frac{12+\epsilon}{144+24\epsilon}$, where ϵ is a small smoothing parameter. Then, working our way both clockwise and counter-clockwise around our nested circle of fifths (Figure 4.3), we assign the two next closest related chords a probability of $\frac{11+\epsilon}{144+24\epsilon}$, the next two a probability of $\frac{10+\epsilon}{144+24\epsilon}$ and so on, until we reach the most distant chord, which we assign a probability of $\frac{0+\epsilon}{144+24\epsilon}$.

Given that we know nothing about a particular piece of music to be modeled, we at least know that most composers, especially from the era of common practice, are (most of the time) going to make smooth chordal transitions from one note onset to the next. Without knowing anything else about a piece of music, we state that it is much more likely for that piece of music to transition from a C major to a G major to an A Minor to a C major, than it is for it to transition from a C major to a B major to a G Minor to a C major. It is not impossible, and the standard hidden Markov reestimation technique covered in Section 4.4 should adjust the probabilities if this latter sequence is more likely for the piece of music under consideration. But by making certain transitions more likely than others, and in a manner which resembles actual composed practice, our hope is that we may avoid some of the local maxima at which parameter reestimation might get stuck.

The full initial state transition matrix is found in Table 4.1. Major triads are written in uppercase; minor triads are written in lowercase. In the interest of space, each element in the table has been multiplied by 144 and we do not include the smoothing parameter, ϵ. Thus, to recover the actual probability, one should add ϵ and divide by $144 + 24\epsilon$.

4.3.2.3 A Short Critique

One critique of this work is that by initializing the distribution in this manner we might only sucessfully do retrieval on music from the era of common practice. In a sense, this is a circular problem. We have chosen the initial distributions in this manner because we know the type of music we are dealing with. If we were working with another type of music, we would create different initial distributions more reflective of that music. And if our collection were a mixed bag of some music from the era of common practice, and other music outside of that era which did not follow the same theoretical foundations, we could either (1) initialize our distributions in a manner which makes fewer (weaker) assumptions, or (2) train some sort of statistical classifier which learns to differentiate between the different types of music in our collection, and then chooses different initialization parameters based on the class. Either way, we wish to emphasize that the assumptions we make in this section do not limit us permanently to a single type of music, nor do they in any way invalidate the statistical modeling approach as a whole. It is beyond the scope of this work to test different initialization assumptions on collections of different types of music; however, it is entirely possible to apply our techniques in other musical contexts.

4.3.3 Observation Symbol Probability [B] Initialization

Choices for a proper initial observation symbol distribution are not as clear. While music-theoretic notions of harmonically-related chords provided an inspiration for the state transition distribution,

	C	a	F	d	B♭	g	E♭	c	A♭	f	C♯	b♭	F♯	e♭	B	a♭	E	c♯	A	f♯	D	b	G	e
C	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11
a	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10
F	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9
d	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8
B♭	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7
g	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6
E♭	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5
c	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4
A♭	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3
f	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2
C♯	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0	1
b♭	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1	0
F♯	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2	1
e♭	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3	2
B	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4	3
a♭	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5	4
E	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6	5
c♯	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7	6
A	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8	7
f♯	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9	8
D	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10	9
b	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11	10
G	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	11
e	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12

Table 4.1. Initial HMM state transition distribution

there are fewer formal notions for chord-to-simultaneity matching. There are certainly many algorithms for the analysis of chords, as we have detailed in Chapter 2. However, these often involve complicated sets of rules or heuristics. Our intent at this stage is not to do a full-blown harmonic analysis. Rather, we are looking for simple, effective methods for initialization; the automated mechanisms of the hidden Markov model formalism should take care of the rest.

In the following pages we present two models based on slightly different initial observation symbol distributions: $Model_0$ and $Model_1$. Both models use the same initialization values for π and A. They just differ in how they initialize their observation distribution, B. $Model_0$ was developed to make the observation distribution as generalizable as possible. $Model_1$ was developed to fit the observation distribution closer to the potential "true" estimates. In order to get a more accurate baseline comparison, $Model_1$ is patterned after the Harmonic modeling approach, which will be explored in the next chapter.

4.3.3.1 $Model_0$ - Participatory

We give $Model_0$ the nickname *participatory*. When giving the initial estimate for the probability of a (simultaneity) observation given a (chord) state, all observations that "participate", or share at least a single note with the given chord, are given equal probability mass. Observations which do not participate in the given chord are given a small probability, as it still might be possible for a state to generate these observations. The pseudocode for this algorithm is:

```
1   initialize every element of B to zero
2   for all 24 states s_k
3       for all 4095 observations o_l
4           if s_k and o_l have at least one note in common
5               B_{s_k,o_l} = 1 + ε
6           else
7               B_{s_k,o_l} = 0 + ε
8       normalize all elements in row B_{s_k} by the sum for that row
```

An example subset of the initial output symbol probability matrix, $P(o|s)$, can be found in Table 4.2. In the interest of space, we have not added the ε minimum probability, nor have we normalized by the sum for the entire row, which is the sum of all overlaps (3584 of 4095 observations share at least one note with any given chord) plus the sum of all ϵ which have been added to each value in the entire row (4095ε). Thus, to recover the actual initial probability for, say, the observation 001001000100 (the notes [d, f♯/g♭, a]) given a D minor triad, we have, for $\varepsilon = 0.00001$:

$$P(001001000100 \mid D\ Minor) = \frac{1+\varepsilon}{3584+4095\varepsilon} \approx 2.79 \times 10^{-4}$$

For comparison, the initial probability of the observation 000000001001, which shares no notes with $D\ Minor$, is:

$$P(000000001001 \mid D\ Minor) = \frac{0+\varepsilon}{3584+4095\varepsilon} \approx 2.79 \times 10^{-9}$$

4.3.3.2 $Model_1$ - Proportional

As mentioned in the previous section, for any given state, $\frac{3584}{4095}$ (approximately 7 of every 8 observations) participate in that state. Thus the initial $Model_0$ probability across all observations, given that state, is almost uniform. Such a model makes weak assumptions about the connection between states and observations. This might allow us to generalize better, but the model also has to rely more on the initial chord transition probabilities [A] to come up with an accurate model for a particular piece of music. We feel this might place too much burden on the HMM learning mechanisms.

For our second model, $Model_1$, we want to make stronger assumptions about states and observations. Specifically, we weight the initial observation probabilities commensurate with the number of

	C	a	F	d	B♭	g	E♭	c	A♭	f	C♯	b♭	F♯	e♭	B	a♭	E	c♯	A	f♯	D	b	G	e
000000000001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1
000000000010	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
000000000011	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
000000000100	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
000000000101	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
000000000110	0	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0
000000000111	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
000000001000	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0
000000001001	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1	1
⋮	⋮																							
001001000001	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1
001001000010	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0
001001000011	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
001001000100	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0
001001000101	0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	1	1	1	1	1	1
001001000110	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1	0
001001000111	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
001001001000	0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0
001001001001	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1
⋮	⋮																							
100010010000	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	1	1
100010010001	1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1	0	0	1	1	1
100010010010	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1	1	0	0	0	1	1
⋮	⋮																							
111111111011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
111111111100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
111111111101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
111111111110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
111111111111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 4.2. Initial HMM observation symbol distribution — $Model_0$

	C	a	F	d	B♭	g	E♭	c	A♭	f	C♯	b♭	F♯	e♭	B	a♭	E	c♯	A	f♯	D	b	G	e
000000000001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1
000000000010	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
000000000011	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
000000000100	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
000000000101	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
000000000110	0	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0
000000000111	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
000000001000	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0
000000001001	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	2	2	1	0	0	0	1	1	1
⋮	⋮																							
001001000001	0	0	1	2	2	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	2	2	1
001001000010	0	0	1	2	3	2	1	0	0	1	1	2	1	1	0	0	0	0	0	0	1	1	1	0
001001000011	0	0	1	2	3	2	1	0	0	1	1	2	1	1	1	1	1	0	0	0	1	2	2	1
001001000100	0	1	2	3	2	1	0	0	0	1	1	1	0	0	0	0	0	0	1	1	2	1	1	0
001001000101	0	1	2	3	2	1	0	0	0	1	1	1	0	0	1	1	1	0	1	1	2	2	2	1
001001000110	0	1	2	3	3	2	1	0	0	1	1	2	1	1	0	0	0	0	1	1	2	1	1	0
001001000111	0	1	2	3	3	2	1	0	0	1	1	2	1	1	1	1	1	0	1	1	2	2	2	1
001001001000	0	0	1	2	2	1	0	0	1	2	2	1	0	0	0	1	1	1	0	0	1	1	1	0
001001001001	0	0	1	2	2	1	0	0	1	2	2	1	0	0	1	2	2	1	0	0	1	2	2	1
⋮	⋮																							
100010010000	3	2	1	0	0	1	1	2	1	1	0	0	0	0	0	0	1	1	1	0	0	0	1	2
100010010001	3	2	1	0	0	1	1	2	1	1	0	0	0	0	1	1	2	1	1	0	0	1	2	3
100010010010	3	2	1	0	1	2	2	2	1	1	0	1	1	1	0	0	1	1	1	0	0	0	1	2
⋮	⋮																							
111111111011	3	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	3	3	3
111111111100	3	3	3	3	2	2	2	3	3	3	3	2	2	2	2	2	2	3	3	3	3	2	2	2
111111111101	3	3	3	3	2	2	2	3	3	3	3	2	2	2	3	3	3	3	3	3	3	3	3	3
111111111110	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	3	3	3	3	2	2	2
111111111111	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Table 4.3. Initial HMM observation symbol distribution — $Model_1$

notes the observation and the state have in common. Thus, $Model_1$ is a *proportional* model. Initial probabilities are assigned proportional to the number of notes a state and an observation share, with a small smoothing amount also given for observations with no overlap. The pseudocode for this algorithm is:

```
1    initialize every element of B to zero
2    for all 24 states s_k
3        for all 4095 observations o_l
4            proportion = number of notes that s_k and o_l have in common
5            B_{k,l} = proportion + ε
6        normalize all elements in row B_k by the sum for that row
```

The states in our models are triads, so an observation can have at most 3 notes in common with any state. To be exact, for any given state, there are exactly 511 observation symbols with 0 common notes, 1536 symbols with 1 common note, 1536 symbols with 2 common notes, and 512 symbols with 3 common notes. This breaks down to roughly $\frac{1}{8}$, $\frac{3}{8}$, $\frac{3}{8}$ and $\frac{1}{8}$ symbols with 0, 1, 2 and 3 common notes, for a sum per state of 6144 common notes. $Model_1$ is slightly more discriminative, initially, than $Model_0$, and should yield better retrieval results.

An example subset of initial output symbol probability matrix, $P(o|s)$, can be found in Table 4.3. Again, in the interest of space, we do not add ε, nor do we normalize by the sum across the entire state. This sum is the total of all common notes across the entire state (6144) plus the sum total of all ϵ which have been added to each value in the entire row (4095ε). Thus, to recover the actual initial probability for, say, the observation 001001000100 given a $D\ Minor$ triad, we have, for $\varepsilon = 0.00001$:

$$P(001001000100 \mid D\ Minor) = \frac{3+\varepsilon}{6144+4095\varepsilon} \approx 4.883 \times 10^{-4}$$

For another observation, 001001000010, with two overlaps, the probability is:

$$P(001001000010 \mid D\ Minor) = \frac{2+\varepsilon}{6144+4095\varepsilon} \approx 3.255 \times 10^{-4}$$

An observation, 100000000100, with one overlap is:

$$P(100000000100 \mid D\ Minor) = \frac{1+\varepsilon}{6144+4095\varepsilon} \approx 1.628 \times 10^{-4}$$

And finally, an observation, 000010001001, with no notes in common with the state looks like:

$$P(000010001001 \mid D\ Minor) = \frac{0+\varepsilon}{6144+4095\varepsilon} \approx 1.636 \times 10^{-9}$$

4.4 Model Estimation

Though it is one of the more difficult basic problems facing the creation and usage of HMMs, reestimation of model parameters has a number of solutions. The goal is to adjust π, A, and B in a manner so as to "maximize the probability of the observation sequence, given the model" [95]. There is no closed form solution to the problem of a globally optimal set of parameters, so we instead turn to a standard technique known as Baum-Welch, a type of Expectation-Maximization. This is an iterative technique which produces locally optimal parameter settings. Therefore in the previous section we have attempted to set our initial parameters in a manner such that the local maximum found is close to the global maximum. The optimization surface is quite complex, however, and so we have no way of verifying these parameters, by themselves. Instead, we validate them by their performance on the task to which we apply them: ad hoc music retrieval. This will be covered in Chapter 6.

The Baum-Welch parameter reestimation algorithm proceeds in two stages. In the first stage we compute, using the current model parameters and given observation sequence, the probability of

being in state s_i at time t. If we then sum over all values of t (the entire length of the sequence), we can compute a number of different *expected* values. For example, summing over t on the probability of being in s_i gives us the expected number of state transitions from s_i. Summing over t on the probability of being in s_i at time t and in state s_j at time $t+1$ yields the expected number of transitions from s_i to s_j.

With this knowledge in hand, we then proceed to the second stage, where these expected values are used to reestimate the model parameters, *maximizing* the likelihood of the observation. The reestimate $\bar{\pi}_i$ is simply the expected number of times in state s_i at the beginning of the sequence ($t = 1$). The reestimate $\bar{A}_{i,j}$ is the expected number of times going from state s_i to state s_j, normalized by the total (expected) number of outgoing transitions (to all states including s_j) from state s_i. Finally, the reestimate $\bar{B}_{i,l}$ is the expected number of times in state s_i while also observing the symbol k_l, normalized by the total (expected) number of times in state s_i.

The two stages are linked. The expected value for the state transitions depends on the current (either previously reestimated or initialized) parameters setting of the model, and the parameter reestimates then depend on the expected value for the state transitions. Having good initial estimates can be an important factor in learning the correct transition structure. Moreover, the learning algorithm provides an integrated framework in which the state transition probabilities [A] are learned concurrently with the observation probabilities [B]. They are not considered independently of each other, as the reestimate for one will affect the expected values computed for the other.

The tightly coupled relationship between A and B can be advantageous, particularly because training can occur without labeled data (observations which have been tagged with their corresponding latent variables, the states). Hand-labeling can be an expensive procedure and it is useful to avoid it. However, we feel that for our immediate task, ad hoc (query-based) music information retrieval, this coupling can reduce the overall effectiveness of the algorithm. Estimation of a model is the problem of "optimiz[ing] the model parameters so as to best describe how a given observation sequence comes about" [95]. The goal of our retrieval system is to be able to find variations on a piece of music, whether "real-world composed" variations or "audio-degraded originals" variations. When reestimating A and B, those parameters get values which best describe the current observation sequence. They do not get values which best describe hitherto unknown observation sequences which might be (relevant) variations of the current observation sequence.

One would hope that the probabilistic nature of the hidden Markov model could account for this. As we will see through the evaluation in Chapter 6, this is sometimes the case, though not always. Therefore, we will address this issue by introducing another model in Chapter 5 in which the state-to-state and the state-to-observation processes are decoupled.

4.5 Scoring Function - Query Likelihood

Now that we have estimated an HMM for every piece of music in the collection, we turn to the problem of ranking these pieces by their similarity to the query (see Figure 4.1). As stated in Section 1.1, the conceptual framework under which we are operating is that of *evocativeness*. Once we have captured the statistical regularities of a collection of music, through the process of creating probabilistic model of each piece, we may then rank those models by their probability of generating (or *evoking*) a music query.

Fortunately, there exists an algorithm which is part of the standard suite of HMM algorithms which solves the problem of computing an observation generation probability. The observation in this case is just the raw query itself, after the preprocessing stages in which the query, in whatever form it originally existed, is recast as a sequence of simultaneities. Each HMM in the collection has an observation symbol distribution [B] which ties states in the model to the actual observations one might see in the query. Each HMM also has an initial state [π] and state transition [A] distribution, which accounts for the sequence of states. With these distributions in hand, we then use the Forward algorithm to determine the probability of a particular HMM having generated the query observation sequence.

4.5.1 Forward Algorithm

We do not give a full explanation of the Forward algorithm here. Readers are again referred to the tutorial by Rabiner [95]. However, a short explanation is in order. Because we have assumed independence between the observation symbols, we may break down the probability of a query observation sequence O $= o_1, o_2, \ldots, o_T$, given an estimated model of a piece of music from the collection, $\widehat{M}_D$, into the following terms:

$$P(O|\widehat{M}_D) = \sum_{all\ Q_i} P(O|Q_i, \widehat{M}_D) P(Q_i|\widehat{M}_D) \tag{4.1}$$

Again, Q_i is a sequence of states, $q_1, q_2, \ldots, q_T$, equal in length to the observation sequence. The reader will notice a slight shift of notation from Section 4.2, where a sequence of states was referred to as X $= x_1, x_2, \ldots, x_T$. The reason for this shift is that we wish to emphasize that while X is one particular sequence, Q_i is one of all possible state sequences.

Now, with this factorization, we can use our state sequence distributions [π] and [A] to compute $P(Q_i,\ \widehat{M}_D)$, and our observation symbol distribution [B] to compute $P(O \mid Q_i,\ \widehat{M}_D)$, keeping in mind that the two distributions work together. For example, we have:

$$P(O|Q_i, \widehat{M}_D) P(Q_i, \widehat{M}_D) = \pi_1 B_{q_1,o_1} A_{q_1,q_2} B_{q_2,o_2} \ldots A_{q_{T-1},q_T} B_{q_T,o_T} \tag{4.2}$$

In other words, for a particular state sequence Q_i, $P(O \mid \widehat{M}_D)$ is the product of the probabilities of starting in state q_1 and generating observation o_1, going from state q_1 to q_2 and generating observation o_2, and so on along the entire sequence of states and observations.

One major problem is that, since we are summing over all possible sequences of states, Q_i, there are actually exponentially many sequences (on the order of N^t, the number of states to the power of the length of the sequence). *Dynamic programming*, using the idea of *memoization* pares this down to an order N^2t process. Essentially, sequences of states which share common initial subsequence may also share the probability values computed over those common subsequences.

For example, consider two arbitrary sequences of states of length 7: $Q_a = q_5q_9q_3q_3q_6q_8q_2$ and $Q_b = q_5q_9q_3q_3q_6q_8q_7$. Both share the initial length six subsequence $q_5q_9q_3q_3q_6q_8$. Therefore the probability for each of the two sequences of starting in certain states, generating observations, and transitioning through the sequence of states is going to be the same, up to the sixth timestep. We can use this fact and cache that probability the first time it is computed, say during the processing of the first of the two sequences. When it comes time to compute the probability of the second sequence, we look up the value in the cache for the shared subsequence, rather than recomputing it. A storage array of size $O(N^2)$ is required, but it does change the time complexity of the entire algorithm from exponential to a low-degree polynomial.

4.5.2 Ranking

Once the probability of generating the query observation sequence is computed from every model $\widehat{M}_{D_1} \ldots \widehat{M}_{D_C}$ in the collection, the models (and thus the original pieces) are then ranked by this probability. That model with the highest probability has the highest likelihood of having generated the query observation sequence, and therefore is taken to be the most relevant to the query.

We must note that another group of authors who have used HMMs as the basis of their retrieval system (modeling, scoring, and ranking) detected bias in the Forward algorithm, which they claim was the result of their model topology and their [π] initial distributions [110]. Their topology is such that there are large number of "illegal", or forever zero-probability, transitions in the state distribution [A]. As we do not have these same restrictions, and any state should, in principle, be reachable from any other state, we believe that there is not any bias in our particular application of the Forward algorithm for scoring and ranking. Therefore we use the Forward algorithm as is, with no modification. Another way of stating this is that if the Forward algorithm does indeed suffer from a bias, based on the topology of the model or on anything else, all models in the collection share the same exact bias, and thus the relative ranking does not change. This hearkens back to some of

our original motivations for this work, based on ideas from Borges, at the beginning of Chapter 1. Ultimately, it is the relative ranking we are interested in, and not the actual calculated probability.

CHAPTER 5

HARMONIC MODELS

Recall from Chapter 1 that a language model is "a probability distribution over strings in a finite alphabet". In the previous chapter, we used hidden Markov models to infer from observable polyphonic note sequences a distribution over sequences of (hidden state) triads. In this chapter, we instead *decouple* the hidden state inference process from the hidden state sequence (state transition) distribution process by (1) creating our own heuristic mapping function from notes to triads and then (2) using these triad "observations" to estimate a standard Markov model of the state sequence distribution. To these entities we give the name *harmonic models*.

It is our feeling that hidden Markov models have the problem of focusing too much on estimating parameters which maximize the likelihood of the training data, at the cost of overfitting. As a separate model is estimated for every piece of music in the collection, no one model has enough training data, and so overfitting is inevitable. Harmonic models, on the other hand "spread" the available observation data around, and to do so in a manner such that the models estimated from these "smoothed observations" do not overfit the training data as much.

In other words, rather than creating probabilistic models of music in which we seek to discover the "true" harmonic entities for each individual piece of music, as in the HMM case, we instead create probabilistic models in which our estimates for the harmonic entities are somewhat "wider", and thus (hopefully) closer to an a priori unknown variation on that piece. By separating triad discovery from triad sequence modeling we believe we are able to gain more control over the entire modeling process, especially as it relates to the ad hoc information retrieval task.

We also show that harmonic models may be improved over their original estimates by using structural and domain knowledge to further heuristically temper the state observation function. However, Markov modeling is still used to string together chains of states, letting the statistical regularities of the (albeit heuristically estimated) state-chain frequencies speak for themselves, as per the language modeling approach. Furthermore, the time and space complexity for harmonic models is less than for hidden Markov models. We will give a detailed comparison of the two methods, outlining strengths and problems, in Chapter 6. We consider the methods developed in this chapter to be the core of the dissertation.

5.1 System Overview

The process of transforming polyphonic music into harmonic models divides into two stages. In the first stage, the music piece to be modeled is broken up into sequences of simultaneities (see Section 3.1). Each of these simultaneities is fit to a chord-based *partial observation* vector, which we name the *harmonic description*. Each simultaneity and its corresponding partial observation vector is initially assumed to be distinct from the other simultaneities in the piece. However, this assumption is not always accurate, in particular because harmonies are often defined by their context. The harmonic description process is therefore modified with a *smoothing* procedure, designed to account for this context.

The second stage is the method by which *Markov models* are created from the smoothed harmonic descriptions. As part of this stage, estimates of zero probability are adjusted through the process of *shrinkage*. It should be stressed that our methods do not seek to produce a formal music-theoretical harmonic analysis of a score, but merely to estimate a model for patterns of harmonic partial observations which are hoped to be characteristic of the broader harmonic scope of that score.

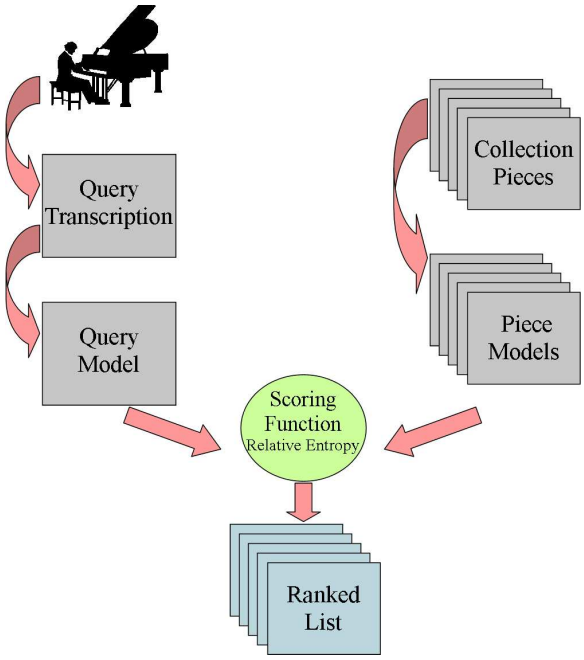


Figure 5.1. Overview of harmonic model-based retrieval system

As with the HMM system described in Section 4.1, we estimate a model for every piece of music in the collection. However, we also estimate a model for the query. Then, using conditional relative entropy, a special form of risk minimization, we rank the pieces in the collection by their model's dissimilarity to the query model.

5.2 Harmonic Description

This system has as its foundation a method for polyphonic music retrieval which begins by preprocessing a music score to describe and characterize its underlying harmonic structure. The output of this analysis is a *partial observation* vector over all chords, one vector for each simultaneity occurring in the score. By partial observation, we simply mean that instead of recording one full observation of some particular chord for a given simultaneity, we break that observation down into multiple proportional or fractional observations of many chords. Thus, instead of doing a chord reduction, extracting (observing) one C-major chord and no other chords from some particular simultaneity, we instead might extract $6/10^{ths}$ of a C-major chord, $3/10^{ths}$ of an A-minor chord, and $1/10^{ths}$ of an F-major chord. The vector of all partial observations of every chord in the lexicon, for some particular simultaneity, sums to one. Rather than choosing a single chord at each time step and using that as the full observation, we allocate a partial observation, no matter how small, to each chord in the lexicon.

We define *harmonic description* as the process of fitting simultaneities to lexical chords in a manner proportional to each chord's influence within the context of a simultaneity. A number of researchers have focused on the harmonic description task. However, most of these authors only extract a single, most salient chord at every time step [93, 29, 9, 85]. The difference in our technique is that we assume *all* chords describe the music, to varying degrees. The purpose of the harmonic description is to determine to what extent each chord fits. But no chord is eliminated completely, no matter how unlikely. We know of two other approaches which do this, but none which have been as specifically applied to the music IR task [94, 113].

So while we are not the first in the music information retrieval community to suggest using chords as shallow structural features, we are the first that we know of to use partial observations rather than reductions. The use of all chords rather than a single chord reduction, to characterize a polyphonic music sequence, is a distinctive trait of our system.

5.2.1 Fitting Simultaneities to Partial Observation Vectors

The first phase in the harmonic description begins with data preparation. We follow the same (optional) polyphonic audio transcription and simultaneity creation procedures described in Section 3.1. Next, we map each simultaneity s in a piece of music, onto a *partial observation* vector.

Harmony, as musicians perceive it, is a highly contextual phenomenon which depends on many factors. It is common for non-chord tones (that is, notes that do not belong to the prevailing harmony) to occur in a simultaneity. It is also common for harmonic notes (that is, notes that do belong to the prevailing harmony) not to occur in a simultaneity. We need a technique for lowering the partial observation allocated to those lexical chords in which non-harmonic notes occur (without eliminating them completely), while raising the allocation to those lexical chords in which harmonic notes do not occur.

In the first step toward this goal we define the following function, which takes into account not only the size of the overlap between a simultaneity and one lexical chord, but also the total number of notes in the simultaneity as well as the total number of notes in the entire chord lexicon.

$$Context(s, c) = \frac{|s \cap c|}{|s|} \sum_{p \in lexicon} \frac{|s \cap p|}{|s| + 1} \tag{5.1}$$

Equation 5.1 is an interesting function. Essentially, there are two terms in the product, $\frac{|s \cap c|}{|s|}$ and $\sum_{p \in lexicon} \frac{|s \cap p|}{|s|+1}$. The purpose of the first term is to count the number of notes the simultaneity and the current chord c have in common. This is downweighted by the size of the simultaneity, so that simultaneities with many notes do not attach too much importance to seeing any one note. The value $|s \cap c|$ runs from 1 to 3, and $|s|$ runs from 1 to 12. The purpose of the second term, on the other hand, is to establish an initial observation "mass", or how much weight or importance we give to any note that occurs in a simultaneity. As more and more notes are found in the simultaneity, this weight increases, but at a decreasing rate. When we put the two terms together into our overall context score, we obtain the following values, found in Table 5.1:

	$\|s \cap c\|$		
$\|s\|$	1	2	3
1	3	–	–
2	2	4	–
3	1.5	3	4.5
4	1.2	2.4	3.6
5	1	2	3
6	0.86	1.72	2.58
7	0.75	1.5	2.25
8	0.67	1.33	2
9	0.6	1.2	1.6
10	0.545	1.09	1.64
11	0.5	1	1.5
12	0.46	0.92	1.39

Table 5.1. Range of the function $\frac{|s \cap c|}{|s|} \sum_{p \in lexicon} \frac{|s \cap p|}{|s|+1}$

This function has a nice pyramid shape, where the highest score for some chord c is given for simultaneities with exactly three notes, which also exactly match that lexical chord c. The score

goes down when a simultaneity contains all three notes of c, but also contains extra notes as well. The score goes down if the simultaneity contains exactly three notes, but only two or one of those notes match c. The score goes down if the simultaneity contains fewer than three notes, even if all existing notes match c.

This context score is computed for every chord c in the lexicon. To obtain the vector of partial observations, we normalize by the sum:

$$PartialObservation(s, c) = \frac{Context(s, c)}{\sum_{q \in lexicon} Context(s, q)} \tag{5.2}$$

In addition to providing good estimates for the harmonic description, this context score also accounts for more complex input chords, such as 7^{th} chords, by retaining higher partial observation for those nearby triads in which a 7^{th} chord "participates". For example, if the simultaneity contained the notes [A-C-E-G], both the A-minor and the C-major chords would get substantial partial observations.

Currently, this harmonic reallocation is performed within the scope of a single simultaneity, not over time. In the next section. we extend the harmonic context to take into account the profile of neighboring simultaneities.

5.3 Smoothing of the Harmonic Description

While the method above takes into account the presence of all chords in the lexicon, it only does so within the current simultaneity, the current timestep. The astute observer would realize that, because only the values in the current simultaneity are used, the partial observation function, Equation 5.2, is equivalent to $\frac{|s \cap c|}{\sum_{q \in lexicon} |s \cap q|}$. Clearly, we did not need to go through all the trouble of creating Equation 5.1 if we were only going to compute a context score within the current simultaneity. In this section we explain how we move beyond a single simultaneity.

Harmony, as musicians perceive it, is a highly contextual phenomenon which depends not only on the harmonic distances at the current timestep, but is also influenced by previous timesteps: the harmonies present in the recent past are assumed to be a good indication of the current harmony. Thus, a simultaneity with only one note might provide a relatively flat partial observation vector, but when that simultaneity is taken in historical context the description becomes more accurate.

We have developed a simple technique for taking into account this event-based context by examining a window of n simultaneities and using the values in that window to give a better estimate for the current simultaneity. This is given by the following equation, where s_t is the simultaneity at timestep t:

$$SmoothedContext(s_t, c) = \sum_{i=1}^{n} \frac{1}{i} \left(\frac{|s_{t-i+1} \cap c|}{|s_{t-i+1}|} \sum_{p \in lexicon} \frac{|s_{t-i+1} \cap p|}{|s_{t-i+1}| + 1} \right) \tag{5.3}$$

When the smoothing window n is equal to 1, this is equivalent to Equation 5.1. When n is greater than 1, the score for the lexical chord c at the current timestep is influenced by previous timesteps in proportion to the distance (number of events) between the current and previous timestep. As in the unsmoothed ($n = 1$) version, the smoothed context score is computed for every chord c in the lexicon and then the entire vector is normalized by the sum, yielding the partial observations (see Equation 5.2).

Now, because we are taking multiple simultaneites into account, and feeding all of their context scores into the score for the current simultaneity, the magnitude of those scores matter. For example, as we can see from Table 5.1, it does not only matter that a simultaneity at the previous time step has three notes in common with the lexical chord c. If there were five notes in that previous simultaneity, the value contributed to the context score of c by the same chord at the previous time step is $\frac{1}{2} * 3 = 1.5$. However, if there were only two notes in the previous simultaneity, but both notes are "members" of c, then the amount contributed would be $\frac{1}{2} * 4 = 2$. So it does not just matter what notes are present; it matters what other notes are present as well.

For more discussion on the harmonic description, as well as errata, we invite the reader to examine Appendix A on page 99.

5.4 Markov Modeling

Once the harmonic description is complete, we create visible Markov models using fixed-length Markov chains. Markov models are often used to capture statistical properties of state sequences. We want to be able to predict the current occurrence of a state using the value of the previous state. As with the hidden Markov models of the previous chapter, the states of the model are the 24 lexical chords introduced in Chapter 3:

$$\{s_1, \ldots s_N\} \quad \text{the size N set of states (N=24)}$$

Next, the following probability distributions involving these states are needed:

$$\begin{array}{ll} \pi_i & \text{the probability of starting a sequence in state } s_i \\ A_{i,j} & \text{the probability of transitioning from state } s_i \text{ to state } s_j \end{array}$$

The initial state distribution, π, is a distribution over the starting state of the sequence. The state transition probability matrix, A, is a distribution over how likely it is that we transition into some current state, given the previous state. For hidden Markov models, we only dealt with 1^{st}-order models. However, the lack of an observation symbol distribution in standard Markov models makes it much easier and more efficient to construct higher-order models. One need simply examine a larger window of previous states to assist in constructing a distribution over current states.

For higher-order models, the previous state space is implemented by taking an n-dimensional cross of the state space, $\{s_1, \ldots s_N\}$. Thus, for an n^{th}-order model, a $24^n \times 24$ matrix is constructed. 24^n elements represent the previous state, and 24 elements represent the current state. To avoid confusion, instead of writing A_i, j, with s_j as the current state and s_i as the previous state, we will write the state transition distribution as $P(chord_x|history_y)$. $Chord_x$ is shorthand for the current state, and $history_y$ is shorthand for the previous state.

5.4.1 Model Estimation

The main problem we now face is how to estimate a Markov model from the harmonic description sequence created in the previous sections. Two parameters are needed: [A] and [π]. A standard approach on observable-state data is to use the relative frequency counts over the data. This is also known as the maximum likelihood estimate, because the parameters of the model, M, are set such that the probability of the data given the model, $p(data|M)$ is maximized.

For example, with 1^{st}-order (bigram) Markov model consisting of three states P, Q, and R, and the observation data PRRPQPPRQ, the maximum likelihood estimate for $A_{P,R} = p(R|P)$ is one half. In the data, there are instances of exactly four bigrams in which the state P is followed by another state: $P \rightarrow R$, $P \rightarrow Q$, $P \rightarrow P$ and $P \rightarrow R$. Of these four observations, P is followed by R exactly twice. Thus, the parameter setting for $p(R|P)$ that makes the data look the most likely is $\frac{2}{4}$. Furthermore, the maximum likelihood estimate for π_P, the prior or initial state probability for P, is the number of times P occurs at the beginning of the sequence: $\pi_P = p(P)$ is 1.

There is a problem, however, in that our data from Sections 5.2 and 5.3 does not consist of one-dimensional sequences of observations. At a given time step, we did not reduce a simultaneity to a single chord (a *full observation*). Instead, we created a harmonic description, a partial observation vector of chords. The solution, then, is to do a maximum likelihood count using these partial rather than full observations

Unfortunately, at every time step (at every simultaneity or harmonic description vector) in a piece of music, there are as many partial observation state-chains as parameters in the model. In other words, at every time step, there is not a single observation or state-chain; there are 24^{n+1} of them, where n is the order of the model. This is an exponential value, but fortunately n is fixed.

The algorithm proceeds by counting the partial observation frequency for a particular chord given a particular history, by summing the partial observations across all time steps i in the sequence. We note that this algorithm is abstractly similar to an approach suggested by Doraisamy [40].

$$\text{Frequency of } chord_x \text{ following } history_y = \sum_{i=2}^{n} history_y(i-1) * chord_x(i) \tag{5.4}$$

We repeat this count for all possible history/chord state combinations. The resulting counts are the partial observation maximum likelihood counts. Finally, when the entire observable sequence has been counted, the values of each history state are summed and the elements of that row normalized by the sum for that row. The counts become conditional probability estimates. Thus, the estimated state transition $[A_{x,y}]$ probability for chord x from the lexicon, conditioned on the order-n history y, are:

$$\hat{p}(chord_x|history_y, M) = \frac{\text{Frequency of } chord_x \text{ following } history_y}{\text{Frequency of all chords following } history_y} \tag{5.5}$$

We note that this estimation procedures is similar to that of the Baum-Welch state transition reestimation procedure for hidden Markov models. The main difference is that we are not summing over probabilities or expectations of probabilities; we are summing over heuristically constructed partial observations. Furthermore, observations are final. There is no reestimation procedure which revisits the distribution and alters its values.

Additionally, we need a prior probability π on the starting state of the sequence. Recall from the previous (hidden Markov model) chapter that the assumption was made that, prior to encountering an actual piece of music, we have no reason to believe, that our model will start out in any particular state. We make the same assumption here; initially, we are equally likely to start off in any state:

$$\pi_i = \frac{1}{\mathrm{N}} = \frac{1}{24} \tag{5.6}$$

However, one of the main differences here is that, unlike with hidden Markov models, we do not reestimate π. It remains $\frac{1}{24}$. In the future, it might be useful to explore other approximations of π. For example, if we know that a piece of music is written in a certain key, we might have reason for a non-uniform initial state distribution.

5.4.2 Model Estimation Example

Figure 5.2 is an example of the output from the harmonic description phase; it is a sequence of partial observation vectors for a very short piece of music with only 5 simultaneities. For the purpose of this example, we assume a lexicon of only three chords, to which we arbitrarily assign the names P, Q, and R. We use this harmonic description to estimate a number of models of various orders.

Lexical	Timestep (Simultaneity)				
Chord	1	2	3	4	5
P	0.2	0.1	0.7	0.5	0
Q	0.5	0.1	0.1	0.5	0.1
R	0.3	0.8	0.2	0	0.9
Total	1.0	1.0	1.0	1.0	1.0

Figure 5.2. Example partial observation vector sequence

5.4.2.1 1^{st}-order Model

With this sequence of partial observation vectors in hand, we may obtain the counts of all (bigram = 1^{st}-order) Markov chains in the entire sequence. We begin with the window from timestep 1 to

timestep 2. The sequence $P \rightarrow P$ is observed in proportion to the amount in which we observe P at timestep 1 and also observe P at timestep 2 (0.2 * 0.1 = 0.02). The sequence $Q \rightarrow R$ is observed in proportion to the amount in which we observe P at timestep 1 and then Q at timestep 2 (0.5 * 0.8 = 0.4), and so on. The entire timestep 1 to timestep 2 window is illustrated in Figure 5.3.

$P \rightarrow P$	=	0.2 * 0.1	=	0.02
$P \rightarrow Q$	=	0.2 * 0.1	=	0.02
$P \rightarrow R$	=	0.2 * 0.8	=	0.16
$Q \rightarrow P$	=	0.5 * 0.1	=	0.05
$Q \rightarrow Q$	=	0.5 * 0.1	=	0.05
$Q \rightarrow R$	=	0.5 * 0.8	=	0.4
$R \rightarrow P$	=	0.3 * 0.1	=	0.03
$R \rightarrow Q$	=	0.3 * 0.1	=	0.03
$R \rightarrow R$	=	0.3 * 0.8	=	0.24
		TOTAL	=	1.0

Figure 5.3. Example 1^{st}-order (bigram) Markov chain partial observations, from timestep 1 to timestep 2

It should be emphasized that these values are not the probabilities of the final model at this stage; they are counts. Perhaps the term *scaled counts* might be more appropriate. Think of it this way: Suppose 100 musicians were each given an instrument which was only capable of playing one lexical chord at a time (such as an autoharp), and not arbitrary individual notes. Suppose further that these musicans were given sheet music of the documents which we are modeling, which documents contain only individual notes and not chords. The musicians would be forced to make a choice about which lexical chord to play. We are saying that 2 of the musicians would then choose to play $P \rightarrow P$, 40 would play $Q \rightarrow R$, and so on. Our actual observation of what lexical chords were played includes every single one of these possibilities, concurrently and independently. It is as if this one snippet of music were played 100 times, 9 different ways, and we simply count the number of times each way was played. Divide those integer counts by 100 and the proportions, and thus the final model, remains the same.

We add these transition counts from Figure 5.3 to the count matrix (which was initialized to all zeros) and repeat the process for timesteps $2 \rightarrow 3$, $3 \rightarrow 4$, and $4 \rightarrow 5$ (the remainder of the piece). When finished, we follow the standard method of transforming the matrix into a Markov model by normalizing the outgoing arcs from each previous state, so that the total scores sum to 1.0, as shown in Figure 5.4. Finally, π is, of course, $\frac{1}{3}$ for each of the three states, P, Q and R.

Counts			
	P	Q	R
P	0.44	0.43	0.63
Q	0.17	0.16	0.87
R	0.69	0.21	0.40

[A] distribution				
	P	Q	R	Total
P	0.293	0.287	0.42	1.0
Q	0.142	0.133	0.725	1.0
R	0.531	0.161	0.308	1.0

Figure 5.4. Example 1^{st}-order state transition counts (*left*) and normalized state transition distribution [A] (*right*)

5.4.2.2 2^{nd}-order Model

Counting the state transitions and initial states in a 2^{nd}-order model is not that much different than in a 1^{st}-order model. Recall that the history state in a n^{th}-order model is simply the cross of the previous n states. Whereas in a 1^{st}-order model the previous states are P, Q, and R, in a 2^{nd}-order model the previous states are PP, PQ, PR, QP, QQ, QR, and RP, RQ, RR. We also need

$PP \rightarrow P$	=	0.2 * 0.1 * 0.7	=	0.014
$PP \rightarrow Q$	=	0.2 * 0.1 * 0.1	=	0.002
$PP \rightarrow R$	=	0.2 * 0.1 * 0.2	=	0.004
$PQ \rightarrow P$	=	0.2 * 0.1 * 0.7	=	0.014
$PQ \rightarrow Q$	=	0.2 * 0.1 * 0.1	=	0.002
$PQ \rightarrow R$	=	0.2 * 0.1 * 0.2	=	0.004
$PR \rightarrow P$	=	0.2 * 0.8 * 0.7	=	0.112
$PR \rightarrow Q$	=	0.2 * 0.8 * 0.1	=	0.016
$PR \rightarrow R$	=	0.2 * 0.8 * 0.2	=	0.032
$QP \rightarrow P$	=	0.5 * 0.1 * 0.7	=	0.035
$QP \rightarrow Q$	=	0.5 * 0.1 * 0.1	=	0.005
$QP \rightarrow R$	=	0.5 * 0.1 * 0.2	=	0.01
$QQ \rightarrow P$	=	0.5 * 0.1 * 0.7	=	0.035
$QQ \rightarrow Q$	=	0.5 * 0.1 * 0.1	=	0.005
$QQ \rightarrow R$	=	0.5 * 0.1 * 0.2	=	0.01
$QR \rightarrow P$	=	0.5 * 0.8 * 0.7	=	0.28
$QR \rightarrow Q$	=	0.5 * 0.8 * 0.1	=	0.04
$QR \rightarrow R$	=	0.5 * 0.8 * 0.2	=	0.08
$RP \rightarrow P$	=	0.3 * 0.1 * 0.7	=	0.021
$RP \rightarrow Q$	=	0.3 * 0.1 * 0.1	=	0.003
$RP \rightarrow R$	=	0.3 * 0.1 * 0.2	=	0.006
$RQ \rightarrow P$	=	0.3 * 0.1 * 0.7	=	0.021
$RQ \rightarrow Q$	=	0.3 * 0.1 * 0.1	=	0.003
$RQ \rightarrow R$	=	0.3 * 0.1 * 0.2	=	0.006
$RR \rightarrow P$	=	0.3 * 0.8 * 0.7	=	0.168
$RR \rightarrow Q$	=	0.3 * 0.8 * 0.1	=	0.024
$RR \rightarrow R$	=	0.3 * 0.8 * 0.2	=	0.048
		TOTAL	=	1.0

Figure 5.5. Example 2^{nd}-order (trigram) Markov chain partial observations, from timestep 1 to timestep 3

more contiguous time steps in which to observe a partial Markov chain count, as it is by definition that higher-order models require more history. Thus, we may obtain the counts of the (trigram = 2^{nd}-order) Markov chains in the entire sequence starting with the window from timestep 1 to timestep 3. This is illustrated in Figure 5.5.

The state transition matrices and initial state distribution matrices are creating in the same manner as in Section 5.4.2.1, by summing state transition counts and state history counts across the entire piece of music. We do not repeat these examples here.

5.4.2.3 0^{th}-order Model

It is also possible to create 0^{th}-order models. These models have no history, no previous states; there are only current states. Or, expressed in another manner, the history in a 0^{th}-order model is always *null* $(\cdot)$, and the null-history is always present. We observe the null-history in full (count of 1.0) at any and every point in the sequence. Figure 5.6 shows the counts for timestep 1.

As with previous examples, we then sum the partial observation counts across all simultaneities in a piece of music. It should be clear that the resulting state transition distribution [A] is simply the average of the partial observation counts. The initial state distribution $[\pi]$ has a single point, the null point, and the probability of that point is 1.0.

$\cdot \rightarrow P$	=	1.0 * 0.2	=	0.2
$\cdot \rightarrow Q$	=	1.0 * 0.5	=	0.5
$\cdot \rightarrow R$	=	1.0 * 0.3	=	0.3
		TOTAL	=	1.0

Figure 5.6. Example 0^{th}-order (unigram) Markov chain partial observations, from timestep 1 to timestep 1

5.5 Scoring Function - Relative Entropy

As mentioned in Section 1.1.4, we are taking the language modeling approach to information (music) retrieval. This means that we have made the assumption that O_Q, a user query statement, is actually a sample from a generative model, $p(O_Q|M_Q)$. A piece of music O_D from the collection is also assumed to be generated by a model $p(O_D|M_D)$. The unknown parameters for each model are M_Q and M_D, and the estimates for those models, from Equations 5.5 and 5.6, are $\widehat{M}_Q$ and $\widehat{M}_D$, respectively.

In the Ponte work on this subject [92], language models are inferred for every document in a collection and then documents are ranked by their probability of generating a query statement (query likelihood). The only unknown parameter is M_D. This was also the approach taken for hidden Markov models in the previous chapter. For harmonic models, we instead take the Zhai and Lafferty approach [121], wherein language models are estimated for both document and query and then comparisons are made by how well one *model* ($\widehat{M}_D$) approximates (evokes) the other *model* ($\widehat{M}_Q$). Thus, we are still taking the language modeling approach to information retrieval, even if we do not rank by the probability of a model generating a query observation. The point is that in neither the query or the music document do we attempt to discover what the music is "about". For both query and document, the purpose of modeling is to capture the statistical regularities of a piece of music, and let those regularities speak for themselves.

Our goal is to produce a ranked list for a query across the collection. We wish to rank those pieces of music at the top which are most similar to the query, and those pieces at the bottom which are least similar. In order to do this we need a scoring function which compares models directly. In Section 5.5.1 we will explore one method for comparing hidden Markov models directly, in order to establish a precedent. In Section 5.5.2 we will explain some of the problems with this approach, and then offer an alternative scoring function based on conditional relative entropy.

5.5.1 Divergence Scoring Function for HMMs

The approach taken in Chapter 4 for music document ranking was query likelihood. Pieces in the collection were ranked by the likelihood that their associated models could have produced the query observation. Rabiner [95] outlines another possible method for determining similarity. Suppose that instead of a model and an observation, one now has two models, $\mu_1 = (\pi_1, A_1, B_1)$ and $\mu_2 = (\pi_2, A_2, B_2)$. Though the models themselves might have very different probability distributions, they could have very similar likelihoods of producing observation sequences. Thus, one way of measuring the similarity of two models to each other is to measure how similar they are in what they generate. This is done with the following formula, which defines a distance $D(\mu_1, \mu_2)$ between models μ_1 and μ_2:

$$D(\mu_1, \mu_2) = \frac{1}{t} \left[\log P(O_2|\mu_1) - \log P(O_2|\mu_2)\right] \tag{5.7}$$

$O_2 = \{o_1, o_2, o_3, \ldots, o_t\}$ is a sequence of observations generated by μ_2. So $D(\mu_1, \mu_2)$ "is a measure of how well model μ_1 matches observations generated by model μ_2, relative to how well model μ_2 matches observations generated by itself [95]." Interpretations of this measure include cross-entropy, or divergence.

If we were to apply this to music information retrieval, we would, in addition to estimating a model for every piece of music in the collection, also estimate a model for the query itself. We could then either generate a number of observations by sampling from the query model, or else use the original observation from which the query model was estimated (the query itself). But in either case, we would then measure how well a music piece model matched the query observation against how well the query model matched its own observation. The weaker the relative match, the more dissimilar the query and the music piece, and the lower the overall rank.

5.5.2 Relative Entropy for Harmonic Models

Though the distance metric in the previous section is a useful way for comparing two different hidden Markov models, there are a number of reasons why we chose not to use it. The first reason is that the O_2 is heuristically chosen. What should the length of O_2 be? A longer or shorter observation sequence might favor one model over another. One solution to that is to generate multiple independent observation sequences of various lengths, and then take the average of the model distances calculated across all of them. However, the choice of how many sequences to generate is also heuristic. Furthermore, given the fact that strings may be of "infinite" length, we also rule out the possbility of generating all possible strings of all possible lengths, as this is not only intractable but impossible. While it is true that the exceptionally long strings are going to be less and less likely for both models, and that we could therefore stop generating strings larger than a certain length and obtain a divergence score within ϵ of the true score, the choice of length and therefore ϵ is still a heuristic one.

The second reason we chose not to use this particular scoring function was that as long as the distance metric was going to be heuristic, we wanted to choose a heuristic function with an established precedent in the information retrieval community. As mentioned previously, Zhai and Lafferty [121] take the approach wherein language models are estimated for both document and query and then comparisons are made by how well one model approximates [evokes] another. The distance between models is determined using Kullback-Liebler (KL) divergence (see Equation 5.8).

As the terms in a document are assumed to be generated by a unigram (0^{th}-order) model, this is equivalent to the relative entropy of two probability mass functions over a single random variable. It is a measure of dissimilarity between two conditional distributions over the same event space. Divergence is always zero if two distributions are exactly the same, or a positive value if the distributions differ. Intuitively, it is a measure of "how well distribution q approximates [evokes] distribution p; or, more precisely, how much information is lost if we assume distribution q when the true distribution is p" ([73], page 304) [73]. The general form is found in Equation 5.8.

$$D(p(x) \mid\mid q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{5.8}$$

However, we are dealing with not only unigram harmonic models, but bigram and trigram models as well. Our models are factored into two terms: the initial state or history prior $p(history_y)$, and the state transition conditional distribution, $p(chord_x|history_y)$; for shorthand, we will write $p(y)$ and $p(x|y)$. The value $p(y)p(x|y)$ does not actually define a joint distribution over the current state x and the history state y. Rather, $p(y)\prod_{i=1}^{t} p_i(x|y)$ defines the joint probability over an entire sequence of states of length t. For reasons explained earlier, we consider it impractical and ineffective to generate any number of strings on which to test this joint distribution. Instead, we set forth that two n-gram Markov models may be compared directly by examining their conditional state transition distributions. Our precedent is that this same assumption is made by Stolcke for n-gram language models [114].

The general form of relative entropy for conditional distributions is:

$$D(p(x|y) \mid\mid q(x|y)) = \sum_y p(y) \sum_x p(x|y) \log \frac{p(x|y)}{q(x|y)} \tag{5.9}$$

Thus, to compute the relative entropy of $\widehat{M}_Q$ in the presence of $\widehat{M}_D$ requires summation over all chord states x in the model, weighted by the query model history prior:

$$D(\widehat{M}_Q||\widehat{M}_D) = \sum_y p(y|\widehat{M}_Q) \sum_x p(x|y,\widehat{M}_Q) \log \frac{p(x|y,\widehat{M}_Q)}{p(x|y,\widehat{M}_D)} \tag{5.10}$$

Recall from Equation 5.6 on page 48 that the intial state distribution, $p(y)$, is uniform. Thus, we may drop this term from consideration, as it will not affect the relative rankings, one document to the next. We denote this modified conditional relative entropy as $\breve{D}$:

$$\breve{D}(\widehat{M}_Q||\widehat{M}_D) = \sum_y \sum_x p(x|y,\widehat{M}_Q) \log \frac{p(x|y,\widehat{M}_Q)}{p(x|y,\widehat{M}_D)} \tag{5.11}$$

With $p(y|\widehat{M}_Q)$ gone, this begins to look like standard relative entropy. We may further arrange the terms:

$$\begin{aligned}
\breve{D}(\widehat{M}_Q||\widehat{M}_D) &= \sum_y \sum_x p(x|y,\widehat{M}_Q) \log \frac{p(x|y,\widehat{M}_Q)}{p(x|y,\widehat{M}_D)} \\
&= -\sum_y \sum_x p(x|y,\widehat{M}_Q) \log \frac{p(x|y,\widehat{M}_D)}{p(x|y,\widehat{M}_Q)} \\
&= -\sum_y \sum_x p(x|y,\widehat{M}_Q) \left[\log p(x|y,\widehat{M}_D) + \log \frac{1}{p(x|y,\widehat{M}_Q)} \right] \\
&= -\sum_y \sum_x p(x|y,\widehat{M}_Q) \log p(x|y,\widehat{M}_D) + p(x|y,\widehat{M}_Q) \log \frac{1}{p(x|y,\widehat{M}_Q)}
\end{aligned}$$

The second term in the summation, $p(x|y,\widehat{M}_Q) \log \frac{1}{p(x|y,\widehat{M}_Q)}$, is the pointwise entropy of the query model. Zhai [121] discards this term, as it is also constant across all document models in the collection and thus does not affect ranking. Without this term, the ranking function is essentially the cross-entropy of the query model with respect to the document model. We discard it as well. Thus, our doubly modified conditional relative entropy, denoted $\bar{D}$, is:

$$\bar{D}(\widehat{M}_Q||\widehat{M}_D) = -\sum_y \sum_x p(x|y,\widehat{M}_Q) \log p(x|y,\widehat{M}_D) \tag{5.12}$$

In review, at index time, a model of every piece of music in the collection $\widehat{M}_{D_1} \ldots \widehat{M}_{D_C}$ is estimated and stored. At query time, a query model $\widehat{M}_Q$ is created with exactly the same structure, or model-order, as the collection models. If the collection was modeled using 2^{nd}-order models, the query must be modeled in the same fashion, or else there will be a problem with a differing number of events. Our modified divergence score between a query and every piece in the collection $\bar{D}(\widehat{M}_Q||\widehat{M}_{D_i})$ is computed, and pieces are ranked in ascending order of dissimilarity.

Currently, query models and document models are estimated using the same parameters. Thus, if the documents in a collection are 1^{st}-order models smoothed with two previous onset windows and shrunk by backing off toward a globally estimated music model, then the query model will be estimated in the same manner. (Note: shrinkage will be covered in the next section.) It is entirely possible, however, that the query could be a 1^{st}-order model smoothed with four previous onset windows and shrunk by linear interpolation with a key-specific music model. Exploring this is beyond the scope of this work.

5.6 Shrinkage

There is a problem with using divergence for determining the similarity between a query and a piece of music. The issue is that the document sample d which we assume is generated by some unknown model $p(d|M_D)$, and which we are using to estimate the parameters of our document (music piece) language model $\widehat{M}_D$, suffers from the problem of data sparseness. Especially as the model-order increases, the number of parameters (different Markov chains) that need to be estimated increases, and the amount of data afforded by a single sample d, from M_D, becomes dangerously low. In fact, when computing $\bar{D}(\widehat{M}_Q||\widehat{M}_D)$, the presence of a single zero value estimate for any parameter of $\widehat{M}_D$ will cause $\bar{D}$ to become infinite, essentially throwing off the entire score no matter how good a match the remainder of the distribution.

The fact that our maximum likelihood estimates come from partial observations rather than full observations alleviates this zero value problem to some degree, as the harmonic descriptions spread the observations out onto many more chords, and consequently onto the Markov chains of which these chords are a part. Smoothing the harmonic descriptions, described in Section 5.3, decreases the presence of zero estimates even further. However, many zero estimates still remain. *Shrinkage* is the technique by which these estimates of zero probability are overcome.

Shrinkage works by using data rich states of simpler models to better inform the estimates in data sparse states of more complex models [47]. There are essentially two types of shrinkage: *interpolation* and *backoff* [122]. Interpolation works by "borrowing" some probability mass from every state in the data-rich model and mixing it with the probability mass of its corresponding state in the data-sparse model. With interpolation, probabilities are combined using mixture weights λ_1 and λ_2, where λ_1 and λ_2 are both positive real numbers and sum to 1. The new interpolated model $\widehat{M}_{Dnew}$ is $(\lambda_1 \cdot \widehat{M}_{D_R}(x)) + (\lambda_2 \cdot \widehat{M}_{D_S}(x))$, over all x, where $\widehat{M}_{D_R}(x)$ and $\widehat{M}_{D_S}(x)$ are states or parameters of the the data-rich and data-sparse models, accordingly.

With backoff on the other hand, the idea is to retain confidence in the maximum likelihood estimate for the high frequency data points, only redistributing probability mass for the low frequency points. For example, suppose that the data-sparse model, $\widehat{M}_{D_S}$, has a zero-frequency estimate for one of its points: $\widehat{M}_{D_S}(x) = 0$. We then back off to the data-rich model, and set $\widehat{M}_{D_S}(x) = \widehat{M}_{D_R}(x)$. Finally, in order for $\widehat{M}_{D_S}$ to remain a probability distribution, we must discount (adjust or renormalize) the remaining points equally, to the degree that we increased $\widehat{M}_{D_S}(x)$.

5.6.1 Global Music Backoff

Now that backoff and interpolation have been defined, we may introduce our first shrinkage approach: *global music backoff*. In this approach, the data-rich model is a harmonic model created using the same Markov-order and harmonic description smoothing techniques as the rest of the models in the system. However, instead of estimating the model from a single piece of music in the collection, it is estimated from the entire collection. One may think of this as a background model for music, at least within the scope of the collection we have available to us.

Whenever there is an estimate of zero probability in a music collection document model, we back off to the global music model. In principle, there could still remain zero values in the global music model, depending on the size and properties of the collection. In our experiments we found that this did sometimes happen, though quite rarely.

5.6.2 Key-specific Backoff

Our second shrinkage method takes a slightly different approach. In this case, instead of having a single global music model, we have 24 key-specific models, 12 for each of the major and 12 for each of the minor keys. The intuition is that, depending on the key in which a piece of music was written, there are going to be different estimates for different states. We therefore do not want to update the estimate for a piece written in one key with values determined by pieces in many, perhaps completely unrelated keys. Other researchers have had similar intuitions [123].

An analogy to text modeling may be helpful. Suppose one has models for two different topics: finance and ecology. Both topic models have some non-zero probability estimate for the word "bank". However, the estimate of "bank" under the finance topic might be much higher, because banks and banking is a common feature of financial documents, while stream or river "bank" is an active, though less common, feature of ecological documents. Thus, if you determine that a particular document is about ecology, and not about finance, and this document also contains a zero probability estimate for "bank", you would want to shrink that estimate toward the ecology topic model rather than toward the finance topic model.

By analogy, a key can be though of as the "topic" for a particular piece of music. No matter what the unique melodic or harmonic progression of that piece, the expected frequency of a C major triad is going to be very different if that piece is written in C major, versus if that piece is written in $F^\sharp$ major. Yet if we smooth this piece with a global model, created by averaging every piece in the collection including those written in $F^\sharp$ major, then we will be shrinking the estimates for this piece further away from what the true values might be.

So rather than create a global model, we create 24 "topic" models of music, one for each possible key (C Major, C minor... B Major, B minor). At indexing time, each piece in the collection is tagged by key, and key-specific models are created by collecting the counts for every piece in that key into a single model, creating a key-based average. With these 24 key-specific models in hand, each document model in the collection is then updated by backing off zero frequency values to its key-specific estimate.

As we cannot guarantee that key information is present for every document in the collection, we instead use a naive heuristic key-finding algorithm. We begin by estimating the probability of being in a chord state, for a particular music piece:

$$\hat{p}(chord_x|M_D) = \frac{\text{Partial observation frequency of } chord_x}{\text{Length of the piece of music in Simultaneities}} \tag{5.13}$$

Our key-finding algorithm is simply to select that chord with the highest probability; the root of this triad determines the key/tonic, while the third of this triad determines the modality (major or minor). We do not evaluate the efficacy of this algorithm directly, however. The goal of this work is retrieval, not key-finding. Therefore we will evaluate the key-finding algorithm, in Chapter 6, in terms of whether or not key-specific backoff improves retrieval.

5.6.3 Key-specific Interpolation

As with the key-specific backoff approach, 24 key-specific data rich models are created from the collection as a whole. However, instead of backing off each music document toward its associated key-specific model, we interpolate between the key-specific and the music piece models, regardless of zero value estimates. The parameters of the interpolation are given a maximum entropy value of $\lambda_1 = 0.5$ for the key-specific model and $\lambda_2 = 0.5$ for the data-sparse individual document model. In the future these lambda values could be better set using Expectation-Maximization. However, we will see in Chapter 6 that the current settings still yield significant improvements in the retrieval results.

CHAPTER 6

EVALUATION

In this chapter we wish to provide precise quantitative measures of how well a user's information need is met by the various systems introduced in previous chapters. There are two main goals of these experiments. The first is to determine whether the probabilistic modeling techniques we use are able to capture a credible sense of musical similarity. The user information need is assumed to be along the lines of "find more like this one." We hope that by presenting and then refining a number of different models, we may get a better sense of how to detect realistic variations.

Our second goal is to extend the familiar "query by humming" music retrieval framework into the polyphonic realm. As humming in multiple voices is quite difficult, the task is more accurately described as "query by audio example". To our knowledge, we are the first to use polyphonic audio queries to retrieve from polyphonic symbolic collections.

Our systems can bridge the gap between audio and symbolic music using transcription algorithms together with harmonic modeling techniques. In this manner we allow users to present queries in the audio format and retrieve pieces of music which exist in the symbolic format. This is one of the earliest goals of music retrieval, and until now it has only been possible within the monophonic domain. We extend the realm of possibility into the enormously more difficult polyphonic domain, and show this through successful retrieval experiments for both known-item and variations queries. We believe the ability to use polyphonic audio queries to retrieve pieces of music from a polyphonic symbolic collection is a major step forward in the field. Further examples and commentary on the type of problems we are aiming to solve may be found in Section 1.2.

6.1 Experiment Description

A retrieval experiment which follows the Cranfield model [31] needs the following three components: a source collection which is to be searched, a set of queries, or statements of information needs, and relevance judgements that determine which items in the collection meet the information need of each query. We detail each of these components below, as well as introduce the standard information retrieval measures of Mean Reciprocal Rank (MRR) for queries with a single relevant item, and Recall-Precision for queries with multiple relevant items.

6.1.1 Source Collection

The basic collection on which we test our retrieval methods is assembled from data provided by the Center for Computer Assisted Research in the Humanities (CCARH) at Stanford University [54]. It comprises approximately 3000 files of separate movements from polyphonic, fully-encoded music scores by a number of Baroque and Classical composers, including Bach, Beethoven, Corelli, Händel, Haydn, Mozart, and Vivaldi. The scores have been subdivided into their natural (composed) movements and/or parts, so it is rare that a single piece of music in our collection is a full symphony. These pieces are in all major and minor keys, have a wide variety of textures (related to the numbers of notes in a simultaneity) and are of widely-varying lengths (numbers of simultaneities). The average length of a piece is 460 onsets (simultaneities), with a standard deviation of 355 onsets. The average number of notes in a simultaneity, across the entire collection, is 2.75, with a standard deviation of 1.67. The files were supplied to us in the Humdrum kern format [55], a type of symbolic representation, and were converted into our own internal music-representation format.

A significant proportion of the collection (approximately $\frac{2}{5}$) is composed of the works of J. S. Bach, and included in the collection are the 24 Preludes and 24 Fugues from Book I of the Well-Tempered Clavier. Each Prelude and Fugue is in a different key: 12 major key Preludes and Fugues, and 12 minor key Preludes and Fugues. This is important because we will conduct a number of experiments with these pieces and using this data set demonstrates that our algorithms are not biased toward any one particular key.

To this basic collection we add three additional sets of polyphonic music data: the [T]winkle, [L]achrimae and [F]olia variations. The description of these pieces, the number of variations in each set, and some basic properties of each set, are given below. With these additional pieces, the total collection size is approximately 3,150.

T 26 individual variations on the tune known to English speakers as "Twinkle, twinkle, little star" (in fact a mixture of mostly polyphonic and a few monophonic versions). Half are from Mozart's Variations on "Ah! Vous dirai-je, Maman, K.265"

L 75 versions of John Dowland's "Lachrimae Pavan", collected as part of the ECOLM project (www.ecolm.org) from different 16th and 17th-century sources, sometimes varying in quality (numbers of "wrong" notes, omissions and other inaccuracies), in scoring (for solo lute, keyboard or five-part instrumental ensemble), in sectional structure and in key;

F 50 variations by four different composers on the well-known baroque tune "Les Folies d'Espagne".

Within these three sets, the Folia are most consistent while the Twinkle pieces are the least consistent. Not only are many of the Twinkle pieces in different keys, but a few are monophonic and a few are missing certain entire sections of the piece. The amount of embellishments, elaborations, and rhythmic extensions in the Twinkle piece is fairly large as well. Twinkle is by far the most peculiar of our variations sets. As an example, Figure 6.1 contains a few of the "Twinkle" variations. These illustrate the relatively low note consistency (the number of and actual notes played) that characterizes some of the variations sets.

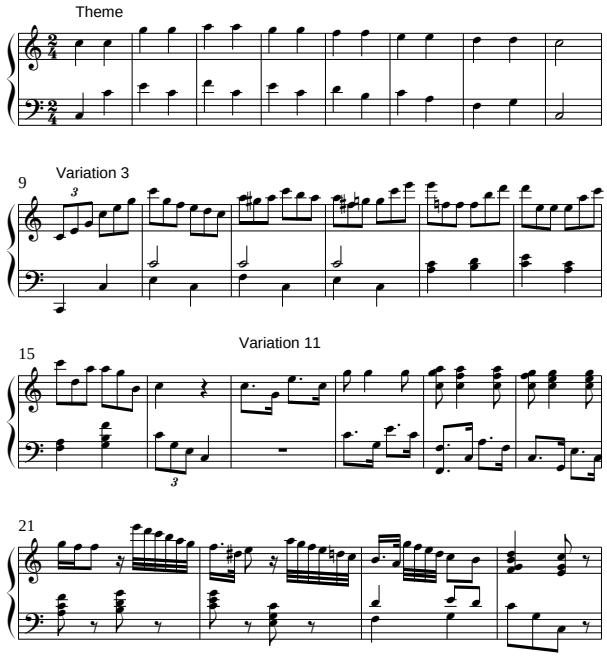


Figure 6.1. Sample excerpts from Mozart's "Ah! Vous dirai-je, Maman"

	GM	JPB	"Pure"
Known item			
24 Preludes	Naxos Audio	Naxos Audio	CCARH
24 Fugues	Naxos Audio	Naxos Audio	CCARH
Variations			
26 Twinkle	Simulated Audio	Simulated Audio	<Various>
75 Lachrimae	Simulated Audio	Simulated Audio	ECOLM
50 Folia	Simulated Audio	Simulated Audio	<Various>

Table 6.1. Query sets and their various sources

The Lachrimae pieces also vary, much more than the Folia pieces but somewhat less than the Twinkle pieces. However, the Lachrimae pieces have the additional complication of variugated sectional structure. For example, a single Lachrimae variation might have three main parts, ABC. Another variation might repeat these sections: AABBCC. Another might shuffle or intersperse these sections: ABABC. Another might repeat the entire piece: ABCABC. All these variations make retrieval more difficult.

6.1.2 Queries and Relevance Judgements

We have assembled, in various guises, fifteen different query sets. The basic five sets are, in order, 24 Preludes from Book I of Bach's Well-tempered Clavier, 24 Fugues from Book I of Bach's Well-tempered Clavier, and the same 26 Twinkle, 75 Lachrimae, and 50 Folia variations mentioned in the previous section.

From the CCARH Musedata collection (www.musedata.org) we have all 48 Preludes and Fugues in symbolic format; the exact note pitches, durations, and exact onsets are known. From various sources (see the previous section) we have obtained symbolic representations of the Twinkle, Lachrimae and Folia pieces. Collectively, these five sets of queries, which all exist primitively in the symbolic form, are called the "Pure" dataset, so as to denote that there has been no corruption of any of the notes in these pieces.

Additionally, from the Naxos Audio collection we have these same 48 Preludes and Fugues in raw audio form, as performed and recorded by human musicians [2]. We were unable to obtain, and did not have the resources to create for ourselves, actual human recordings of the Twinkle, Lachrimae and Folia pieces. Instead, we wrote software to convert these pieces to MIDI, then used a high quality MIDI soundfont (over 29 Megabytes in size) to convert these pieces to raw audio. This apparent weakness in our evaluation is countered by two facts: (1) These audio queries are still polyphonic, even if synthesized, and automatic transcription of overlapping and irregular-duration tones is still quite difficult, and (2) many of the variations on a piece are themselves quite different from a potential query, as we see in Figure 6.1, and good retrieval is still a difficult task. Even if the exact notes of a variation were used as a query, rather than the inaccurate (though perhaps slightly better because of the synthesized audio), quality retrieval is not guaranteed. While we hope to work with a human-produced audio collection for this retrieval experiment someday, as we have done with the known-item Naxos data above, we feel the gist of the evaluation has not been compromised.

With the audio versions of all five query sets in hand, we used the two polyphonic audio transcription algorithms to create imperfect symbolic representations of the audio pieces (see Section 3.1.1). The transcriptions from the first algorithm [82] are called the GM dataset, after the initials of the author of the algorithm. The transcriptions from the second algorithm [11, 12] are called the JPB dataset, also after the author's initials.

A summary of these fifteen query sets can be found in Table 6.1. In addition to labeling the datasets by their transcription class (GM, JBP, or "Pure"), we make the additional distinction that the Preludes and Fugues are the "Known item" queries, while the Twinkle, Lachrimae, and Folia

sets are the "Variations" queries. What this means, as well as how relevance is determined, is the subject of the next two sections.

6.1.2.1 Relevance for the Known Item queries

The 24 Bach Preludes and 24 Bach Fugues are collectively labeled the *Known Item* queries. We sometimes refer to these as the *PF* queries, as they are comprised of the preludes and fugues. Recall from Section 1.2.1 that with known item queries the user knows exactly what they want. There is a single piece in the entire collection that meets this information need; the user just needs to find it. Thus, relevance is defined as the one music piece that the user wants, the one piece of music that is the "same" as their query. For the 48 GM-transcribed preludes and fugues, the 48 JPB-transcribed preludes and fugues, and the 48 CCARH preludes and fugues, the piece in the source collection which is judged relevant to each query is the corresponding CCARH prelude or fugue.

The situation in which one has the full "Pure" (symbolic notation) CCARH version of a piece of music, and then wants to find that exact same piece, is somewhat artificial. One does not need complicated probabilistic retrieval systems to find the correct piece; a simple note-for-note string match solves the problem with 100% accuracy, as there is no degradation or variation in the query. Therefore, the "Pure" query set can be thought of as a measure of how "lossy" our probabilistic models are. The Markov assumption made in both the HMM and the harmonic modeling approach means that we ignore a certain amount of history, a certain amount of context. This assumption might cause another piece of music to look either as likely or perhaps even more likely than the original. If we cannot successfully retrieve the original piece from the collection, using the same original piece as a query, then our modeling techniques are problematic. We include the "Pure" query set to test this situation.

The two transcription query sets are more realistic. In these cases, the system is presented with imperfect transcriptions of raw, human played audio files. The question is whether this degraded query (Figure 3.2) can retrieve, at a high rank relative to all other music in the collection, the original "perfect" score (Figure 3.1). A system that does this well is robust and is able to capture a credible sense of musical similarity.

6.1.2.2 Relevance for the Variations queries

The 26 Twinkle pieces, 75 Lachrimae pieces, and 50 Folia pieces are collectively labeled the *Variations* queries. We sometimes also label this dataset as the *TLF* queries, after [T]winkle, [L]achrimae, and [F]olia. Unlike the known item queries, there are multiple pieces which are relevant to a query, as detailed in Section 1.2.2. For this work, relevance is defined as all variations on piece of music. In other words, when the query is one of the 26 Twinkle pieces, then all 26 Twinkle variations are relevant to that query. When the query is one of the 75 Lachrimae pieces, then all 75 Lachrimae variations are relevant to that query. In these experiments, we wish to determine whether our modeling approaches are useful for retrieving variations on a piece of music, rather than just the original. A good retrieval system would therefore return all variations toward the top of the 3,150 item list, and all non-variations further down.

We have chosen to make a query relevant to itself, in addition to being relevant to all other variations of its set. This was a decision that could have gone either way. The argument against making a query relevant to itself is that with the original symbolic queries, it is trivial to return the exact same piece at the top of a ranked list, for reasons discussed in the previous section. Interpolated precision at 0% recall will therefore be artificially boosted. The arguments for making a query relevant to itself are: (1) With the GM-transcribed and JPB-transcribed queries, it is not trivial to return the symbolic version of the same piece at the top of the ranked list, and (2) even if precision is artifically boosted at low recall, that boost is going to be consistent across all systems evaluated, as each system will make use of the exact same relevance judgements. Overall, we felt the arguments for outweighed the arguments against, and so we constructed the relevance judgements in that manner.

6.1.3 Training versus Testing Data

In this work, we made the decision not to split the collection into training and testing data sets. The primary reason for this is that the number of queries we were able to assemble are small enough that significance testing could become an issue if we fragmented the data in any way. Furthermore, we are following precedent in that the manner in which we have conducted our retrieval experiments is the traditional manner in which this has been done. From the earliest days of information retrieval, the approach as been to throw as many queries at a problem as possible with the notion that fluctuations in performance on a per-query basis get averaged out, and an accurate picture of the effectiveness of a system can be obtained.

We also point out that, due to the information retrieval nature of our experiments, we are not really testing on the training data to begin with. For example, for the known item queries, the audio-transcribed queries are from a completely separate source as the collection score files. For the variations queries, using one variation to find others implicitly assumes that variation is different from the others. Therefore, for basic system comparison, a split into testing and training data is not necessarily required.

However, we must make note of the following exceptions. For the known item queries, the "Pure" dataset queries are indeed samples drawn straight from the collection. This is testing on the training data. However, the point in that case *is* to test on the training data, as a method for establishing an upper bounds for each of the various methods. We discuss this to some extent already in Section 6.1.2.1.

Also, we made the choice in the variations queries to make a piece relevant to itself. For the remainder of the relevant and non-relevant pieces this is not an issue. For the audio transcriptions this is not a problem, either, as the transcribed piece never perfectly matches the score version of itself. However, for the straight, score-based variations experiments (again, the "Pure" dataset), we do acknowledge, as we have already acknowledged in Section 6.1.2.2, that precision at 0% interpolated recall is going to be artifically boosted. However, this boost is essentially the same across all systems and/or parameter settings; if there is train/test overlap bias, the bias is consistent. So what matters, and what we feel still remains clear, is the comparison of one system or parameter setting to the next, rather than the absolute value or magnitude of any one result.

In all, we feel we have taken proper care and have followed adequate procedures to ensure that we are not adversely affected by the training versus testing issue.

6.1.4 Mean Reciprocal Rank and Recall-Precision

Mean reciprocal rank is the standard measure for evaluating ranked lists of documents in which a single document is relevant to a query. The rank of that one relevant item, denoted r, is the position in the list at which that item occurs. Thus, the reciprocal rank is $\frac{1}{r}$. As we are interested in expected system performance, and not just the performance of a single query, we take the average of this reciprocal rank across all queries in a set, n. This is the mean reciprocal rank (MRR):

$$MRR = \frac{\sum_{i=1}^{n} \frac{1}{r_i}}{n} \tag{6.1}$$

Recall and *precision* are standard measures for evaluating ranked lists of documents in which multiple documents are relevant to a query. Recall is defined as:

$$Recall = \frac{Number\ of\ pieces\ retrieved\ and\ relevant}{Total\ relevant\ pieces\ in\ the\ collection} \tag{6.2}$$

and precision is defined as:

$$Precision = \frac{Number\ of\ pieces\ retrieved\ and\ relevant}{Total\ retrieved\ pieces\ in\ the\ collection} \tag{6.3}$$

In practice, recall is usually limited to what can be found in the top 1000 items in the ranked list. Precision is given *at various levels of recall*. For example, at that point in the ranked list when

30% of the relevant pieces of music have been retrieved from the collection, it is useful to know what proportion of those are relevant. Precision results are often shown in graphical or tabular form, at 11 interpolated points: 0%, 10%, 20%... recall through 100% recall. Additionally, the *mean average precision* score is often given, which is the average of all precision scores at every (non-interpolated) place in the ranked list at which a relevant document occurs. Recall-precision results are often averaged over a number of different queries.

When two systems are compared against each other, tabular recall-precision values are often accompanied by percentage change values and statistical significance tests. Sometimes, one system may have higher precision than another at low recall (the top of the ranked list), and lower precision at high recall. Evaluation of recall-precision results is user-dependent: A casual user or a DJ might deem any single variation on a piece of music acceptable, and thus would only be interested in precision at low recall, while a music copyright lawyer would likely be interested in finding all variations, and thus would be interested in precision at 100% recall. Ideal, however, is the case where one system outperforms another system at all levels of recall.

6.2 Effects of Initialization Parameters on HMMs

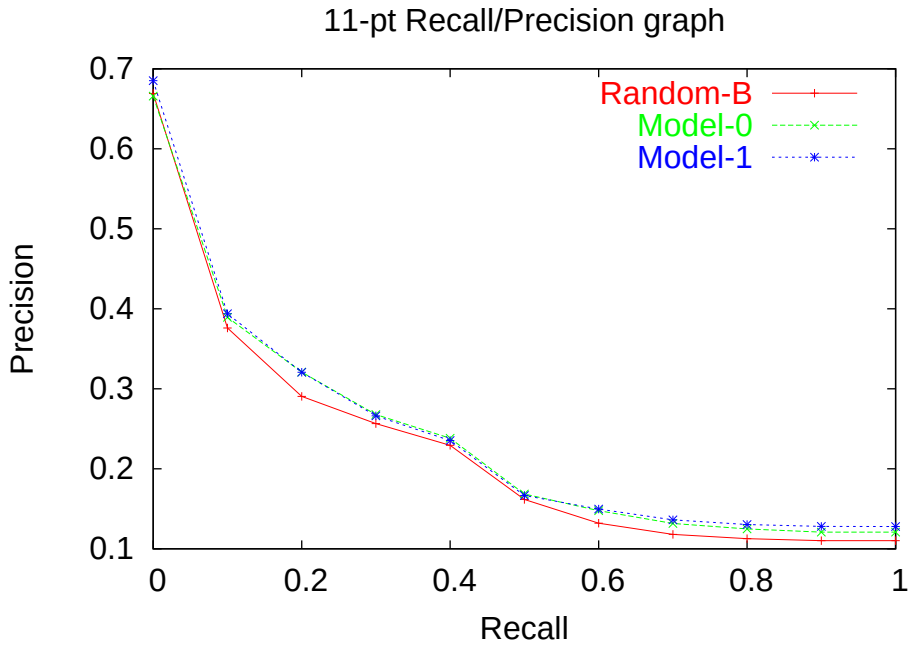
For our first experiment we turn our attention to the hidden Markov models developed in Chapter 4. Our intent is to use HMMs simply as a baseline system, showing that they are capable of solving the music retrieval task well, though not as well as harmonic models. Nevertheles, we still need to develop the best possible HMMs for use as that baseline. It would not be fair to compare harmonic models against a weakened or suboptimal HMM. Therefore, we want to find those initialization parameters which give the best performance. The queries used for these experiments are the full set of TLF and PF queries from the GM, JPB, and "Pure" sources. Essentially, we are using all the queries available to us. In later sections we will separate some of the queries out, and make comparisons based on the type or source of the query. For the moment, however, we need the big picture, though we did verify by hand the fact that the behavior across each of these individual query sets parallels the amalgamated behavior of the results presented in this section.

6.2.1 Selecting a [B]-initialization Method

In Section 4.3.3 we introduce two methods for observation symbol [B] distribution initialization, the participatory $Model_0$ and the proportional $Model_1$. In the participatory model, all observations which share at least one note in common with a chord state are given equal initial probability mass. In the proportional model, probability mass is assigned proportionally with the number of notes that a chord state and an observation have in common. The intuition behind $Model_0$ is that it would be generalizable; it would be able to capture a wider field of variations because it gave equal initial probability mass to observations tangentially related to a given chord state. In other words, the intuition is that it would enhance recall. The intuition behind $Model_1$ is that it would be more accurate or specific. By assigning more probability mass to observations that had more in common with a chord state, it would be able to weed out more non-relevant variations. In other words, the intuition is that it would enhance precision.

Our results can be found in Figure 6.2. For comparison, we also did a random initialization of the parameters, just to make sure that $Model_0$ and $Model_1$ were both performing intelligently. All experiments were run using the principled state transition [A] initialization described in Section 4.3.2. The table accompanying Figure 6.2 shows the values from which Figure 6.2 was constructed. It also shows the percentage change (%Chg) and statistical significance along each of the 11-pt interpolated precision-recall values, as well as the non-interpolated mean average precision. An asterisk indicates statistical significance at a 0.05 level.

On average, $Model_0$ significantly outperforms random initialization at every recall-precision point, including on average, except for interpolated 0% recall (the very top of the ranked list). At this point, $Model_0$ is essentially equivalent to random. $Model_0$ also pulls in many more (5.25%) pieces in the top 1000 (Rel|ret) than does random ranking. We feel that this justifies the intuition that $Model_0$ is a recall-enhancing technique.



	Random B	$Model_0$			$Model_1$		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	597000	597000			597000		
Relevant:	26689	26689			26689		
Rel|ret:	15423	16233	+5.25	0.0000*	15406	-0.11	0.5987
Interpolated Recall - Precision							
at 0.00	0.6691	0.6658	-0.5	0.6388	0.6851	+2.4	0.0160*
at 0.10	0.3760	0.3894	+3.6	0.0091*	0.3938	+4.8	0.0004*
at 0.20	0.2905	0.3208	+10.4	0.0000*	0.3206	+10.4	0.0000*
at 0.30	0.2565	0.2678	+4.4	0.0020*	0.2662	+3.8	0.0007*
at 0.40	0.2293	0.2381	+3.9	0.0125*	0.2354	+2.7	0.0287*
at 0.50	0.1616	0.1685	+4.3	0.0464*	0.1667	+3.2	0.0570
at 0.60	0.1320	0.1477	+11.9	0.0019*	0.1497	+13.4	0.0000*
at 0.70	0.1180	0.1316	+11.5	0.0073*	0.1361	+15.3	0.0000*
at 0.80	0.1127	0.1248	+10.7	0.0171*	0.1303	+15.7	0.0000*
at 0.90	0.1101	0.1208	+9.7	0.0343*	0.1280	+16.3	0.0000*
at 1.00	0.1101	0.1208	+9.7	0.0343*	0.1279	+16.2	0.0000*
Average precision (non-interpolated) over all rel docs							
	0.2066	0.2162	+4.65	0.0125*	0.2210	+6.96	0.0000*

Figure 6.2. HMM [B] Initialization: $Model_0$, $Model_1$ and random

The second conclusion we draw is that $Model_1$ is consistently better than random initialization at every level of precision-recall, including at interpolated 0% recall, as well as on average. However, the number of relevant documents in the top 1000 (Rel|ret) is equivalent to the random initialization. Furthermore, $Model_1$ is 2.9% better than $Model_0$ at interpolated 0% recall (this improvement is statistically significant). We feel that this justifies the intuition that $Model_1$ is a precision-enhancing technique.

It is unfortunate that neither $Model_0$ or $Model_1$ is better than the other across all measured values. $Model_0$ is better for enhanced recall; $Model_1$ is better for enhanced precision. However, we feel that most users are interested in precision at the top of the ranked list and therefore select $Model_1$ as the better of the two. We will use $Model_1$ as the baseline observation symbol [B] distribution initialization technique.

6.2.2 Selecting an [A]-initialization Method

In the previous section, all experiments were run using the principled state transition [A] initialization described in Section 4.3.2. Now that we have selected $Model_1$ as the [B] initialization technique, we want to make sure that our principled [A] initialization is doing something significant. It is one thing to posit clever initialization schemes; it is another to ensure that they improve retrieval. Therefore, we compare $Model_1$ with the principled [A] initialization against $Model_1$ with a random [A] initialization.

As we see from Figure 6.3, the principled [A] outperforms random [A] at every level of recall, and the improvement is statistically significant at all points but one. This shows that our state transition distribution initialization methods work well. However, if we examine Rel|ret, the number of relevant documents in the top 1000, the principled [A] method actually yields a statistically significant drop. However, the drop is small enough (-1.15%) that we nevertheless conclude that the principled [A] initialization should be used when constructing triad-based hidden Markov models of music.

6.3 Comparison of HMMs with Basic Harmonic Models

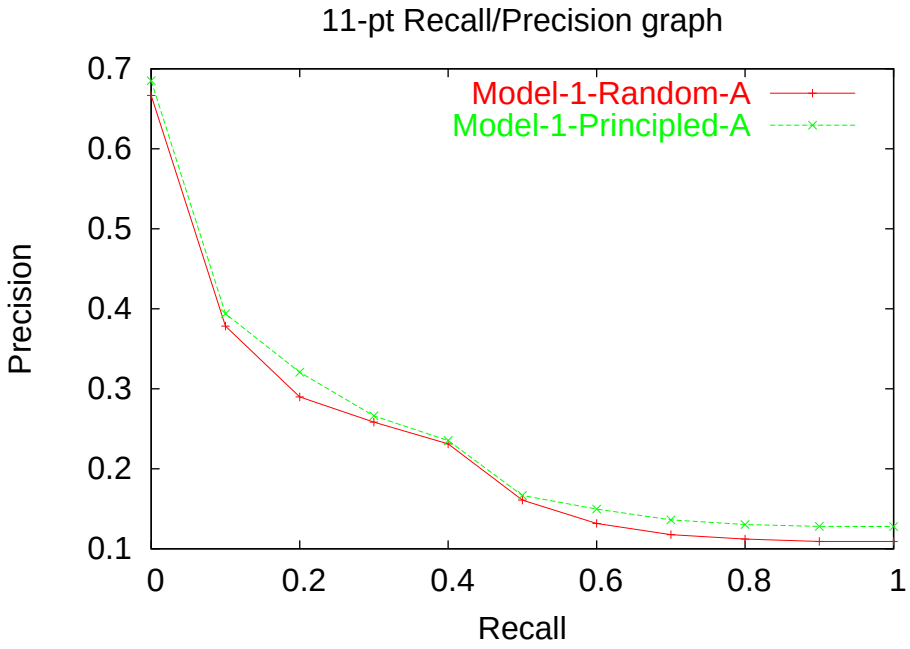
In the previous section we concluded that the "best" hidden Markov model to use for the music information retrieval task was one which had been initialized with the principled [A] distribution and the $Model_1$ [B] distribution from Section 4.3. We now compare a system based on these HMMs with one based on harmonic models.

In order to compare these HMMs with a basic harmonic modeling approach, the parameters of the harmonic model should be as close or analagous as possible to those of the HMM. The HMMs are 1^{st}-order models and the standard HMM state transition reestimation function uses values across a single previous timestep to update the estimate of the current timestep. Therefore, our basic harmonic models will also be 1^{st}-order, and the harmonic descriptions will be smoothed using a single previous timestep. Global backoff, which we consider the least "intrusive" upon a model's existing probability estimates, is used to ensure that estimates of zero probability do not adversely affect the relative entropy harmonic model scoring function.

In Section 6.2 we lumped together the known item and the variations queries. We also lumped together the various query sources: GM, JPB, and "Pure". This was done to get an overall, average picture of how well HMM-based retrieval was working. The goal was to get the overall best HMM-based retrieval system as possible, within the structure of the model and the parameters available. In this section, however, our goal is to compare HMMs and harmonic models directly. Therefore, whereas we previously amalgamated the various query sets and asked the reader to trust our offline analysis that behavior was similar across all of them, we now take a closer look.

6.3.1 Known Item Queries

The first query sets we wish to test are the Prelude and Fugue known item queries. Mean reciprocal rank results for the various query sources ("Pure", GM, and JPB) across the "best" HMM and the



	$Model_1$ with Random [A]	$Model_1$ with Principled [A]		
			%Chg	T-test
Total number of documents over all queries				
Retrieved:	597000	597000		
Relevant:	26689	26689		
Rel|ret:	15586	15406	-1.15	0.0000*
Interpolated Recall - Precision				
at 0.00	0.6666	0.6851	+2.8	0.0097*
at 0.10	0.3783	0.3938	+4.1	0.0035*
at 0.20	0.2898	0.3206	+10.6	0.0000*
at 0.30	0.2583	0.2662	+3.1	0.0077*
at 0.40	0.2311	0.2354	+1.8	0.1374
at 0.50	0.1609	0.1667	+3.6	0.0385*
at 0.60	0.1317	0.1497	+13.6	0.0000*
at 0.70	0.1176	0.1361	+15.7	0.0000*
at 0.80	0.1121	0.1303	+16.2	0.0000*
at 0.90	0.1092	0.1280	+17.2	0.0000*
at 1.00	0.1092	0.1279	+17.1	0.0000*
Average precision (non-interpolated) over all rel docs				
	0.2072	0.2210	+6.68	0.0000*

Figure 6.3. [A] Initialization for $Model_1$: Principled versus random

"basic" harmonic model results are given in Table 6.2. For statistical significance we use a sign test, with an asterisk indicating significance at the 0.05 level.

	GM	JPB	"Pure"
HMM	0.118	0.373	0.996
Harmonic	0.417 (+253.4%)*	0.433 (+16.1%)*	0.993 (-0.3%)

Table 6.2. Comparison of HMMs with basic harmonic models, MRR values, known item queries

The first thing we notice is that on the "Pure" queries, both the HMMs and the harmonic models do not, through their limited horizon Markov assumption, lose enough information to make other pieces of music appear more likely. In the HMM case, 47 of the queries had the correct known item ranked 1^{st}, and one query had the correct item ranked 3^{rd}. In the harmonic model case, 46 of the queries had the correct item ranked 1^{st}, one query had the correct item ranked 2^{nd}, and one query had the correct item ranked 3^{rd}. The queries on which each model "failed" were not the same. Overall, this is not a statistically significant difference, so we conclude that in terms of information loss due to the Markov assumption the two modeling approaches are equivalent. When we examine the known item queries from the two audio-transcribed sources, however, the results are different. With the JPB transcriptions harmonic modeling is 16% better, and with the GM transcriptions harmonic modeling is over 250% better. Both improvements are stastically significant.

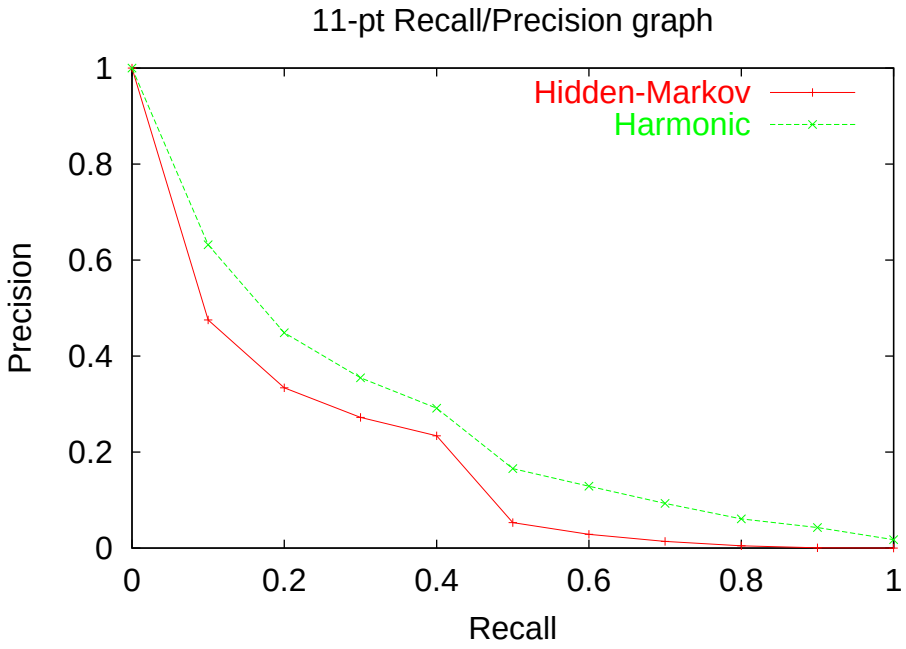
Though one of our goals is to show that the harmonic modeling approach is better, it is also interesting to look at the differences between the two transcriptions, GM and JPB, for the hidden Markov approach. Though we do not spend much time in this work discussing the differences between these two transcription techniques, we have noticed that, in general, the GM transcriptions tend to be "looser", and the JPB transcriptions tend to be "tighter". What we mean is that if one looks at the notes that each techniques is supposed to recover, GM tends to get more of the correct notes, but also adds more incorrect notes, resulting in additional simultaneities as well as fuller existing simultaneities. JPB, on the other hand, tends not to add too many extra notes, but also misses more of the notes that should be there. The GM transcriptions have roughly 130-150% of the number of simultaneities of the JPB transcriptions.

These differences between the two transcriptions go far, we feel, in explaining the large difference between the hidden Markov results on each transcription set. We will get into this more in section 6.4, but our basic intuition is that HMMs do quite well at modeling sequences of chords in which a few notes have been inserted or deleted *from existing simultaneities*. However, when simultaneities themselves are added, when there are insertions and deletions between simultaneities, rather than just within simultaneities, HMMs have a difficult time. The GM transcriptions add more simultaneities than the JPB transcriptions take away. This could explain why HMMs do much worse on the GM data than on the JPB data. In any case, these results show that HMMs are much more sensitive to the quality of the transcription than are harmonic models.

6.3.2 Variations Queries

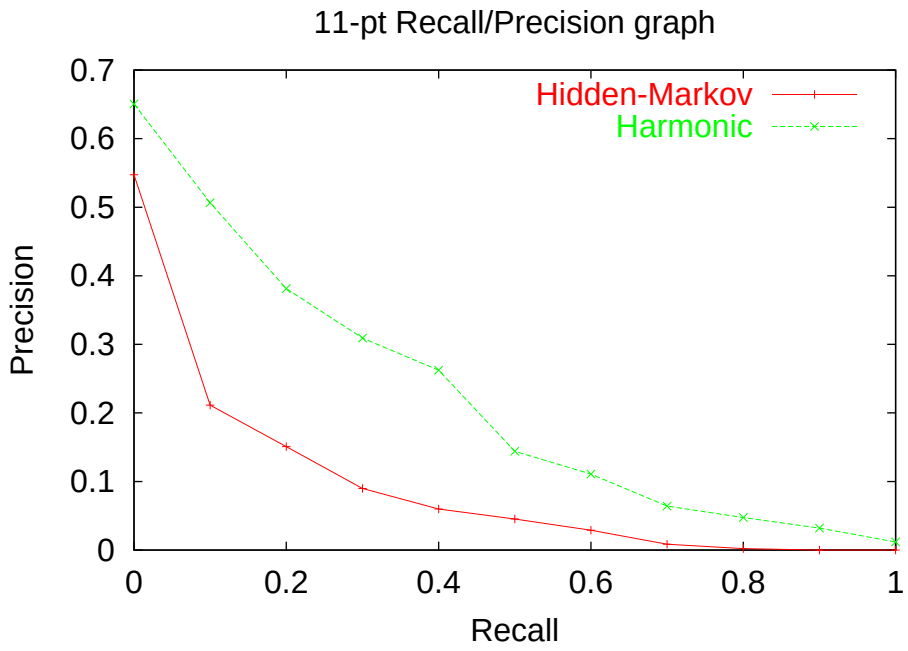
The second query sets we wish to use are the Twinkle, Lachrimae, and Folia variations queries. Recall-precision results for the various sources ("Pure", GM, and JPB) across the "best" HMM and the "basic" harmonic model results are given in Figures 6.4, 6.6, and 6.5, respectively. Statistical significance is determined with a T-test and indicated with an asterisk.

In the "Pure" dataset, from Figure 6.4, we notice that at 0% interpolated recall, there is no difference between hidden Markov models and basic harmonic models, as both are at 100% precision. We attribute this to the decision made in Section 6.1.2.2 to let a variation be relevant to itself. 100% precision at 0% recall on the "Pure" dataset is essentially telling us that neither modeling approach is lossy. Beyond that, however, harmonic models outperform hidden Markov models at every recall point, on average, and they even pull in many more relevant documents in the top 1000 retrieved (Rel|ret). The differences are not only statistically significant, they are fairly large: 13.5% more



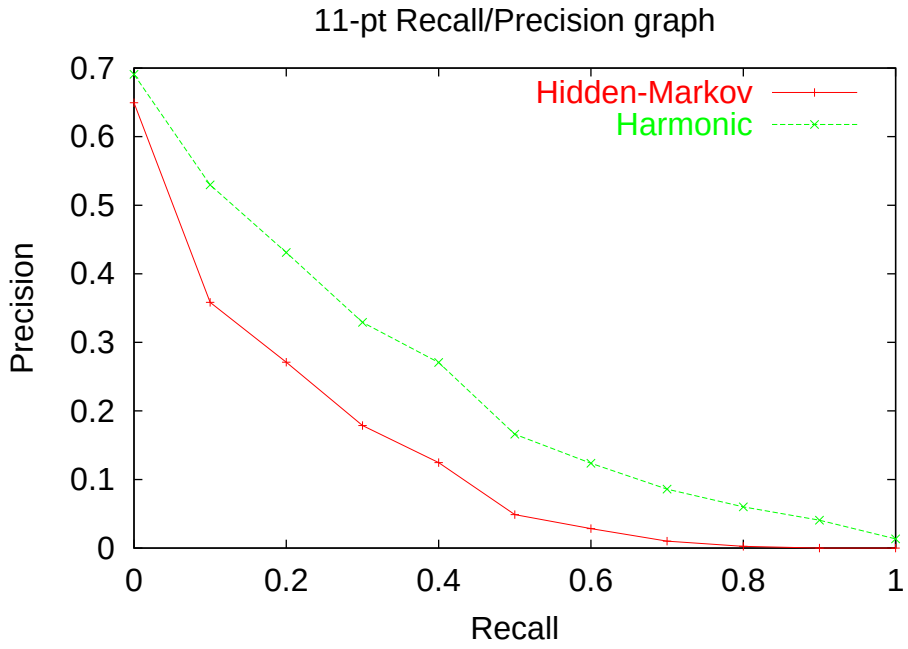
	Hidden Markov Model	Harmonic Model		
			%Chg	T-test
Total number of documents over all queries				
Retrieved:	151000	151000		
Relevant:	8801	8801		
Rel|ret:	5093	5782	+13.53	0.0001*
Interpolated Recall - Precision				
at 0.00	1.0000	1.0000	+0.0	1.0000
at 0.10	0.4754	0.6320	+32.9	0.0000*
at 0.20	0.3338	0.4487	+34.4	0.0001*
at 0.30	0.2721	0.3547	+30.4	0.0045*
at 0.40	0.2340	0.2913	+24.5	0.0480*
at 0.50	0.0530	0.1655	+212.3	0.0000*
at 0.60	0.0287	0.1289	+349.0	0.0000*
at 0.70	0.0138	0.0930	+576.1	0.0000*
at 0.80	0.0048	0.0608	+1175.0	0.0000*
at 0.90	0.0005	0.0427	+8737.8	0.0000*
at 1.00	0.0000	0.0174	+undef	0.0000*
Average precision (non-interpolated) over all rel docs				
	0.1779	0.2513	+41.25	0.0000*

Figure 6.4. Comparison of HMMs with basic harmonic models, variations, “Pure” data



	Hidden Markov Model	Harmonic Model		
			%Chg	T-test
Total number of documents over all queries				
Retrieved:	151000	151000		
Relevant:	8801	8801		
Rel|ret:	4881	5665	+16.06	0.0000*
Interpolated Recall - Precision				
at 0.00	0.5474	0.6506	+18.9	0.0148*
at 0.10	0.2114	0.5065	+139.6	0.0000*
at 0.20	0.1508	0.3813	+152.9	0.0000*
at 0.30	0.0899	0.3091	+244.0	0.0000*
at 0.40	0.0599	0.2624	+337.8	0.0000*
at 0.50	0.0453	0.1441	+218.0	0.0000*
at 0.60	0.0289	0.1109	+283.7	0.0000*
at 0.70	0.0085	0.0641	+656.4	0.0000*
at 0.80	0.0021	0.0474	+2190.7	0.0000*
at 0.90	0.0000	0.0321	+undef	0.0000*
at 1.00	0.0000	0.0119	+undef	0.0000*
Average precision (non-interpolated) over all rel docs				
	0.0776	0.1974	+154.39	0.0000*

Figure 6.5. Comparison of HMMs with basic harmonic models, variations, GM data



	Hidden Markov Model	Harmonic Model		
			%Chg	T-test
Total number of documents over all queries				
Retrieved:	151000	151000		
Relevant:	8801	8801		
Rel|ret:	5148	5576	+8.31	0.0138*
Interpolated Recall - Precision				
at 0.00	0.6495	0.6908	+6.4	0.2599
at 0.10	0.3584	0.5297	+47.8	0.0000*
at 0.20	0.2711	0.4311	+59.0	0.0000*
at 0.30	0.1786	0.3291	+84.3	0.0000*
at 0.40	0.1247	0.2707	+117.1	0.0000*
at 0.50	0.0488	0.1661	+240.1	0.0000*
at 0.60	0.0284	0.1236	+335.4	0.0000*
at 0.70	0.0102	0.0860	+740.4	0.0000*
at 0.80	0.0026	0.0601	+2183.8	0.0000*
at 0.90	0.0000	0.0406	+undef	0.0000*
at 1.00	0.0000	0.0135	+undef	0.0000*
Average precision (non-interpolated) over all rel docs				
	0.1245	0.2184	+75.46	0.0000*

Figure 6.6. Comparison of HMMs with basic harmonic models, variations, JPB data

relevant documents in the top 1000, 41% higher mean average precision, and precision at high recall 1100% or more. Harmonic models are better at finding variations.

These findings are consistent across the GM and JPB datasets as well (see Figures 6.6 and 6.5). The absolute precision values are lower, due to the lower "quality" of the query. But otherwise, harmonic models outperform hidden Markov models across all recall-precision points, on average (75% for the GM data, 150% for the JPB data), and also in terms of Rel|ret. Harmonic models are significantly better than HMMs at finding variations even when presented with an imperfect audio transcription as a query.

6.4 Analysis of HMMs versus Harmonic Models

We see from the MRR performance of the "Pure" known item queries in Section 6.3.1 and the 100% precision at 0% recall performance of the "Pure" variations queries in Section 6.3.2 that intelligently initialized HMMs are able to recognize the data on which they were estimated quite well. They are not lossy models. We see from the GM and JPB known item queries that even when the query is degraded through imperfect transcription from the audio, HMMs are able to retrieve the correct piece at a fairly decent ranking; an MRR of 0.37 or 0.12 is not perfect, but it is still very good. Random ranking would place the known item at a MRR of 0.000635. Furthermore, HMMs yield a mean average precision for the variations queries of 0.178, 0.125, and 0.078 on the "Pure", GM, and JPB datasets, respectively. Random ranking has a mean average precision of 0.019. It is clear that HMMs are able to capture a credible sense of music similarity, using both audio-degraded "variations" and real-world composed variations; they significantly outperform random retrieval.

However, recall from Chapter 5 that the reason we developed harmonic models is that we felt hidden Markov models would not perform as well. When we compare the results obtained by harmonic models against those from HMMs the picture is clear: HMMs are lacking. We attempted to create HMMs and harmonic models in a manner as analogous to each other as possible. Nevertheless, harmonic models significantly outperform HMMs on almost all counts, and are equal to HMMs on the remaining few counts, as was demonstrated in Section 6.3.

6.4.1 Explanation: Overfitting of the P(o|s) Distribution

Evaluation and analysis of ranked lists is always a tricky matter, and it is not always completely clear why certain techniques fail or why other techniques succeed. But we would like to discuss a few reasons why this might be happening. There are two major sources of variability in our collection. The first is texture, which is related to the number of notes in a simultaneity. For example, the Mozart Twinkle Variation #11 and the Mozart Twinkle Theme use different sets of notes to express the same melody (see Figure 6.1 on page 57). In the theme, at most two notes at a time are used. In Variation #11, there is a much richer texture, with three and sometimes four different notes used within a single simultaneity.

One explanation that has been suggested that the reason HMMs do not do as well as harmonic models is because the Baum-Welch reestimation algorithm overfits the P(o|s), or [B], distribution to each individual piece in the collection. What might be needed to overcome this is problem is an output distribution that remains fixed, that does not get reestimated, and that is the same for every model/piece in the collection.

We wish to test this with a couple of quick experiments. In these experiments, the output distribution [B] is initialized in a certain manner and then not allowed to change. Reestimation of the state transition distribution proceeds as normal. The first model we test is an HMM estimated using a $Model_1$ [B] distribution, from Section 4.3.3.2; we call this model ϕ-fixed. The second model is an HMM using the harmonic model partial observation function, Equation 5.1 in Section 5.2.1. Mappings between s and o are computed as between (observation) simultaneity and (lexical) chord from this harmonic description function. We call this model τ-fixed. In both cases, [B] remains fixed.

Our first experiment is with the known item queries. The results may be found in table 6.3. For comparison, we list standard HMM $Model_1$ (in which B is allowed to be reestimated) as well as the

	GM		JPB		"Pure"	
Random	0.000635		0.000635		0.000635	
HMM, ϕ-fixed	0.00354	(+457%*)	0.00371	(+484%*)	0.00351	(+453%*)
HMM, τ-fixed	0.00347	(+446%*)	0.00334	(+426%*)	0.00329	(+418%*)
HMM, $Model_1$	0.118	(+18k%*)	0.373	(+59k%*)	0.996	(+157k%*)

Table 6.3. Comparison of fixed output distribution HMMs with various other models, MRR values, known item queries

score for a random ranking of the list. Percentage difference from the random baseline are listed in parenthesis, and statistical significant (signtest) is indicated with an asterisk. These results show us that, while the fixed [B] approaches still do fairly well against random ranking, they still pale in comparison with the HMM in which we allow Baum-Welch to reestimate [B].

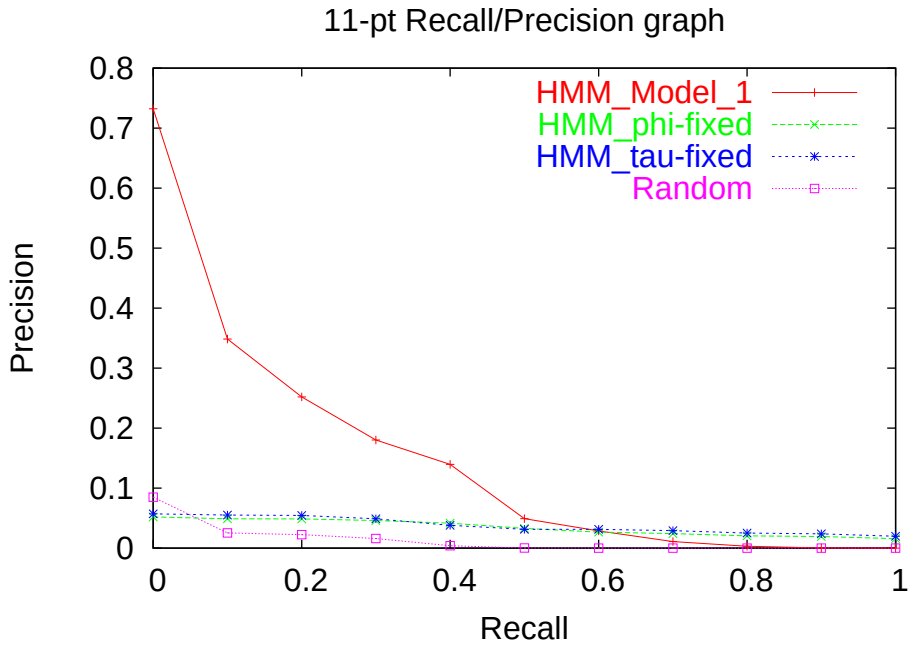


Figure 6.7. Comparison of fixed output distribution HMMs with various other models, recall-precision curve, variations queries

Our second experiment is with the variations queries. Again, we compare against the standard (variable [B]) HMM $Model_1$ as well as random ranking. The results in Figure 6.7 show us that these fixed [B] approaches are not quite as good as the standard HMM approach. While it is true that precision at high recall is better than standard HMMs for the fixed [B] approaches (the different is statistically significant), it is also true that precision at low recall is no different than random ranking for the fixed [B] approaches (the difference is not statistically significant).

Our interpretation of these results is that the issue of overfitting the output distribution is not much of a factor. These experiments do not prove this, of course, as it could be that we have simply not found the correct fixed output distribution. Perhaps training the output distribution on the entire corpus might be a better approach. But these experiments seem to suggest that a different explanation might be necessary to explain the difference between HMMs and harmonic models.

6.4.2 Explanation: Gaps in the state sequence

In the previous section we mentioned that there are two major sources of variability in our collection. The first was texture. The second is the insertion and deletion of entire simultaneities. For example, the "salient" (melody- or theme-carrying) simultaneities of Mozart Twinkle Variation #3 and the Mozart Twinkle Theme (those on beats 1 and 2 of each measure) have roughly the same textures (same number of notes and, mod 12, the same actual notes). But Variation #3 adds two extra simultaneities after each of the Theme's one simultaneity. This is the situation in which we believe that HMMs get into trouble.

Unfortunately, we have not come up with any additional experiments to test the following explanations. But our intuition is that HMMs do not do as well as harmonic models because the Baum-Welch reestimation procedure pushes the state transition [A] distribution toward the most likely states *for the data from which the model is estimated*. This happens regardless of whether or not [B] is fixed. When simultaneities (and therefore states) in a hitherto unseen piece of music are "inserted" it throws off the HMM enough so that the model of some other (non-relevant) piece begins to look more likely. The original HMM has no principled estimate for the new state transitions introduced by the insertions into the observation (query) sequence.

We believe the situation is similar when simultaneities are "deleted". For example, the GM and JPB audio transcriptions do not always accurately detect the onset of new notes. Therefore, many simultaneities get dropped. The HMM which was estimated from the original, score form of that audio expects to see a certain progression of states. Baum-Welch has pushed its state-transition distribution toward those state transitions which make the original score appear most likely. When a simultaneity is dropped, the HMM has little notion of what the next state to bridge that gap might be.

A quick analogy to speech recognition might be helpful. Consider a basic speech recognition task in which words are recognized by estimating, on a per-word basis, hidden Markov models of phoneme (state) to audio-signal (observation) distributions, and models of phoneme (state) to phoneme (state) transition distributions. Suppose that we have models of two words: "incredible" and "inedible". If the phonemes which make up the word "incredible" were spoken by both a Bostonian and a Texan, we would expect that the model for "incredible" would yield a higher likelihood for both the Bostonian audio and the Texan audio, than would the model for "inedible", as differences in dialect are accounted for by the observation distribution function. However, suppose for whatever reason that the [cr] sound had been dropped from the audio signal (loose connection on the microphone wire during the recording process). The observation then presented to the system would sound like "inedible", and the model estimated from "inedible" would be more likely than the model estimated from "incredible", even though it should not be, given the speaker's intention.

On the other hand, suppose phonemes had been added to the sound. Speakers often do this for emphasis. Something is not "fantastic". It is absolutely "fan-flipping-tastic"; this is called tmesis. And yet if a model were trained on the phoneme sequence from *fantastic*, there would be less guarantee that model would be the one selected as having been the most likely to have produced the word. We believe that what creates the most problems for the hidden Markov modeling approach is "music tmesis", contiguity rather than texture, insertions and deletions of simultaneities rather than of notes in a simultaneity [81]. This is far more common in music than speech.

Harmonic models overcome this problem by decoupling the observation function from the state-transition estimation. By attaching a vector of partial observations to each simultaneity, we have the liberty to shuffle around portions of observations from timestep to timestep in a manner guided by music-theoretic or structural notions. In particular, by pushing portions of observations forward in time, we overcome the negative effects of insertions and deletions of simultaneities in a sequence without introducing too many false positives.

Returning to the speech recognition example, if, when estimating a model for the word *incredible* we "push forward" a portion of the observation of the [in] chunk into the same window in which we observe the [cr], then we can have some sort of realistic estimation for the entire [inedible] sequence as well as the [incredible] sequence. If we continue "pushing forward" chunks into neighboring windows we will also have estimates for [incrible] and [inedle], too. But those are not going to be similar to

any other "non-relevant" terms we might observe (such as the words "giraffe" or "indecision"), and thus will not adversely affect the performance of our system.

6.4.3 Final Word on HMMs

By no means are the HMMs introducted in Chapter 4 and evaluated in this chapter the final word on hidden Markov models applied to the problem of ad hoc music information retrieval. To a certain extent, the HMMs introduced in this dissertation were somewhat of a strawman. We wished to show that HMMs in general did a fair job of solving our problem but that harmonic models did better. As such, there are undoubtedly many modifications and extensions that can be added to HMMs, such as links or conditional relationships on not just the previous state, but the previous two states, or on some cache model of previous states, and so on. However, inference and model estimation undoubtedly become more expensive and complicated as well. Harmonic models are as analogous as possible to our current, basic HMM structure and are not only more effective, but computationally much cheaper. If HMMs may be improved upon, then harmonic models may be improved upon as well.

In the remainder of this evaluation chapter we will therefore examine a number of techniques which should yield improvements over basic harmonic models. Section 6.6 will address the issue of smoothing of the harmonic description and how that affects retrieval, which should hopefully further support the analysis given in this section. But overall, the fact that these basic harmonic models outperform similarly-estimated hidden Markov models, and that hidden Markov models themselves are not an unreasonable approach to music information retrieval demonstrates that harmonic modeling is an effective foundation for polyphonic music retrieval.

6.5 Detailed Look at the TLF Queries

To this point in the evaluation we have been averaging the 24 Bach Preludes and 24 Bach Fugues together into a single 48-query known item set. We have also lumped the 26+75+50 Twinkle, Lachrimae, and Folia variations into a 151-query variations set. We will continue to lump, or average, all these queries together for the remainder of the evaluation. However, we feel it is a useful or interesting interlude to quickly take a look at the performance of each individual query set.

In the Prelude and Fugue queries, there was essentially no difference or variation in average scoring across each set. However, the Twinkle, Lachrimae, and Folia pieces are quite heterogeneous in terms of their intra-set consistency. The graph in Figure 6.8 shows the recall-precision curve for each of the TLF query sets, as determined by the basic harmonic modeling approach.

Referring the reader back to Section 6.1.1, we recall that the Folia queries were the most internally consistent. Therefore, we are not surprised to see that precision remains relatively high as recall increases. The Folia were, in a sense, the easiest query set. On the other hand, the Twinkle set was the most difficult. The number of notes, insertions and deletions of simultaneities, key shifts, rhythmic variations, and even length was different from piece to piece. Precision is initially high for those pieces most similar to a query, but the Twinkle recall-precision curve has the sharpest or steepest drop. There are certain pieces in the set that are just too different, and while we still do better than the HMM approach, we do not do as good, in general, on the Twinkle queries as on the Folia queries. Finally, the Lachrimae set somewhat mirrors the Twinkle set, though precision does remain a bit higher at medium recall.

We do not wish to draw any more conclusions from this data, nor do we wish to engage in further analysis on a query-by-query basis across different systems. We simply wish to show that, like any information retrieval system, some queries are going to be more or less successful. It is not always possible to know, a priori, which queries will be which, though it was interesting in this small study to see that the query set which we thought was going to be the easiest (Folia) was indeed so.

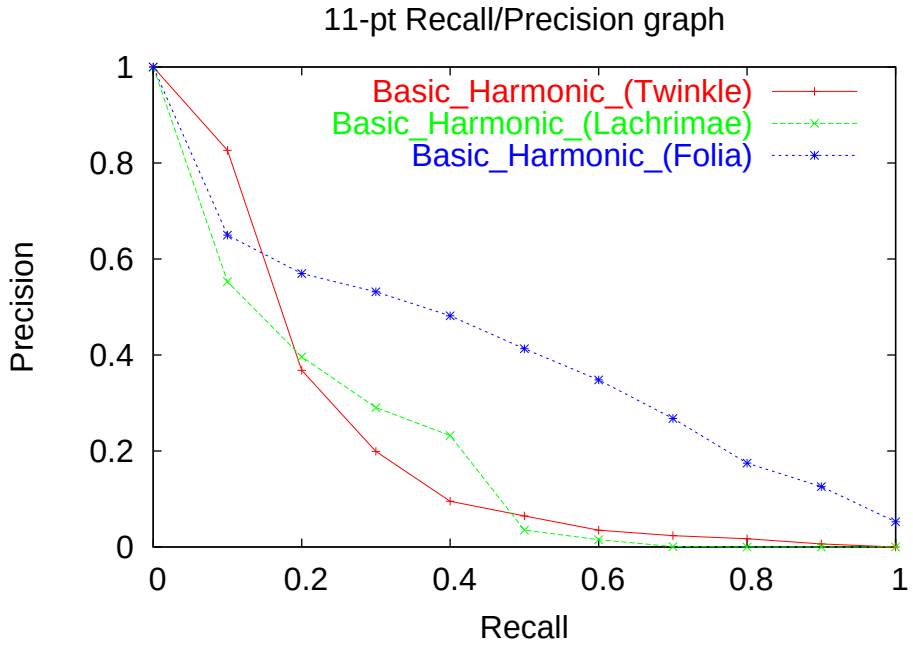


Figure 6.8. TLF queries presented individually, basic harmonic modeling approach

6.6 Effects of Smoothing on Harmonic Models

In Section 6.4 we made a number of claims about the problem that arise from insertions and deletions in the sequence of simultaneities. Our solution to these problem is to "push forward", or smooth, the harmonic description partial observations from one time step to the next in order to bring these gaps. In this section, we wish to further verify some of those claims. To begin with, we use the 1^{st}-order harmonic models evaluated in Section 6.3. However, instead of smoothing using only the previous window, we enlarge that window and evaluate smoothing with the previous two and previous three windows. For comparison we also smooth using the current window only.

6.6.1 Known Item Queries

The results for the known item queries are given in Table 6.4. An asterisk indicates statistical significance at the 0.05 level using a sign test. Our original, "basic" harmonic model, with smoothing across a previous window only, is used as a baseline and is indicated in bold.

	GM		JPB		"Pure"	
Current Window Only	0.081	(-80.6%)*	0.107	(-75.3%)*	0.960	(-3.3%)*
Previous Window	**0.417**		**0.433**		**0.993**	
Previous Two Windows	0.467	(+12.0%)	0.564	(+30.3%)*	0.995	(+0.2%)
Previous Three Windows	0.606	(+45.3%)*	0.525	(+21.2%)*	0.991	(-0.2%)

Table 6.4. Effects of smoothing on harmonic models, MRR values, known item queries

These results show that, with no smoothing, performance decreases across all data sets, the two transcribed as well as the "Pure" queries. The performance hit is most noticable for the tran-

scribed queries. Again, as was mentioned in the previous section, we believe this is because of the simultaneity insertions and deletions that inevitably come with imperfect audio transcriptions.

On the other hand, when we increase the size of the smoothing window, there is little to no effect on the "Pure" data, but significant improvement on the transcribed queries. It is reassuring to see that the increase in smoothing does not alter the resulting estimated Markov models enough to cause incorrect pieces in the collection to appear more similar than the correct piece. It is even more reassuring to see that this same larger-window smoothing process yields a large increase in mean reciprocal rank for the imperfect transcription queries . The largest smoothing window we tested aids the GM data more than the JPB data, but regardless of the parameter setting, larger windows yield significant, substantial improvements in the mean reciprocal rank. This is further evidence that our decoupled harmonic modeling approach is an effective one.

6.6.2 Variations Queries

In the interest of space, we now amalgamate the GM, JPB, and "Pure" data. The results in Section 6.3.2 show us that performance follows the same patterns for all three datasets, but that because the pure data is a better "transcription", it gives categorically better results. If we now average all the query sets together, we will still get the same *relative* comparison, one parameter setting to another. Again, we wish to emphasize that this is done in the interest of space, rather than to obscure any results.

In Figure 6.9 we can see that the results for the variations queries are quite similar to those of the known item queries. The larger smoothing windows give significantly better results; current window only smoothing gives significantly worse results. In the table accompanying the figure, we present the current windows and the previous three window smoothing. There is no statistically significant difference between the previous two and previous three windows, as the graph shows, so we do not include the previous two.

We draw the same conclusions from these variations results as we did from the known item results in Section 6.6.1: no smoothing hurts performance, and more smoothing helps performance. Though we did not test window sizes beyond the previous three, remember from Equation 5.3 on page 46 that the effect that the previous simultaneity has on the current simultaneity is inversely proportional from the distance, in numbers of intervening simultaneities, to the current simultaneity. We therefore do not believe that even larger windows will aid retrieval; nor do we believe that they will hurt retrieval. They will simply have less and less effect. We did not test this, however.

6.7 Effects of Model Order on Harmonic Models

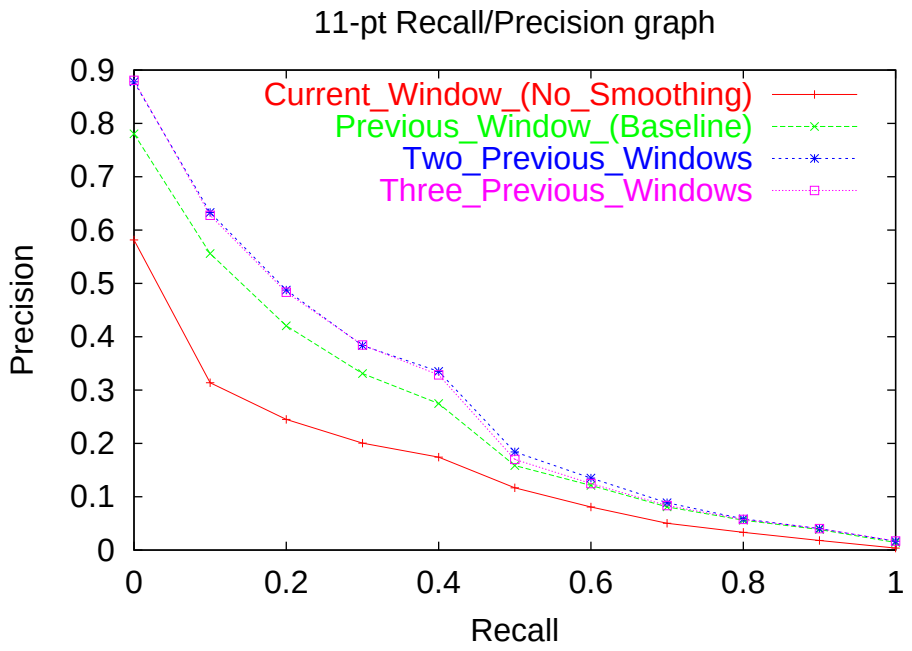
The next aspect of our harmonic models we wish to test is whether increasing the order of the model aids retrieval. Higher model orders capture greater amount of sequence or progression, and since music is sequential in nature, there is reason to test the effect of model order on retrieval.

6.7.1 Known Item Queries

The results for the known item queries are given in Table 6.4. An asterisk indicates a signtest statistical significance at the 0.05 level. As smoothing has been shown to help, we use a 1^{st}-order model with previous three window smoothing (or *full smoothing*) as a baseline.

	GM		JPB		"Pure"	
0^{th} order, full smoothing	0.625	(+3.1%)	0.510	(-2.9%)	0.991	(+0.0%)
1^{st} order, full smoothing	**0.606**		**0.525**		**0.991**	
2^{nd} order, full smoothing	0.701	(+15.7%)*	0.706	(+34.5%)*	0.997	(+0.6%)

Table 6.5. Effects of model order on harmonic nodels, MRR values, known item queries



	Previous	Current Only			Previous Three		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	**453000**	453000			453000		
Relevant:	**26403**	26403			26403		
Rel|ret:	**17023**	16375	-3.81	0.0000*	16622	-2.36	0.0000*
Interpolated Recall - Precision							
at 0.00	**0.7805**	0.5816	-25.5	0.0000*	0.8808	+12.9	0.0000*
at 0.10	**0.5561**	0.3137	-43.6	0.0000*	0.6278	+12.9	0.0000*
at 0.20	**0.4204**	0.2449	-41.7	0.0000*	0.4835	+15.0	0.0000*
at 0.30	**0.3310**	0.2006	-39.4	0.0000*	0.3844	+16.1	0.0000*
at 0.40	**0.2748**	0.1742	-36.6	0.0000*	0.3285	+19.5	0.0000*
at 0.50	**0.1586**	0.1168	-26.3	0.0000*	0.1700	+7.2	0.0028*
at 0.60	**0.1212**	0.0807	-33.4	0.0000*	0.1248	+3.0	0.2052
at 0.70	**0.0810**	0.0503	-37.9	0.0000*	0.0836	+3.2	0.2070
at 0.80	**0.0561**	0.0333	-40.7	0.0000*	0.0574	+2.4	0.3973
at 0.90	**0.0385**	0.0179	-53.6	0.0000*	0.0402	+4.6	0.0870
at 1.00	**0.0142**	0.0037	-74.2	0.0000*	0.0170	+19.0	0.0002*
Average precision (non-interpolated) over all rel docs							
	0.2224	0.1306	-41.29	0.0000*	0.2611	+17.45	0.0000*

Figure 6.9. Effects of smoothing on harmonic models, variations

It is not surprising that a higher model order helps retrieval on the "Pure" dataset. The improvement is not statistically significant, but because higher model orders work by memorizing larger and larger chunks of the state sequence history, and because there really is no variation in the known item from the "Pure" dataset, we were confident that higher model orders would yield good results.

Surprising, however, is the fact that higher model orders also significantly improve retrieval on the transcribed audio queries by as much as 15% or more. Again, because of the problem of insertions and deletions not only of notes, but of entire simultaneities, as was mentioned in Section 6.4, we suspected that higher model orders on the transcribed data might fail, as the query model would essentially be memorizing larger chunks of the "wrong" history. But the decoupled partial observation vector smoothing function has once again shown its value. By shifting partial observations from simultaneity to simultaneity, we are able to recapture a better sense of the "true" harmonic sequence. It is also interesting that, in Table 6.5, with maximal smoothing there is no statistically significant difference between 0^{th}-order and 1^{st}-order models with maximum smoothing. Memorizing a bit more history is apparently not as important as getting the partial chord observations "correct".

To insure that it is not just model order giving us this improvement, we offer the following results for 2^{nd} order models with less and less smoothing. Table 6.6 shows that sequence alone is not enough, and is even dangerous, without smoothing. We believe that the decrease in performance on the transcribed queries (70% with normal, previous window-only smoothing, and 98% with no smoothing) further justifies our analysis from Section 6.4.

	GM		JPB		"Pure"	
2^{nd} order, full smoothing	**0.701**		**0.706**		**0.997**	
2^{nd} order, normal smoothing	0.225	(-67.9%)*	0.211	(-70.1%)*	0.976	(-2.1%)*
2^{nd} order, no smoothing	0.011	(-98.4%)*	0.012	(-98.3%)*	0.928	(-6.9%)*

Table 6.6. Higher model order with less smoothing, MRR values, known item queries

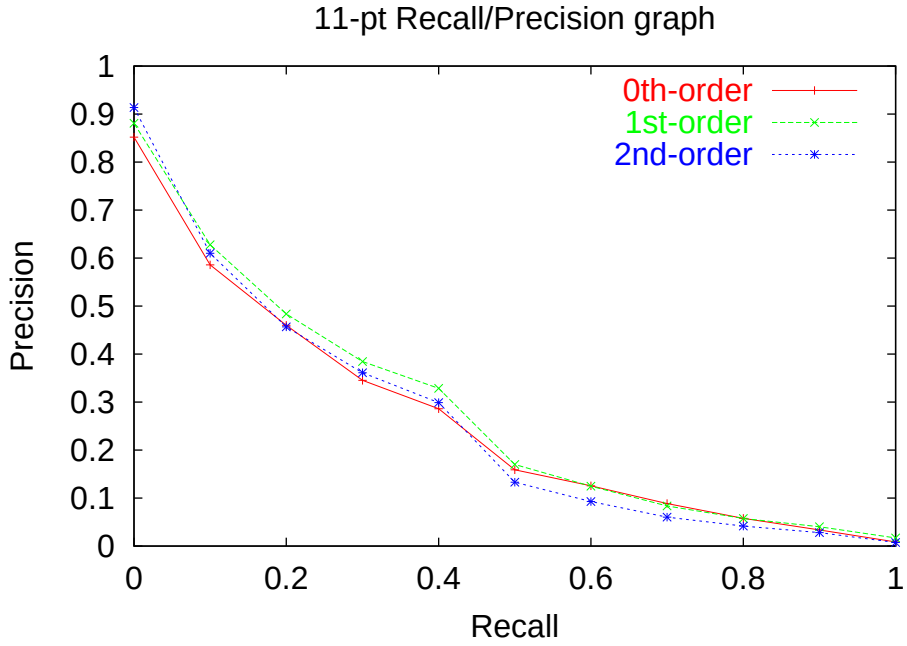
In conclusion, we think these results show that a high model order with no smoothing is not good, a low model order with full smoothing is much better, but a high order model with full smoothing yields the best results. Therefore, these "maximal" models will become our baseline for future improvements.

6.7.2 Variations Queries

In Figure 6.10 we see that the results for the variations queries are not as good as we had hoped. As with the known item queries, 2^{nd}-order models do better, but only at low recall. Precision improves 3.7% at low recall, but declines 7% on average. However, remember that these results are an average of the two transcribed datasets and the one "Pure" dataset. If we separate these query sources, we see that the GM dataset is 5.0% better at low recall, and the JPB dataset is 7.1% better at low recall. Both results are significant. The reason the average is lower is that there is 0% improvement at low recall for the "Pure" dataset; our choice to make a query taken from the collection relevant to itself means that the previous technique already ranks the "query variation" first, and there is no room for improvement. Thus, the +3.7% improvement at low recall shown in Figure 6.10, while already significant, is better than it appears to be.

We conclude that higher-order models are a *precision-enhancing* device. If one is interested in obtaining all variations on a query (recall), one should stick to less "expensive" 1^{st}-order models. However, if one is interested in finding any variation, then 2^{nd}-order models are slightly more valuable.

As in Section 6.7.1, we observe that if higher model orders are used *without* partial observation vector smoothing, results decline significantly. A 2^{nd}-order model with normal (previous window) smoothing drops 26.9% precision at low recall and 34.8% mean average precision. A 2^{nd}-order model with no smoothing (current window only) drops 37.3% at low recall and 58.9% on average. Once



	1^{st}-order **(full smoothing)**	0^{th}-order (full smoothing)			2^{nd}-order (full smoothing)		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	453000	453000			453000		
Relevant:	26403	26403			26403		
Rel|ret:	16622	16257	-2.20	0.0000*	16599	-0.14	0.4795
Interpolated Recall - Precision							
at 0.00	0.8808	0.8519	-3.3	0.0057*	0.9137	+3.7	0.0003*
at 0.10	0.6278	0.5858	-6.7	0.0001*	0.6101	-2.8	0.0326*
at 0.20	0.4835	0.4599	-4.9	0.0255*	0.4566	-5.6	0.0000*
at 0.30	0.3844	0.3453	-10.2	0.0000*	0.3612	-6.0	0.0000*
at 0.40	0.3285	0.2860	-12.9	0.0000*	0.2988	-9.0	0.0000*
at 0.50	0.1700	0.1589	-6.5	0.0237*	0.1331	-21.7	0.0000*
at 0.60	0.1248	0.1256	+0.7	0.8214	0.0927	-25.7	0.0000*
at 0.70	0.0836	0.0887	+6.1	0.0969	0.0604	-27.7	0.0000*
at 0.80	0.0574	0.0577	+0.5	0.9066	0.0417	-27.5	0.0000*
at 0.90	0.0402	0.0336	-16.4	0.0053*	0.0281	-30.1	0.0000*
at 1.00	0.0170	0.0085	-49.8	0.0000*	0.0074	-56.5	0.0000*
Average precision (non-interpolated) over all rel docs							
	0.2611	0.2409	-7.75	0.0000*	0.2425	-7.13	0.0000*

Figure 6.10. Effects of model order on harmonic models, variations

again, it appears that memorizing a bit more history is not as important as getting the partial chord observations relatively "correct".

6.8 Effects of Key-Specific Shrinkage on Harmonic Models

In Section 6.6 we concluded that larger smoothing windows are better. In Section 6.7 we saw that higher model orders are better for Known Item queries, but had mixed results for Variations queries. A careful analysis of all the results across all the parameter setting and across all the queries has shown that some parameter settings work better for some queries and other parameter settings work better for other queries. For example, for the JPB-transcribed Folia queries a 1^{st}-order model smoothed with the previous two windows gives the best results. For the GM-transcribed Twinkle queries a 2^{nd}-order model smoothed with the previous one window gives the best results. For the JPB-transcribed Fugues queries, a 2^{nd}-order model smoothed with the previous three windows gives the best results.

We do not know a priori what type of query or information need a user has in mind; therefore, we will select as our "best" prior known approach 2^{nd}-order harmonic models with full window smoothing, as these parameters have generally been shown to give the best results. For the purposes of evaluation, we give the models estimated with these high parameter setting *maximal* harmonic models. With these maximal models we investigate the effects of various shrinkage approaches on retrieval.

6.8.1 Known Item Queries

Table 6.7 contains the results for the key-specific shrinkage modifications on maximal harmonic models. It is not clear which key-specific approach is better. There is no significant change on the "Pure" dataset, while interpolation performs better on the GM data and Backoff performs better on the JPB data. However, no matter if the shrinkage techique is backoff or interpolation, the key-specificity significantly improves retrieval. We consider this result an unqualified success, especially considering the naivete with which music pieces are fit to the key-specific background model (see Equation 5.13 in Section 5.6.2).

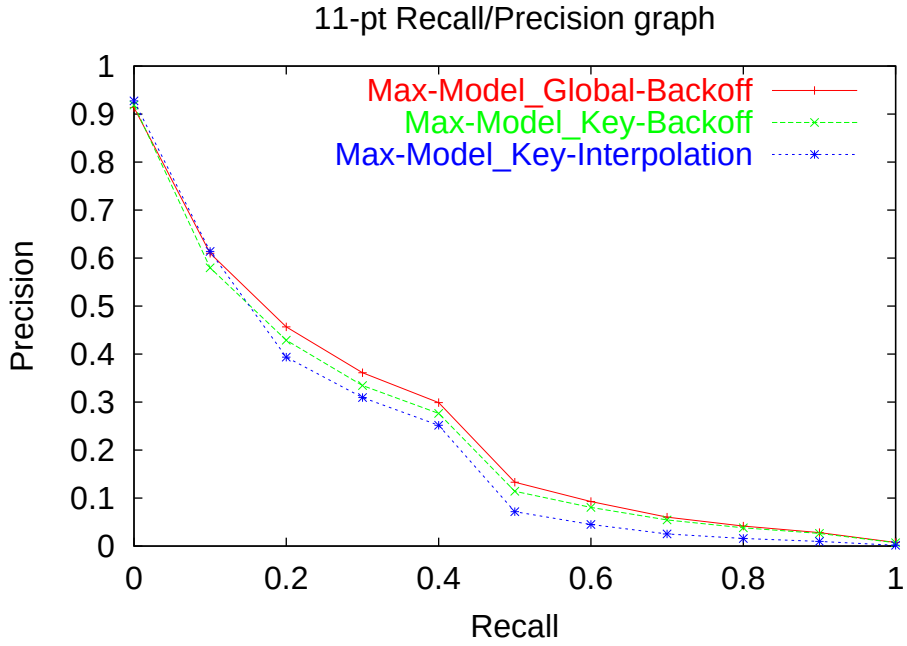
	GM		JPB		"Pure"	
Global Backoff, max model	**0.701**		**0.706**		**0.997**	
Key Backoff, max model	0.829	(+18.3%)*	0.862	(+22.1%)*	1.00	(+0.3%)
Key Interpolation, max model	0.877	(+25.1%)*	0.820	(+16.1%)*	0.974	(-2.3%)

Table 6.7. Effects of shrinkage on harmonic models, MRR values, known item queries

6.8.2 Variations Queries

Unfortunately, the results are not as clear for the variations queries. Key-specific shrinkage only slightly aids precision at low recall and the gains are not statistically significant. Even when we separate the "Pure" query set from the GM and the JPB transcription query sets, because of the slight bias that is present at low recall (see discussion in Section 6.7.2), there is not much difference. For key-specific backoff, the GM queries have a 1.6% improvement at 0% interpolated recall, and the JPB queries have a 0.2% improvement. For key-specific interpolation, the GM and JPB queries show a 2.3% and 2.5% improvement, respectively. Still, none of these improvements are statistically significant. Furthermore, key-specific shrinkage is significantly worse at every other recall point, as well as on average. The results are found in Figure 6.11.

After seeing the large gains obtained by the known item queries using key-specific shrinkage, it was difficult to accept that the technique, by itself, is a poor technique. Therefore, rather than testing the technique on our *maximal* models, we return to the basic harmonic models from Section 6.3.



	Global Backoff	Key-Specific Backoff			Key-Specific Interpolation		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	453000	453000			453000		
Relevant:	26403	26403			26403		
Rel|ret:	16599	16345	-1.53	0.0000*	15160	-8.67	0.0000*
Interpolated Recall - Precision							
at 0.00	0.9137	0.9189	+0.6	0.4341	0.9274	+1.5	0.1824
at 0.10	0.6101	0.5797	-5.0	0.0000*	0.6140	+0.6	0.8310
at 0.20	0.4566	0.4289	-6.1	0.0000*	0.3937	-13.8	0.0009*
at 0.30	0.3612	0.3345	-7.4	0.0000*	0.3094	-14.3	0.0067*
at 0.40	0.2988	0.2763	-7.5	0.0000*	0.2518	-15.7	0.0051*
at 0.50	0.1331	0.1141	-14.3	0.0000*	0.0716	-46.2	0.0000*
at 0.60	0.0927	0.0805	-13.2	0.0000*	0.0451	-51.4	0.0000*
at 0.70	0.0604	0.0545	-9.8	0.0000*	0.0254	-57.9	0.0000*
at 0.80	0.0417	0.0380	-8.7	0.0000*	0.0157	-62.3	0.0000*
at 0.90	0.0281	0.0263	-6.5	0.0000*	0.0096	-65.7	0.0000*
at 1.00	0.0074	0.0069	-7.1	0.0143*	0.0017	-77.3	0.0000*
Average precision (non-interpolated) over all rel docs							
	0.2425	0.2283	-5.87	0.0000*	0.2100	-13.39	0.0002*

Figure 6.11. Key-specific and global shrinkage on max harmonic models, variations

These are 1^{st}-order harmonic models smoothed with only the previous window, and estimates of zero probability are eliminated by backing off to a global music model. If we instead back off or interpolate with a key-specific model, we obtain the results from Figure 6.12. Essentially, these results show that key-specific shrinkage is a precision-enhancing device. The gains at low recall and the losses at high recall are greater for interpolation than backoff, but both techniques yield a mean average net gain in precision.

Whether key-specific shrinkage works by pushing the relevant music pieces closer toward their "correct" probability estimates, or by pushing the non-relevant music pieces further toward their own keys, and thus further from the estimates of the relevant piece, is unclear. However, we should not be surprised that precision is enhanced by this technique, as the known item problem on which the method was already shown to be a success is itself a precision search.

So it is clear now that key-specific shrinkage has some benefits, even for the variations queries. It was unfortunate that it did not improve our previous best models, the maximal models. But there are other manners in which key-specific shrinkage might be useful. We present one more figure in which we compare maximal harmonic models with global backoff against basic harmonic models with key-specific shrinkage. As we see from Figure 6.13, there is no statistically significant difference between the two techniques, whether in Rel|ret, mean average precision, or across most recall-precision points.

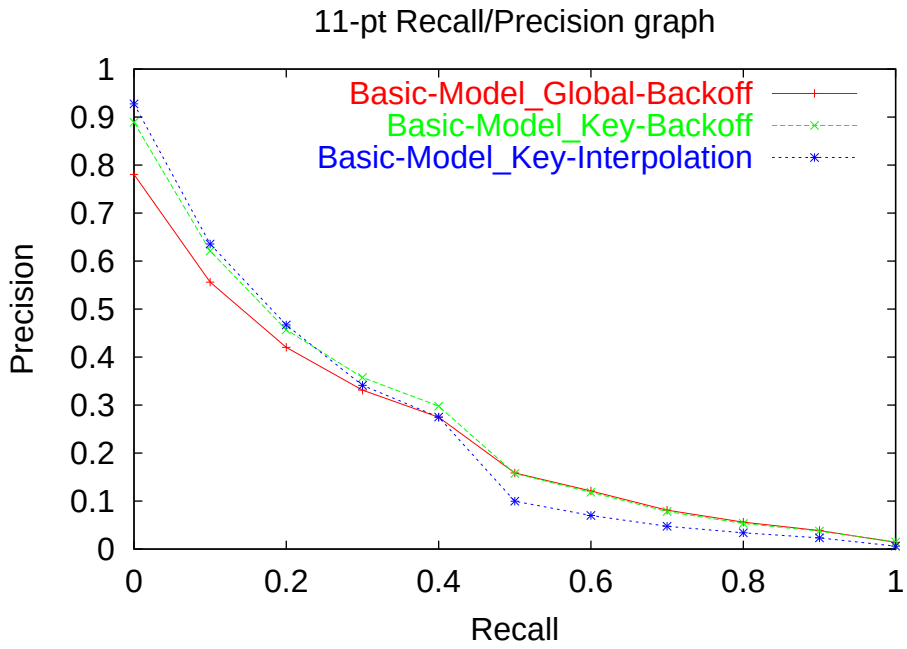
We are still not fully clear why we do not see the overall gains for the variations queries that we did for the known item queries. It is true that there are some variations that are composed in different keys. Key-specific shrinkage makes the assumption that the relevant variation will be found in the same key. By pushing each piece closer to its key-specific estimate, we effectively push certain relevant pieces further from each other, and other non-relevant pieces in the same key closer. This might account for the results we are seeing.

We had hoped that key-specific approaches would improve every query set at every parameter setting. Were this the case, we could unequivocably state that key-based shrinkage techniques should always be used. However, as with many information retrieval techniques, the results are mixed, showing improvement in some areas and not in others. While we were not able to improve overall effectiveness on the variations queries, the fact that key-specific basic harmonic models achieve the same results as globally-shrunk maximal harmonic models means that it is possible to greatly improve efficiency. As the maximal models are 24 times the size of the basic models, we can cut significant time and space complexity from the retrieval system and still obtain similar results. This will be addressed further in Section 6.11.

6.9 Final Quantitative Analysis

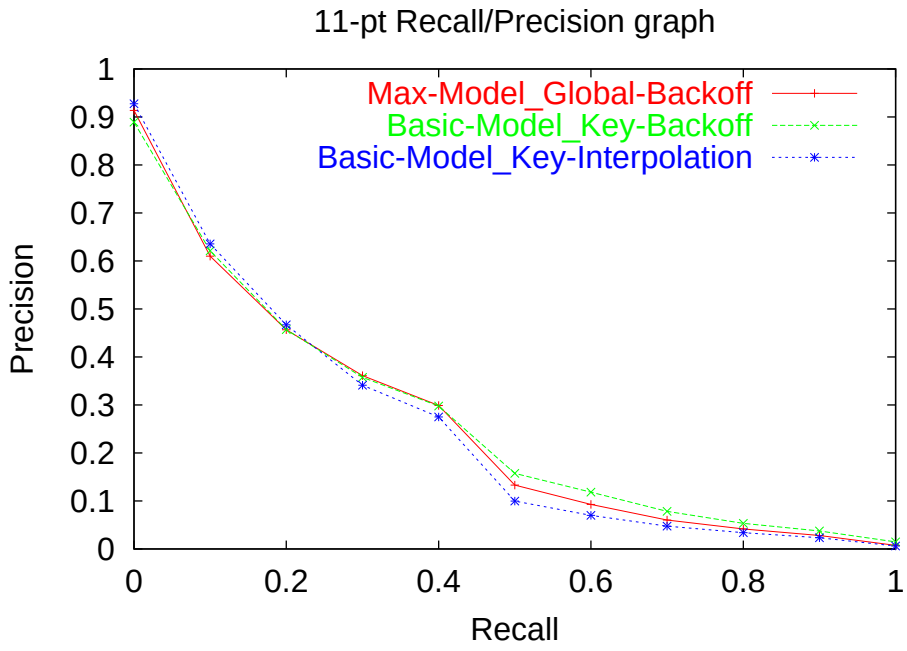
In this chapter we have done a number of different evaluations. We began with hidden Markov models and tested a number of initialization parameter settings. In this section, we wish to summarize these results. We will compare three principal models: the *best hidden Markov models*, the *basic harmonic models*, and the *best harmonic models*.

We define the best hidden Markov models as those with the principled [A] initialization with $Model_1$ [B] initialization. Basic harmonic models were analogous to the HMMs, with one previous window of smoothing on a 1^{st}-order model. We define the best harmonic models as those on which we have applied our full arsenal: full smoothing, highest model order, and key-specific interpolation. For known item queries, the more smoothing, higher model order, and key-shrinkage that we used, the better the results got. For variations queries, high smoothing combined with high model order and global backoff worked equally well as normal smoothing and model order with key-specific shrinkage; using all three together actual decreased results slightly. However, for the purpose of this experiment, we will use these same "best" harmonic models (full smoothing, model order, and key shrinkage) for both known item and variations queries; we do not know a priori which type of query a user has in mind, and therefore should not be able to select the retrieval system parameters arbitrarily.



	Global Backoff	Key-Specific Backoff			Key-Specific Interpolation		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	453000	453000			453000		
Relevant:	26403	26403			26403		
Rel|ret:	17023	16623	-2.35	0.0000*	16714	-1.82	0.3653
Interpolated Recall - Precision							
at 0.00	0.7805	0.8891	+13.9	0.0000*	0.9274	+18.8	0.0000*
at 0.10	0.5561	0.6211	+11.7	0.0000*	0.6356	+14.3	0.0006*
at 0.20	0.4204	0.4569	+8.7	0.0000*	0.4670	+11.1	0.0584
at 0.30	0.3310	0.3574	+8.0	0.0000*	0.3412	+3.1	0.6540
at 0.40	0.2748	0.2979	+8.4	0.0000*	0.2749	+0.0	1.0000
at 0.50	0.1586	0.1575	-0.7	0.6976	0.0998	-37.1	0.0000*
at 0.60	0.1212	0.1183	-2.3	0.1261	0.0700	-42.2	0.0000*
at 0.70	0.0810	0.0783	-3.4	0.0276*	0.0476	-41.3	0.0000*
at 0.80	0.0561	0.0532	-5.2	0.0004*	0.0340	-39.4	0.0000*
at 0.90	0.0385	0.0376	-2.2	0.0522	0.0234	-39.2	0.0000*
at 1.00	0.0142	0.0146	+2.3	0.2370	0.0062	-56.4	0.0000*
Average precision (non-interpolated) over all rel docs							
	0.2224	0.2489	+11.92	0.0000*	0.2363	+6.27	0.1998

Figure 6.12. Key-specific and global shrinkage on basic harmonic models, variations



	Max Model Global Backoff	Basic Model Key-Specific Backoff			Basic Model Key-Specific Interpolation		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	453000	453000			453000		
Relevant:	26403	26403			26403		
Rel|ret:	16599	16623	0.14	0.9493	16714	0.69	0.7752
Interpolated Recall - Precision							
at 0.00	0.9137	0.8891	-2.7	0.1295	0.9274	+1.5	0.3125
at 0.10	0.6101	0.6211	+1.8	0.6584	0.6356	+4.2	0.2933
at 0.20	0.4566	0.4569	+0.1	0.9928	0.4670	+2.3	0.7062
at 0.30	0.3612	0.3574	-1.1	0.8738	0.3412	-5.5	0.4368
at 0.40	0.2988	0.2979	-0.3	0.9664	0.2749	-8.0	0.2772
at 0.50	0.1331	0.1575	18.3	0.1111	0.0998	-25.0	0.0080*
at 0.60	0.0927	0.1183	27.6	0.0385*	0.0700	-24.4	0.0203*
at 0.70	0.0604	0.0783	29.5	0.0493*	0.0476	-21.3	0.0808
at 0.80	0.0417	0.0532	27.7	0.0521	0.0340	-18.3	0.1259
at 0.90	0.0281	0.0376	33.8	0.0189*	0.0234	-16.8	0.1616
at 1.00	0.0074	0.0146	97.5	0.0002*	0.0062	-15.9	0.4030
Average precision (non-interpolated) over all rel docs							
	0.2425	0.2489	2.62	0.6022	0.2363	-2.57	0.6001

Figure 6.13. Global-shrinkage max harmonic models versus key-specific basic harmonic models, variations

6.9.1 Overall Known Item Results

Table 6.15 contains the overall results for the known item queries. The best hidden Markov models are used as the baseline, and percentage improvements (and statistical significance, marked with an asterisk) are based on these values. Also for comparison, we give the value for the expected value of a completely random list ordering. This is not the random initialization of an HMM [A] or [B] parameter that we explored earlier; this is a random ordering of the ranked list.

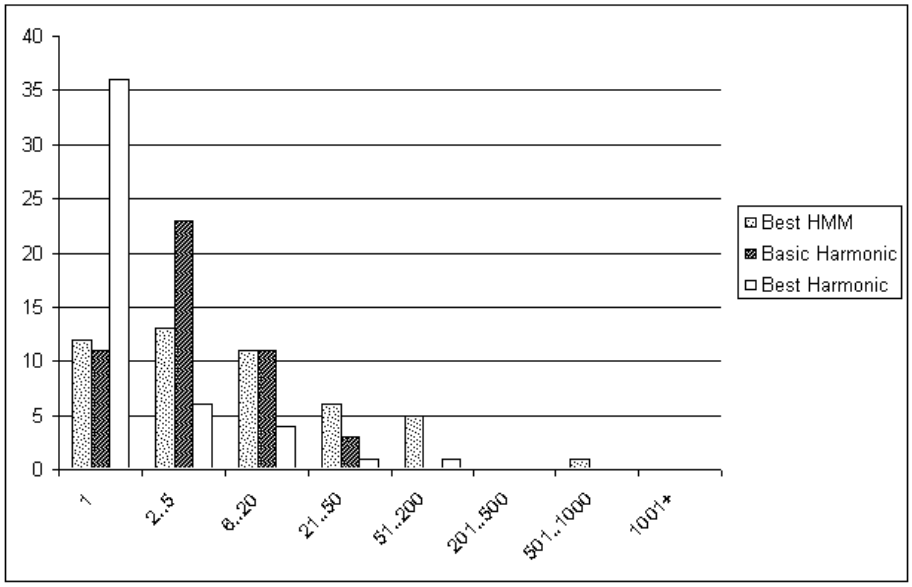


Figure 6.14. Number of queries (y-axis) in which a rank within a given range (x-axis) was obtained, on the JPB known item dataset

	GM		JPB		"Pure"	
Random	0.000635		0.000635		0.000635	
Best HMM	0.118		0.373		0.996	
Basic Harmonic	0.417	(+253.4%)*	0.433	(+16.1%)*	0.993	(-0.3%)
Best Harmonic	0.877	(+643.2%)*	0.820	(+119.8%)*	0.974	(-2.2%)

Figure 6.15. Overall system performance comparisons, MRR values, known item queries

As we can clearly see, HMMs produce good retrieval results. Even though the results are better on the "Pure" dataset, and not as good on the audio transcribed GM and JPB datasets, HMMs are almost twenty *thousand* percent better than random performance. Clearly these models are capturing a credible sense of music similarity. However, harmonic models perform even better. While these models sacrifice some performance on the "Pure" queries, the difference is not statistically significant. However, on the degraded audio-transcribed queries, basic harmonic models sizably outperform HMMs, and the best harmonic models sizably outperform the basic models.

With a mean reciprocal rank of 0.877 on the GM data and 0.82 on the JPB data, this is almost a solved problem, at least for monotimbral polyphonic music. To get an even closer look at this data, Figure 6.14 shows the actual number of queries whose relevant piece was found at rank x. In the interest of space, these ranks have been binned. As we can see, HMMs are able to pull most of the relevant pieces into the top 200. Basic harmonic models are able to pull many more into the top 20 and even the top 5. However, the best harmonic models are able to pull 36 of the 48 queries (75%) to the very first position in the ranked list, with all but 2 remaining queries in the top 20 and top 5. As always, it would be good to test these queries on an even larger collection. But if a user were

guaranteed that almost 90% of the time, the piece they are looking for would be in the top 5, they might be satisfied.

6.9.2 Overall Variations Results

For the variations queries, we once again average the GM, JPB, and "Pure" datasets together in the interest of space. The best hidden Markov models are used as the baseline, and percentage improvements (and statistical significance, marked with an asterisk) are based on these values. Also for comparison, we plot the expected value of a completely random list ordering. The results are found in Figure 6.16.

In the interest of space, we do not present the random ranking values in tabular form. However, as with the known item queries, hidden Markov models significantly outperform random ranking. Rel|ret is 75% greater for HMMs and mean average precision is 1,415% greater. HMM precision is hundreds of percent greater than random at low recall, and much greater at high recall.

So once again, HMMs do capture a credible sense of similarity amongst the real-world, composed variations of a piece of music. And once again harmonic models function even better. Both basic and best harmonic models significantly outperform HMMs at every level of recall and precision. Basic harmonic models retrieve more relevant documents, as witnessed by Rel|ret, while there is no significant difference between HMMs and the best harmonic models.

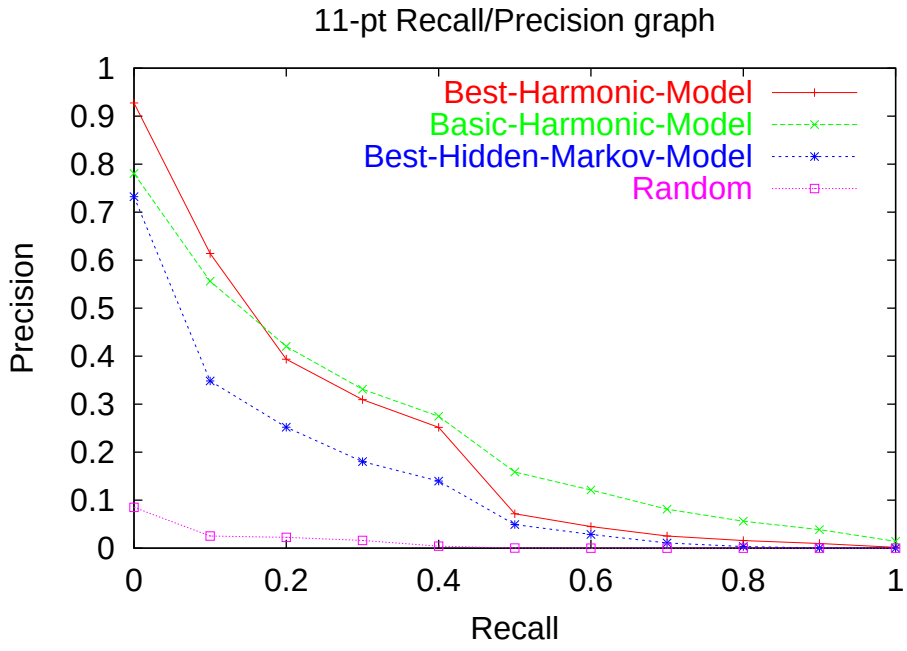
From these results we draw two primary conclusions. The first is that harmonic modeling is an effective approach to polyphonic music information retrieval. Not only does it significantly outperform random ranking, but it significantly outperforms hidden Markov models, which themselves are a viable approach to the problem. The second conclusion is that more smoothing, higher model orders, and key-specific shrinkage are *precision-enhancing* devices. Recall is sacrificed, but the precision at all levels of recall on the best harmonic models is still significantly better than hidden Markov models.

6.10 Qualitative Analysis

It is sometimes useful to see not only what relevant music pieces a retrieval system is able to find, but what non-relevant pieces it finds as well. We select as the object of our case study Variation #5 of the Mozart Twinkle piece, as transcribed from the audio by the JPB algorithm. Table 6.8 shows the top 25 retrieved pieces by three different systems using this query: Best hidden Markov models, basic harmonic models, and best harmonic models. The relevant pieces are listed in bold face. Much of the discussion and analysis in this section we owe to help from Tim Crawford [34].

The first thing we see is that, the original, symbolic version of Variation #5 appears at the top of each list. All three retrieval systems successfully found the symbolic piece from which the raw audio was created. However, only the harmonic modeling systems rank highly other variations on that same piece of music. We are interested not so much in these relevant pieces, however, as we are in the non-relevant pieces. In particular, we wish to see which non-relevant pieces are consistently ranked highly by all three systems. This might offer insight into the types of problems our retrieval methodologies face. Therefore, we created Table 6.9 as an inverted list, or index, of Table 6.8. We only included pieces that were found in two or more of the three retrieval systems. Each piece which did occur more than once is listed alongside its numerical rank in each corresponding ranked list. An asterisk indicates an alternate encoding, as occasionally two different encodings of the same piece of music made their way into the collection.

Two pieces were found in the top 25 by all three retrieval systems: *Bach Cantata 63, pt.4* and *Mozart, Duo for violin and viola, pt.2*. It is a bit of a mystery as to why the Bach piece was retrieved. The piece itself is an example of 'accompanied recitative', but none of the intervals or transitions sounded remotely like those of the Twinkle query. Crawford conjectures that it may be that the crucial chordal transitions fly by in flurries of orchestral interjections (typical of recitative) too quickly to hear individual aural features [34]. We do note, however, that the progressively "better" harmonic models rank this piece increasingly lower.



	Best HMM	Basic Harm			Best Harm		
			%Chg	T-test		%Chg	T-test
Total number of documents over all queries							
Retrieved:	453000	453000			453000		
Relevant:	26403	26403			26403		
Rel|ret:	15122	17023	+12.57	0.0000*	15160	+0.25	0.9065
Interpolated Recall - Precision							
at 0.00	0.7323	0.7805	+6.6	0.0098*	0.9274	+26.6	0.0000*
at 0.10	0.3484	0.5561	+59.6	0.0000*	0.6140	+76.2	0.0000*
at 0.20	0.2519	0.4204	+66.9	0.0000*	0.3937	+56.3	0.0000*
at 0.30	0.1802	0.3310	+83.7	0.0000*	0.3094	+71.7	0.0000*
at 0.40	0.1396	0.2748	+96.9	0.0000*	0.2518	+80.4	0.0000*
at 0.50	0.0491	0.1586	+223.3	0.0000*	0.0716	+46.0	0.0009*
at 0.60	0.0287	0.1212	+322.6	0.0000*	0.0451	+57.3	0.0019*
at 0.70	0.0108	0.0810	+648.9	0.0000*	0.0254	+134.8	0.0006*
at 0.80	0.0032	0.0561	+1677.1	0.0000*	0.0157	+397.0	0.0000*
at 0.90	0.0002	0.0385	+23776.4	0.0000*	0.0096	+5882.7	0.0000*
at 1.00	0.0000	0.0142	undef	0.0000*	0.0017	+undef	0.0006*
Average precision (non-interpolated) over all rel docs							
	0.1267	0.2224	+75.56	0.0000*	0.2100	+65.84	0.0000*

Figure 6.16. Overall system performance comparisons, variations

Query = Twinkle, Mozart Variation 5, JPB audio transcription			
Rank	Best Hidden Markov Model	Basic Harmonic Model	Best Harmonic Model
1	**Twinkle, Mozart Variation 5**	**Twinkle, Mozart Variation 5**	**Twinkle, Mozart Variation 5**
2	Bach, Cantata 63, pt.4	**Twinkle, Hirsch 2nd-Grader**	**Twinkle, Mozart Theme**
3	Bach, Cantata 10, pt.4	Bach, Sinfonia in F Major, pt.3c	**Twinkle, Mozart Variation 2**
4	Corelli, Opus 5, no. 3, pt.5	Handel, Rada no. 53	Corelli, Opus 2 no. 3, pt.1
5	Bach, Cantata 126, pt.4	Lachrimae [OxCh-AltBass]	**Twinkle, Mozart Variation 3**
6	Bach, Orchestral suite in C Major, pt.2	Corelli, Opus 2, no. 3, pt.1	Corelli, Opus 2, no. 3, pt.1, bass line
7	Bach, Passio sec. Matthaeum, pt.41	**Twinkle, Mozart Theme**	Bach, Brandenburg Con. no. 1 in F Major, pt.4
8	Bach, (Two-part) Inventio 1 (in C)	Corelli, Opus 2 no. 3, pt.2	Corelli, Opus 2, no. 3, pt.2
9	Bach, (Two-part) Inventio 1 (in C)*	Bach, Cantata 63, pt.4	**Twinkle, Mozart Variation 4**
10	Bach, WTC Vol.1, Fugue in A minor	Bach, Orchestral suite in C Major, pt.6	Handel, Ott, pt.31*
11	Corelli, Opus 5, no. 7, pt.2	Handel, Clori, Tirsi e Fileno, pt.12*	Haydn, Symphony no. 94, pt.2
12	Mozart, Duo for violin and viola, pt.2	Handel, Ott, pt.31*	**Twinkle, Mozart Variation 11**
13	Handel, Clori, Tirsi e Fileno, pt.27	**Twinkle, Mozart Variation 2**	Mozart, Duo for violin and viola, pt.2
14	Bach, WTC Vol.1, Prelude in C Major	Corelli, Opus 2 no. 3, pt.1, bass line	Bach, Passio sec. Matthaeum, pt.42b
15	Bach, WTC Vol.1, Prelude in C Major*	Mozart, Duo for violin and viola, pt.2	Bach, Four-part Chorales no. 137
16	Corelli, Opus 1, no. 1, pt.2	Beethoven, Symphony no. 1, pt.2	Mozart, Quartet k.170, pt.4
17	Bach, Cantata 175, pt.4	**Twinkle, Mozart Variation 3**	Mozart, Quartet k.155, pt.3
18	Handel, Clori, Tirsi e Fileno, pt.27*	Mozart, Quartet k.170, pt.4	Bach, Four-part Chorales no. 117
19	Bach, WTC Vol.2, Fugue in C Major	**Twinkle, Mozart Variation 4**	Bach, Cantata 15, pt.6c
20	Bach, WTC Vol.2, Fugue in C Major*	Bach, Cantata 31, pt.8	Bach, Cantata 31, pt.8
21	Bach, Cantata 28, pt.5	Corelli, Opus 1, no. 7, pt.1	Corelli, Opus 1, no. 7, pt.1
22	Bach, Cantata 35, pt.4	Mozart, Quartet k.155, pt.3	Bach, Four-part Chorales no. 72
23	Corelli, Opus 6, no. 12, pt.2	Bach, Cantata 31, pt.1	**Twinkle, Mozart Variation 7**
24	Bach, Cantata 61, pt.2	Corelli, Opus 6, no. 9, pt.2	Bach, Cantata 63, pt. 4
25	Bach, Cantata 76, pt.10	Haydn, Symphony no. 94, pt.2	Beethoven, Symphony no. 1, pt.2

Table 6.8. Top 25 retrieved pieces by three different systems

Table 6.9. Inverted lists of ranks for top 25 retrieved pieces

Query = Twinkle, Mozart Variation 5, JPB audio transcription			
Piece	Best Hidden Markov Model	Basic Harmonic Model	Best Harmonic Model
Relevant Pieces			
Twinkle, Mozart Variation 5	1	1	1
Twinkle, Mozart Theme	-	7	2
Twinkle, Mozart Variation 2	-	13	3
Twinkle, Mozart Variation 3	-	17	5
Twinkle, Mozart Variation 4	-	19	9
Non-relevant Pieces			
Bach Cantata 63, pt.4	2	9	24
Mozart, Duo for violin and viola, pt.2	12	15	13
Corelli, Opus 2, no. 3, pt.1	-	6	4
Corelli, Opus 2, no. 3, pt.2	-	8	8
Handel, Ott, pt.31*	-	12	10
Corelli, Opus 2, no. 3, pt.1, bass line	-	14	6
Beethoven, Symphony no. 1, pt.2	-	16	25
Mozart, Quartet k.170, pt.4	-	18	16
Bach, Cantata 31, pt.8	-	20	20
Corelli, Opus 1, no. 7, pt.1	-	21	21
Mozart, Quartet k.155, pt.3	-	22	17
Haydn, Symphony no. 94, pt.2	-	25	11

The second piece found in common by all three systems was Mozart, Duo for violin and viola, pt.2. This piece, along with the *Corelli Opus 3 Number 2*, first and second movements, contains a number of descending suspension chains (dissonance-resolution occurrences) which are highly characteristic not only of Mozart Twinkle Variation #5, but of many of the other Mozart variations, such as #2, #3, #4, and #7, as well. Thus, it is not surprising that either the Mozart duo or the Corelli piece was ranked highly against the Variation #5 query; though they are not "relevant", they do share many similarities in their harmonic patterns.

Beet-sym-01-02 was a piece retrieved by both harmonic modeling systems. If one listens closely to the beginning of the piece, it bears some resemblance to the basic Twinkle theme. The primary difference is that, rhythmically, it has a different signature. We knew this was a consequence of the manner in which we created simultaneities, by ignoring all durational and rhythmic clues. However, even though beet-sym-01-02 is "non-relevant", finding such similar patterns might be interesting to a musicologist and is indicative of the fact that our harmonic modeling techniques do capture a credible sense of music similarity. Even more interesting is the fact that beet-sym-01-02 is in the key of F-Major, even though the original query is in the key of C-Major. Our models do not account for tranposition invariance; we attribute the retrieval of such a similar-sounding piece in a closely related key to the manner in which the harmonic description was created.

Finally, haydnsym-94-02, another piece which was ranked highly by both harmonic models, is a credible Twinkle variation. Parts of this piece sound remarkably similar to the basic Twinkle melody. These same parts also have a very similar rhythmic and phrasal structure as the Twinkle theme. Though we do not attribute this fact to our models, as rhythm and duration information are discarded, we are nevertheless pleased that both harmonic models are able to retrieve this piece. It is one in which a Twinkle "scholar" might be interested, even if not marked relevant by our particular experimental setup.

There are undoubtedly many more comments we could make about these ranked lists, but the purpose of this dissertation is music retrieval rather than music analysis. This case study should therefore serve to illuminate but a few of the possible reasons why and how our systems perform as they do, by examining in detail a few of the pieces retrieved. A complete analysis is beyond the scope of this work.

6.11 Factors other than Recall-Precision

In this chapter we evaluated our systems using the standard Recall-Precision measure introduced by Cleverdon in the Cranfield experiments [31]. However, these measures are not the only factors important for system evaluation. As Reiss [101] also mentions, other factors include the *coverage* of the system (how much relevant matter is included in the system), the *time lag* (how long it takes to answer queries), the form of the system output *presentation*, and the *effort* required from a user to obtain the information he/she is seeking.

Futrelle [48] expressed the coverage issue in another manner. Rather than focusing on how much relevant material is in the collection, he proposes that we should focus on how much *non-relevant* material is present, and whether there is any bias in the collection itself being searched. We have acknowledged the bias in our collection; our collection and our queries consist exclusively of 16^{th} to 19^{th} century baroque, classical, and romantic music. There is no jazz, there is no 20^{th} century music of any kind. Though we believe that our techniques will work on these other genres of music, we have not actually tested this and cannot guarantee it will be so.

Two other factors are the effort required by the user and the presentation of the results. Though we have done some work on this, we have nothing of consequence to report, and thus do not address these issues in this dissertation. Finally, there is the matter of how long it takes a user's query to be answered. This is the subject of Sections 6.11.1 and 6.11.2.

6.11.1 Space Complexity

The first issue we explore is how much storage space each of our models takes. In raw MEF format (our MIDI-like, text-based representation) our 3,000 piece collection uses approximately 82

Megabytes of disk space. When this is modeled using hidden Markov models, the initial state π distribution parameters require $N = 24$ doubles, the [A] state transition distribution requires $N^2 = 576$ doubles, and the observation distribution requires $N \times M = 98{,}280$ doubles. As a separate model is estimated for every piece in the collection, the total disk space required is 2.8 Gigabytes, 3,300% more disk space than the raw data.

Harmonic models, on the other hand, are much smaller. A basic first order model requires only $N = 24$ doubles for the initial/history state distribution π, and $N^2 = 576$ doubles for the state transition distribution [A]. This translates to 14.9 *Mega*bytes of disk space, which is 82% less disk space than the raw data, and 99.5% less than hidden Markov models. 2^{nd}-order harmonic models require $N^3 = 13{,}824$ doubles for the state transition distribution, which becomes 336 Megabytes on disk. This is larger than the original data, but still much less than the HMM approach. Finally 0^{th}-order harmonic models only require $N = 24$ doubles for the state transition distribution. This translate into a huge savings; the entire collection takes only 1.5 Megabytes of disk space.

6.11.2 Time Complexity

As with space complexity, the time complexity of our retrieval algorithms is directly tied to the size of the state space. The time it takes for a system to "answer" a single query is the time it takes for the system to compare that query against every document in the collection. A hidden Markov model takes $O(CN^2t)$ time to run, where C is the size of the collection, N is the size of the state space (24) and t is the length of the (query) observation sequence. The running time for 1^{st}-order harmonic models is comparable. The conditional relative entropy scoring function must compare all $N \times N$ points in the state transition distribution, for a total $O(CN^2)$ running time. A 2^{nd}-order model has $N^2 \times N$ points, for an $O(CN^3)$ running time. Fortunately, N is relatively small. On the other hand, as 0^{th}-order models have no "history", their running time is fast: $O(CN)$.

As the time complexity of the algorithm is an important factor, decisions need to be made. Earlier in this chapter we were able to show that, with proper harmonic partial observation vector smoothing and key-specific shrinkage, some of the lower-order harmonic models archieved results almost as good as the higher-order models, and still much better than the hidden Markov models. If time is an issue, and the user prefers quick answers over slightly higher quality answers, then some of the lower-order models can be used, as they run exponentially faster (in terms of N) than the higher-order models.

There has also been work to cut down the processing time by indexing hidden Markov models in a manner such that obviously non-relevant pieces can be quickly eliminated without having to examine every distribution in the collection [58]. We believe this approach is readily adaptable to visible Markov models, and thus to our harmonic modeling approach, as well, as the indexing is done on the state transition distribution rather than the observation distribution. However, we do not test this. It is also not clear if the indexing would produce exponential speedup, by reducing C to $\log C$, or if it will simply reduce the constant factor. Either way, however, faster processing is always an important consideration, and if it can be done without sacrificing too much precision and recall then it should be explored.

CHAPTER 7

CONCLUSION

There are many sources of variability in music just as there are many methods for determining similarity. In this work we have restricted ourselves to the similarity inherent in the unfolding of multiple pitches over time, that is, melodic or thematic similarity. Rhythmic, timbral, genre and other forms of similarity were not considered. We consider two main sources of variation within the thematic similarity information need: composed arrangements and audio degraded transcriptions.

In support of this goal we developed two systems, one based on hidden Markov models and the other based on visible Markov models estimated using partial observations inspired by music theory, or harmonic models. In both approaches, chords form the basis of the model's state. This allowed us to collapse the multiple overlapping notes in a complex polyphonic music piece to a probability distribution over a one-dimensional sequence. The use of a judiciously selected chord lexicon also allowed for robust accommodation of the heterogenous textures (the pitch values as well as number of notes) characteristic of many of the variations.

While hidden Markov models performed well, basic harmonic models either equaled or outperformed them on every task and data set. Simple and even naive smoothing and shrinkage techniques allowed for significant improvements in precision as well. Furthermore, harmonic models take much less storage space than hidden Markov models and, for the same model order (history window), harmonic models have a retrieval running time smaller than hidden Markov models.

This same harmonic modeling technique that gives us robust, error-tolerant retrieval of known-item queries is also useful for retrieving variations. Indeed, at one level of abstraction, a composed variation can be thought of as an errorful transcription of the original piece. Our harmonic modeling approach succeeded in capturing a degree of invariance (a degree of similarity) across such transcriptions. For these reasons we consider harmonic models an effective foundation for polyphonic music retrieval.

We also consider this work important not only because it is the first system to ever bring together polyphonic query with polyphonic source collection, but because it provides some of the first clear evidence that it is possible to bridge the symbolic/audio gap. Traditionally, music retrieval systems have operated solely within the symbolic or music notation domain, with an inability to generalize to the audio domain, or solely within the audio domain, with an inability to use important symbolic-based music theoretic notions. In the preceeding chapters we have shown that it is possible to start with imperfect transcriptions and still achieve high-grade retrieval results. Our techniques, though far from perfect, are an important first step for crossing the polyphonic audio and symbolic music retrieval divide.

7.1 Future Work I - Harmonic Description

A number of possible extensions of this work may be considered. Besides the obvious and ever-present need for wider-ranging test collections, modifications can be made in the estimation of the models themselves. In this section we propose a few such extensions, all of which relate in some manner to the harmonic description (the vector of partial observations) described in Section 5.2.

7.1.1 Bypassing the Transcription Phase

One useful direction for this work is to bypass the transcription phase for audio music pieces and go directly from audio features to a harmonic description. Part of the difficulty of audio transcription is

determining which pitches, that is, which detected frequencies in the audio signal, are fundamental frequencies and which are harmonic overtones. Rather than splitting the process into two phases, in which harmonic information is first stripped away through transcription and then recaptured through chord sequence modeling, it might be more effective to use the harmonic information already present in a raw audio signal. This work has already shown that perfect transcriptions are not necessary for estimating models that yield high-quality retrieval; perhaps even rawer transcriptions, in the form of basic Fourier transforms on the audio signal would yield good results.

Timbre is intimately related to harmonic transcription (bypassing the actual notes and transcribing a raw audio signal straight to chords or chord distributions). In this work we restricted ourselves to audio that was monotimbral. It was still polyphonic, with more than one note played at a time, but the only instrument playing these notes was the piano. This was necessary because our transcription algorithms use the acoustic properties of the piano to better judge which frequencies are fundamental and which are overtones. It may be possible in the future to disregard timbre. No matter which instrument or combination of instruments is playing which note or combination of notes, a large pattern of both fundamentals and overtones may be enough to obtain useful estimates of chord structure. Thus, retrieval may also be possible with polytimbral, polyphonic music. Further work might even include nonharmonic instruments, such as drums, timpani, marimbas, gongs, or other such percussive instruments, with techniques for smoothing these signals out of the harmonically modeled sequence.

7.1.2 Rhythmically Motivated Smoothing

In Section 5.3 we proposed a naive but effective method for updating or reassessing the partial observation vectors of the harmonic models. Partial observations of chords at timestep i are altered by partial observations of other chords at previous timesteps, in a manner inversely proportional to the distance from the current timestep. Distance, however, is measured in terms of the number of events, which in turn is defined by the number of new onsets.

Generally, we found (in Section 6.6) that the more smoothing we used, the better the retrieval peformance. However, depending on the query set, sometimes smoothing with the previous two windows worked better, sometimes smoothing with the previous three worked better. This was unsatisfying. We suspect that different pieces of music have different rates of change in their harmonic patterns and that onset-based counts are not indicative of this pattern. In other words, a music piece with many quick harmonic shifts might not benefit from larger smoothing windows. A piece with fewer harmonic shifts might not benefit from smaller smoothing windows. Therefore, we would like to develop a technique that adjusts the smoothing window automatically based on intrinsic properties of each individual piece of music rather than on some fixed parameter.

Our idea is that by looking at the duration of the notes themselves as well as the interonset interval, we may gain some clue as to when harmonic shifts occur and how far a particular smoothing window should extend. Smoothing may be increased or decreased at a certain point depending on factors such as metrical stress (whether the simultaneity is on the beat or off the beat) Thus, unlike the case in other retrieval systems in which rhythmic information is explicitly modeled or searched, we want to use rhythmic information as an intelligent, guiding heuristic for windowed harmonic smoothing. Indeed, harmonic smoothing, properly executed, may be a way to integrate the problematic, not-quite-orthogonal dimensions of pitch and duration within a polyphonic source.

Another related idea is to take into account the number of notes present in each simultaneity and modify the size of the smoothing window based on this note count. In this manner, appoggiaturas would be treated differently from single simultaneities in which an entire chord is already present; the algorithm would smooth one more than the other. No matter how it is done, we hope that better smoothing yields a richer harmonic description and better retrieval.

7.1.3 Using HMMs for Simultaneity Selection

Another issue is closely related to the smoothing problem. In Section 3.1 we aligned simultaneities to a music piece by creating a new simultaneity every time there was a new note onset. However, rather

than smoothing over a rhythmically-induced varying window size of these per-onset simultaneities, it might make more sense to vary the actual number of simultaneities. In other words, instead of creating one simultaneity per onset, we might create one simultaneity per n onsets.

We trace this notion back to Mongeau and Sankoff [81], though ultimately what we are trying to do is some sort of intelligent segmentation of a piece of music. If simultaneities themselves are chosen more intelligently, the need for smoothing might not be as great.

In Chapter 3 we explain that the core features of our models are chords. Therefore, it makes sense to create segment boundaries at points in a piece at which transitions between chords are most likely. Pardo [86] uses a greedy, rule-based algorithm to do this. However, we feel it might be useful to also explore hidden Markov approaches, as we have already begun using them in this work. A number of researchers have already used HMMs for chord labeling and/or chord-based segmentation, including Ponsford [91], Sheh [109], and Raphael [100].

Our idea is to use one of these systems to widen the window of notes in which we select a (now slightly misnamed) simultaneity. We would then estimate a second model using these segmented, grouped simultaneities as chunked observation states. The second model could be a harmonic model or a hidden Markov model. Either way, state transition probabilities would be estimated from one chunk of onsets to the next rather than from one onset to the next.

The reason for the second stage of modeling, again, has to do with our analysis from Section 6.4. When building retrieval systems to capture different variations of a single piece of music, those variations may include insertions and deletions of entire simultaneities. For this reason we believe that our hidden Markov models estimated on a per-simultaneity basis did not work as well when used for retrieval. Therefore, if we use an HMM to chunk a group of similar simultaneities together and then use a second HMM to estimate new state transition probabilities between chunks, we might get a better estimate of the state transitions that will be found in a relevant variation.

Our intuition is that models estimated from segmented or chunked simultaneities might yield retrieval results with much better recall than current per-simultaneity HMM results, albeit with slightly worse precision. State transitions estimated using larger raw chunks might capture more relevant variations, again because of the reasons given in the analysis of Section 6.4. They also have the potential to find more nonrelevant pieces as well. However, we will not make any definite claims until we can actually test such systems.

7.2 Future Work II - Transposition Invariance

There are many sources of variation in music. Our harmonic modeling is an attempt to capture similarities in different pieces of music, when those variations are built around relatively stable harmonic patterns. Another important source of variation is key transposition, in which an entire piece of music is shifted up or down by a number of semitones. This has major consequences in our harmonic description in that if the key is shifted, the patterns of lexical chords found at each timestep may vastly differ. Such transpositional variation has the potential to break our retrieval system.

The favored solution to key transposition in monophonic music is to use the semitone interval between contiguous notes rather than the absolute pitches of the notes themselves. One possible solution for polyphony is to use the interval, in steps around the circle from Figure 4.3, between contiguous lexical chords at contiguous timesteps. We count as equivalent all those neighboring (timestep-adjacent) chords that have the same chord interval.

Let us assume that our example lexical chords P, Q, R are arranged circularly. Returning to our example from Figure 5.4, the spread $P \rightarrow Q$ (+1 or -2) is the same as the spread $Q \rightarrow R$ (+1 or -2), so these two chord pairs are equivalent. Furthermore, going up by 1 is the same as going down by the size of the lexical circle minus 1. In a circle with three chords, +1 is the same as -2. In a circle with 24 chords, +1 is the same as -23. We therefore create Markov matrices in terms of the positive intervals only, for simplicity. Figure 7.1 is Figure 5.4 recast as a transposition-invariant model.

Interval	Count Matrix Chord Transitions	Transn. Counts	Count Totals	Markov Model
+0	$P \rightarrow P$ $Q \rightarrow Q$ $R \rightarrow R$	0.44 + 0.16 + 0.40	= 1.00	0.25
+1	$P \rightarrow Q$ $Q \rightarrow R$ $R \rightarrow P$	0.43 + 0.87 + 0.69	= 1.99	0.4975
+2	$P \rightarrow R$ $Q \rightarrow P$ $R \rightarrow Q$	0.63 + 0.17 + 0.21	= 1.01	0.2525

Figure 7.1. Example 0^{th}-order transposition-invariant count matrix and Markov model

Thus, a 1^{st}-order model of absolute-valued chord state transitions becomes a 0^{th}-order model of relative chord-interval transitions. The same number of total simultaneities (two) are used in estimating both the 1^{st}-order absolute model and the 0^{th}-order relative model, but how the simultaneities are used and counted differ.

The goal in developing chord-interval, transposition-invariant models is to be able to capture relevant variations that might exist in a different key. However, researchers have already shown that for monophonic music, transposition-invariant searching also can bring in many more nonrelevant variations together with the relevant variations [77]. We suspect that this will also be the case with polyphonic music.

Although we had begun to explore this issue in earlier work [90], we do not give it full treatment within this dissertation. The primary reason has to do with our test collections. The known item queries have, by definition, no relevant pieces in another key. Of the three remaining variations sets, the Folia has no variations in a different key, the Lachrimae has one or two variations in a different key, and the Twinkle has around 20% of its relevant variations in a different key. Clearly, our current algorithms for determining similarity are not much affected by the existing test collections.

Nevertheless, we present a small, informal study here to determine whether our technique for finding transposed variations might be useful. Our basic transposition invariant model is estimated as follows: (1) standard harmonic descriptions are created; (2) these harmonic descriptions are smoothed using full (previous three window) smoothing; (3) probability estimates using counts of chord intervals, rather than chords, are created for both query and collection pieces; and (4) pieces are ranked using conditional relative entropy of the chord-interval state transition distributions, rather than chord state transition distributions.

	GM		JPB		"Pure"	
1st order	0.606	(+3,969%)*	0.525	(+13,867%)*	0.991	(+7.0%)*
1st order, relative	**0.015**		**0.003**		**0.927**	
2nd order	0.701	(+4,501%)*	0.706	(+17,913%)*	0.997	(+7.5%)*

Table 7.1. Tranposition invariance, MRR values, known item queries

For this study, we use a full smoothing window (previous three simultaneities) on the harmonic description, and we create 1^{st}-order transposition invariant (relative) models. The results for the known item query sets are found in Table 7.1. For comparison, we offer fully smoothed 1^{st}-order and 2^{nd}-order standard harmonic models. The 1^{st}-order standard harmonic model is the same order as our experimentative transposition model, but the 2^{nd}-order model is constructed using the same number of simultaneities (three) as the 1^{st}-order transposition invariant model.

As we can see from these results, in a case such as a known item query in which no relevant pieces exist in other keys, transposition invariant modeling does not work well at all. The standard harmonic models work much better. Performance is still significantly better than random ordering of the result list but not at a level we deem acceptable.

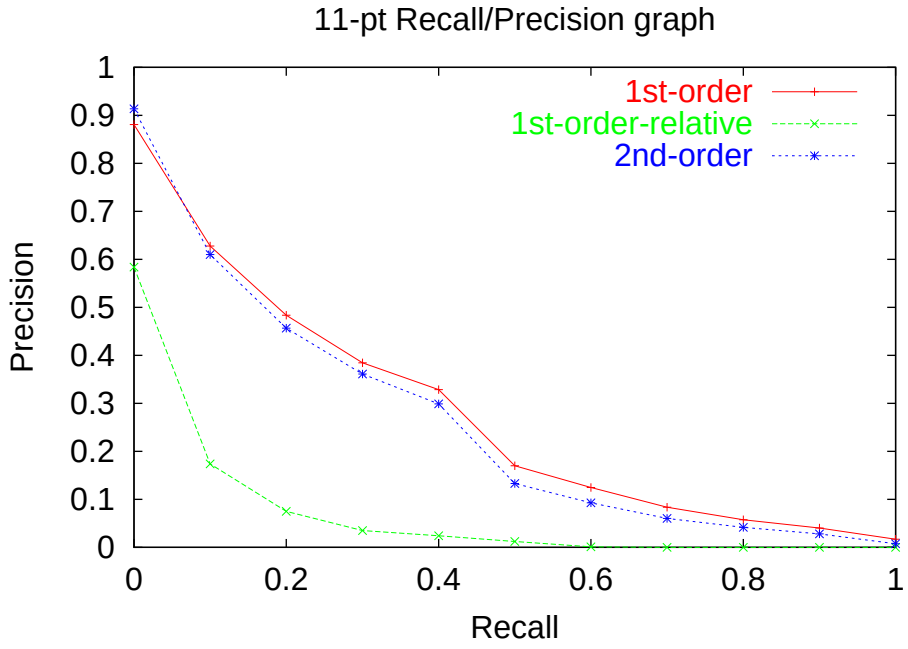


Figure 7.2. Transposition invariance, all variations

Next, we turn our attention to the twinkle, lachrimae, and folia variations queries. The results of this experiment can be found in Figure 7.2. We do not need to show the actual recall-precision values to know that performance also takes a significant hit. Again, these results are better than random but still not very good.

Recall that for the majority of the queries, relevant variations are not found in different keys. The twinkle set was the only one that had a fair number of transposed relevant pieces. The folia set had no transposed relevant pieces. Therefore, we present a subset of the results for these two query sets in Figures 7.3 and 7.4. As we can see, the poor performance on the folia set is undoubtedly exacerbated by there being no transposed relevant variations. While performance on the twinkle set is still not as good for the relative model as on the standard models, we can see from the figure that the performance of the transposition-invariant technique parallels that of the standard techniques. This gives us our first bit of encouragement that some form of transposition invariant modeling may prove useful.

Furthermore, a comparison with the best hidden Markov models from Section 6.3 shows that there is no significant difference between the two approaches. Table 7.2 shows the exact recall-precision breakdown of the two methods on the twinkle query set. The transposition invariant models return slightly fewer pieces in the top 1000 (Rel|ret), but on average, precision is 7.6% better. However, neither of these values, nor any precision values at various interpolated levels of recall, is statistically significant. Essentially, there is no quantitative difference between the two techniques. Thus, the transposition invariant (relative) form of harmonic models might not do as well as the standard harmonic models in general. However, for situations in which relevant pieces exist in transposed form, they do at least as well as the hidden Markov baseline approach. This further encourages us that some form of tranposition invariant modeling would be a useful direction for further research.

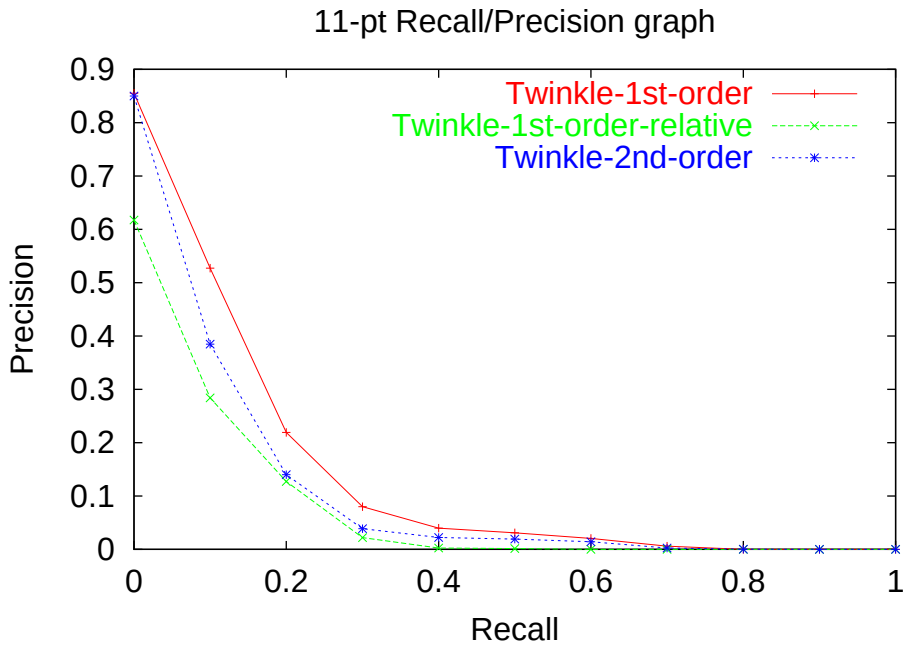


Figure 7.3. Transposition invariance, Twinkle variations

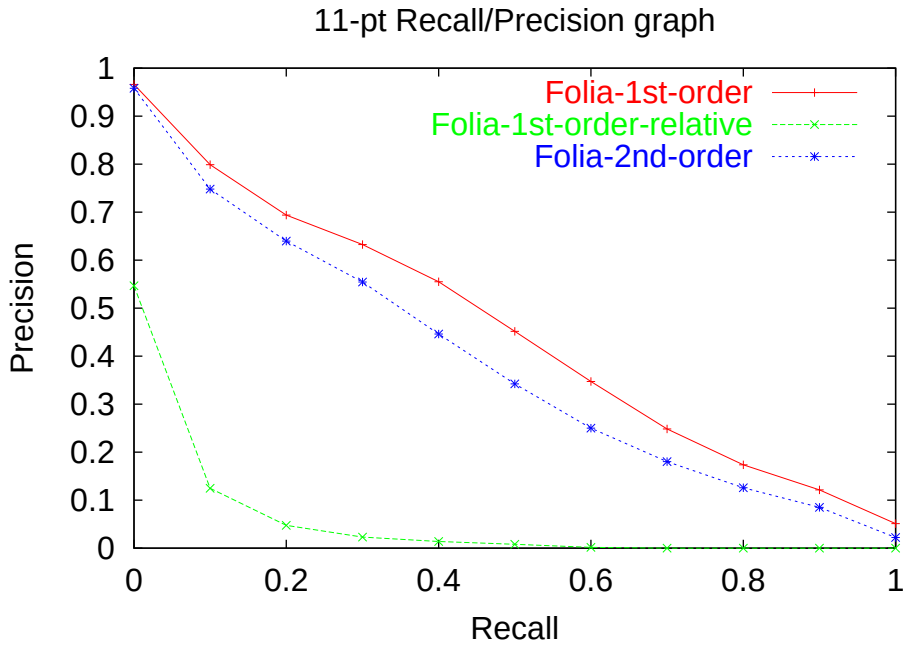


Figure 7.4. Transposition invariance, Folia variations

	Best HMM	1^{st}-order relative model		
			%Chg	T-test
Total number of documents over all queries				
Retrieved:	78000	78000		
Relevant:	2028	2028		
Rel|ret:	613	595	-2.94	0.5829
Interpolated Recall - Precision				
at 0.00	0.6231	0.6175	-0.9	0.8693
at 0.10	0.2820	0.2841	+0.8	0.9226
at 0.20	0.1600	0.1272	-20.5	0.3560
at 0.30	0.0137	0.0220	+60.2	0.3630
at 0.40	0.0050	0.0025	-50.4	0.0585
at 0.50	0.0023	0.0014	-37.7	0.3788
at 0.60	0.0000	0.0000	+0.0	1.0000
at 0.70	0.0000	0.0000	+0.0	1.0000
at 0.80	0.0000	0.0000	+0.0	1.0000
at 0.90	0.0000	0.0000	+0.0	1.0000
at 1.00	0.0000	0.0000	+0.0	1.0000
Average precision (non-interpolated) over all rel docs				
	0.0690	0.0742	+7.62	0.3320

Table 7.2. Comparison of 1^{st}-order tranposition invariant (relative) models with best hidden Markov models, on the Twinkle variations

For one quick final example, we realized at a late stage that our source collection contained another potential query set. BWV 244, or St. Matthew Passion, by Bach, contains a number of different variations on the "O Haupt voll Blut und Wunden" theme. In particular, movements 15, 17, 44, 54, and 62 are all variations on each other. They are also in the keys of E, E$^\flat$, D, F, and C, respectively. This is a perfect candidate query to test transposition invariant modeling.

Figure 7.5 shows the results of this experiment for three experimental configurations: 1^{st}-order absolute, 1^{st}-order transposition invariant (relative) and 2^{nd}-order absolute harmonic models. This time, the transposition invariant setup actually yields the best results. The improvement is almost, but not quite, statistically significant (T-test on the 18.7% mean average precision improvement gives a score of 0.09; statistical significance is taken at 0.05). We attribute this to a relatively small number of queries. So while there is still much room for improvement not only in our transposition invariant modeling, but in our modeling as a whole, this result is nevertheless encouraging.

One particular form of tranposition may bear additional specific attention. Occasionally, variations on a particular piece of music in some major key might occur in the minor key of the same name. For example, the query might be in C major while the relevant variation is in C minor. The harmonic description current differentiates strongly between major and minor harmonies while the transposition invariant version of the harmonic description does not differentiate strongly enough between all 24 major and minor harmonies in every key. A middle ground is needed. Thus, it would be useful to also develop some manner of dealing with this particular major/minor form of transposition.

7.3 Future Work III - Alternativee Applications

Finally, a third main direction in which research into harmonic models may be explored in the future has to do with the applications and information needs to which the models may be applied.

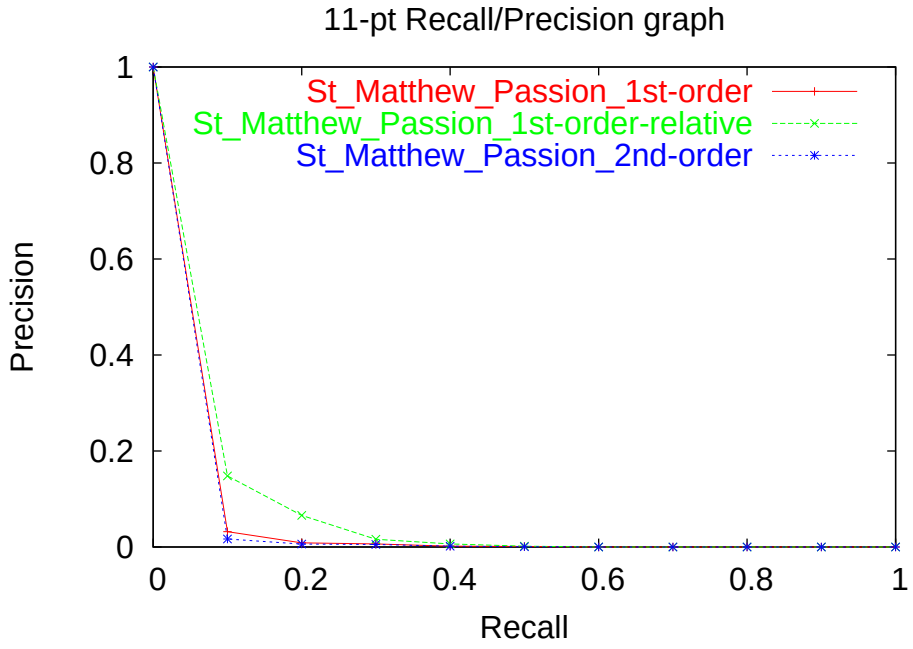


Figure 7.5. Transposition invariance, Bach BWV 244 variations

7.3.1 Passage-level Retrieval

In Section 1.2 we mentioned three basic types of music information needs: finding a known item, finding a variation, and finding a quotation or allusion within one piece of music to another. This dissertation explored the first two information needs but did not do any experiments to test the last need.

Future work might therefore consider this quotation or allusion information need. We propose that one possible way to do this is by building passage-level models of music rather than full-piece models. A single music piece from the collection could be split up into a number of passages, models of each passage could be created, and passages rather than full documents could be ranked by their similarity to a query. Whether these passages are intelligently chosen (in some sort of musically credible manner) or simply enumerated and whether the passages are overlapping or mutually exclusive are future research questions that need to be addressed.

If the harmonic modeling approach is taken once the passages are selected, it may be possible to use existing techniques to obtain better estimates of each passage-level model. For example, instead of shrinking estimates of zero-probability toward a global or key-specific background model, one might do a two-tiered interpolation with the passage estimates first shrunk toward the document model from which the passage was extracted and then shrunk again toward a multidocument (global or key-specific) model. In this manner, each passage model could retain characteristics of the piece from which it was created while also retaining characteristics that make each passage unique.

7.3.2 Classification versus Retrieval

Throughout this work we have made the assumption that the user information need has been to find music that is essentially the same song (the same actual piece of music) as the query. Whether the desired piece is a known item, variation, or passage, the information need is assumed to be melodic or thematic similarity.

Another line of research in the music information retrieval field involves genre-level similarity. Users are not interested in finding the same piece of music but in finding other pieces of music from

the same genre. Pieces of music are not retrieved in the traditional ad hoc ranked-list fashion as much as they are classified or clustered into similar genres.

The most popular application of classification schemes is automated playlist generation. Researchers such as Tzanetakis [117] and Logan [72] and companies such as MoodLogic [83] and Gracenote [50] begin with raw audio music sources and attempt to classify similar pieces by genre or mood, using clues such as spectral histograms, energy, and tempo.

There are many more works that we have not mentioned here that use many other aspects of raw audio signal analysis (see Berenzweig [13] for a more comprehensive literature review). However, we have not seen any work that uses chord structure to augment genric classification. We do not wish to supplant any current approaches with one based purely on chords, but we do think that in addition to timbral or tempo information, chord structure information has the potential to help distinguish between various genres. Many genres often have characteristic chord progressions, such as blues, jazz, flamenco, classic rock, doo wop, folk, ragtime, and rock ballad. Analysis of the chord structure of a raw audio piece may add valuable features to the genre classification problem.

Many researchers in the audio/genre area tend to shy away from pitch- or chord-based features, especially for polyphonic music. The perception is that because the current state of the art does not allow us to perfectly transcribe a piece of music, this information is not useful for audio similarity tasks. We hope that this dissertation begins to change that perception. By showing that even with imperfect transcriptions our chord-based harmonic models are able to achieve good retrieval results, these same techniques may provide useful chord-based genre-distinguishing information as well.

APPENDIX

HARMONIC DESCRIPTION DETAILS AND ERRATA

In Section 3.2, we spend a fair amount of time developing and explaining not only the notion of a chord lexicon, but the reasons for picking our particular lexicon. We will not rehash these intuitions and arguments here. However, we need to mention that, in our early work on harmonic models (see Pickens and Crawford [90] and Pickens et al [89]), there was an additional justification for the use of the 12 major and 12 minor triads as lexical chords: there exists a reliable and easily applied measure of "distance" between any two arbitrary lexicon members which is satisfactory on the grounds of music theory and corresponds with practical experience. This measure is only defined over the 24 major and minor triads. If any more chords are added to the lexicon, this measure is no longer comprehensive.

During the 1970s and 1980s the music-psychologist Carol Krumhansl conducted a groundbreaking series of experiments into the perception and cognition of musical pitch [63]. By using the statistical technique of multi-dimensional scaling on the results of experiments on listeners' judgements of inter-key relationships, she produced a table of coordinates in four-dimensional space which provides the basis for the lexical chord distance measure we adopt here. The distance between triads a and b are expressed (in Equation A.1) as the four-dimensional Euclidean distance between these coordinates, where w_a, x_a, y_a and z_a are the four coordinates of triad a as given in Krumhansl's table (which we do not reproduce here). As Krumhansl demonstrates, her results, derived from controlled listening tests, correspond very closely with music theory.

$$EDist(a,b) = \sqrt{\begin{array}{rcl}(w_a - w_b)^2 & + & (x_a - x_b)^2 \quad + \\ (y_a - y_b)^2 & + & (z_a - z_b)^2\end{array}} \tag{A.1}$$

The intuition in using this metric was that it was necessary for us to account for the presence of all pitches in simultaneity s in terms of the harmonic context of s. Since, in general, contributions of near neighbors in terms of inter-key distance are preferred, we used that fact as the basis for computing a suitable context. We effectively smoothed the partial observation vector by summing contributions to the harmonic description from all members of the lexicon. For each lexical chord c its contribution was proportional to the number of pitches shared with the simultaneity s and inversely proportional to its inter-key distance from the lexical chord p, whose context is being summed. This was given by Equation A.2, the exact form of which is due to Tim Crawford [32].

$$Context(s,c) = \frac{|s \cap c|}{|s|} \sum_{p \in lexicon} \frac{|s \cap p|}{(|s| * EDist(p,c)) + 1} \tag{A.2}$$

Originally, this was going to be the basis for the harmonic description used in this dissertation, until a last minute discovery of a bug in the code. The bug was that our implementation of the Krumhansl distance function did not actually return the correct distance, but instead returned a distance of 1 for every pair of chords. As such, we realized that the actual function we had been using was Equation A.3, as detailed in Section 5.2.1:

$$Context(s,c) = \frac{|s \cap c|}{|s|} \sum_{p \in lexicon} \frac{|s \cap p|}{|s| + 1} \tag{A.3}$$

Unfortunately, this error was introduced at a very early stage of code development. Thus, while it was reported in our early papers ([90] and [89]) that Equation A.2 was used, in actuality, the

correct function behind the results in those papers is Equation A.3. However, we do make clear that the formulae in Chapter 5 of this dissertation verifiably match the results in Chapter 6 of this dissertation. They are just not the formulae we had originally intended to use. But the results do match what is written.

We offer the values in Table A.1 as an example harmonic description (with no neighboring window smoothing) for the notes [c,e,g]. We compare Equation A.2 (Krumhansl) with Equation A.3 (non-Krumhansl). The main difference is that the values for the Krumhansl approach are peaked a little more sharply around the C major and neighboring A minor and E minor triads. The non-Krumhansl approach is slightly flatter in character. It is also interesting to note that, in the Krumhansl approach, a second peak around C minor is starting to form.

Lexical Chord	C	a	F	d	B♭	g
Non-Krumhansl	0.167	0.111	0.056	0	0	0.056
Krumhansl	0.190	0.117	0.054	0	0	0.052
Lexical Chord	E♭	c	A♭	f	C♯	b♭
Non-Krumhansl	0.56	0.111	0.056	0.056	0	0
Krumhansl	0.051	0.115	0.051	0.052	0	0
Lexical Chord	F♯	e♭	B	a♭	E	c♯
Non-Krumhansl	0	0	0	0	0.056	0.056
Krumhansl	0	0	0	0	0.051	0.048
Lexical Chord	A	f♯	D	b	G	e
Non-Krumhansl	0.056	0	0	0	0.056	0.111
Krumhansl	0.050	0	0	0	0.055	0.115

Table A.1. Example harmonic scores for the notes [c,e,g], non-Krumhansl versus Krumhansl

A.1 Preliminary results

We did a few preliminary experiments using the Krumhansl distance metric for our harmonic description. This section details these results.

A.1.1 Known item queries

Table A.2 contains the results of the known item query set. We try to show the effects of the Krumhansl distance metric by comparing the results against the models estimated without this distance metric. For example, the basic harmonic model (first order, previous window smoothing, see Section 6.3) is used as a baseline against which the basic Krumhansl-modified harmonic model (also first order, previous window smoothing, but with the Krumhansl distance metric) is compared. We also increase the size of the smoothing window as well as the model order, and use maximally estimated harmonic models as a second baseline (second order, previous three window smoothing, see Section 6.8). Unless otherwise indicated, the shrinkage used in all models is global. For comparison, we also give the hidden Markov model results.

The first thing we notice is that, for basic harmonic models, mean reciprocal rank is greatly improved using the Krumhansl metric. For almost no additional computation we can use a smarter harmonic description function and improve our ability to retrieve pieces using their audio-degraded transcription as a query.

The second thing we notice is that, for maximal harmonic models, performance also significantly improves. However, it is quite interesting that we get essentially the same results from the Krumhansl measure combined with a global backoff model parameter shrinkage as we do with no Krumhansl measure and a key-specific backoff. This is quite interesting to us, as the key-specific values are themselves estimated using a non-Krumhansl distance metric. Clearly, there is some harmonic

	GM		JPB		"Pure"	
HMM	0.118		0.373		0.996	
Basic Harmonic	0.417		0.433		0.993	
Basic Krumhansl	0.739	(+77.2%)*	0.825	(+90.5%)*	1.0	(+0.7%)
Max Harmonic	0.701		0.706		0.997	
Max Harmonic, Key Backoff	0.829	(+18.3%)*	0.862	(+22.1%)*	0.974	(-2.2%)
Max Krumhansl	0.825	(+17.7%)*	0.867	(+22.8%)*	1.0	(+0.3%)

Table A.2. Effects of Krumhansl distance measure on harmonic models, MRR values, known item queries

information that is being captured both by key-specific shrinkage as well as by the Krumhansl metric. It should prove interesting to try key-specific shrinkage on the Krumhansl models to see whether results continue to improve.

A.1.2 Variations queries

Our next experiment involves the Twinkle, Lachrimae and Folia variations queries. First, we test the basic Krumhansl model against the basic (non-Krumhansl) harmonic models and hidden Markov models. The results of this experiment can be found in Figure A.1. Next, we test the maximal Krumhansl model against the non-Krumhansl maximal harmonic models. The results of this experiment can be found in Figure A.2. (The key-specific shrinkage shrinkage result does not significantly contribute to the discussion here, so we leave it out. See Figure 6.11 on page 79 for more details.)

We see from both of these figures that using the Krumhansl metric is not as useful as it was for the known item queries. Precision at low recall is not affected as much, so it could be that while this metric is indeed giving us better estimates of the "true" chord values, it does so at the expense of the loss of generality or flexibility. Nevertheless, the basic Krumhansl model is still 14% better and the maximal Krumhansl model is still 18% better than hidden Markov models, on average.

A.2 Final comments

These experiments show a number of things. First, we find it quite remarkable that we do not need precise music-theoretic measures of harmony in order to do good music information retrieval. It seems all that is needed are rough estimates and the basic patterns and statistics given by these estimates are enough to distinguish some pieces of music from others.

Another observation we make is that the Krumhansl metric, with its tendency to magnify the sharpness of the harmonic description peaks, is somewhat akin to the hidden Markov $Model_1$ observation distribution, while the non-Krumhansl version of the scoring function is more like $Model_0$. As the reader might recall from Sections 4.3.3 and 6.2.1, $Model_0$ is more general, and enhances recall often at the expense of precision. On the other hand, $Model_1$ is more specific, and enhances precision at the expense of recall. The Krumhansl metric seems to have a similar effect; the more peaked, and less general the harmonic description becomes, the more recall is sacrificed and precision enhanced, and vice versa.

While it is a downside of this work that we did not actually use throughout this dissertation the function we had originally intended, there are a number of silver linings to the cloud. Whether or not we used the Krumhansl metric, our retrieval performed well. We significantly outperform hidden Markov models in almost every case. Clearly there is still utility to approaching the problem in the overall manner we did, even if the specifics of our scoring functions remain "imperfect".

It also might be to our advantage that Krumhansl distances were not necessary to make this approach work. While we still want to continue evaluating their usefulness, knowing that good results are possible without them gives us room to experiment with alternative chord lexicons. For example, if one wanted to add dyads, 7^{th} chords, or augmented/diminished triads to the lexicon, this

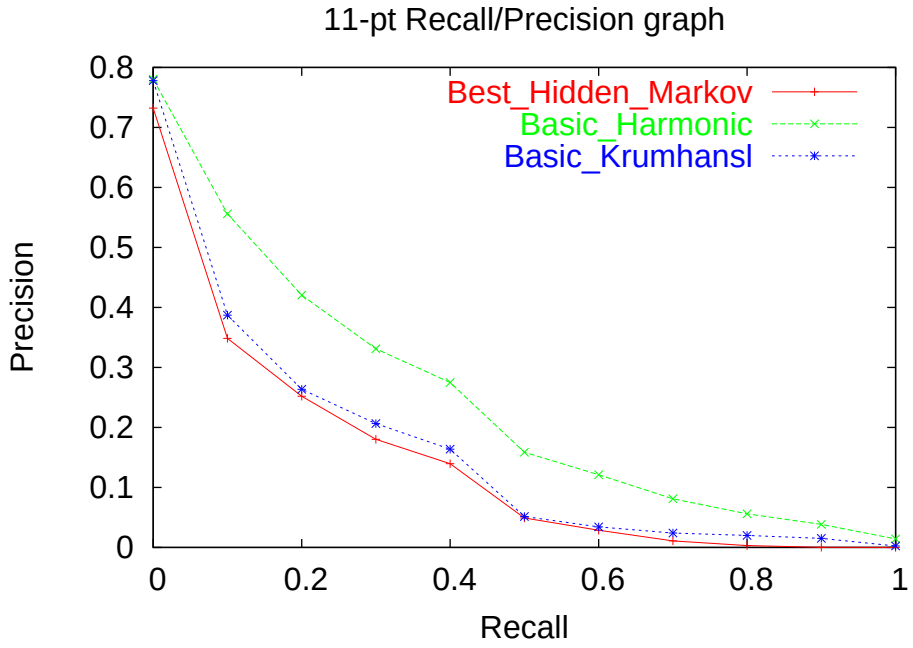


Figure A.1. Krumhansl distance measure with basic harmonic models, variations queries

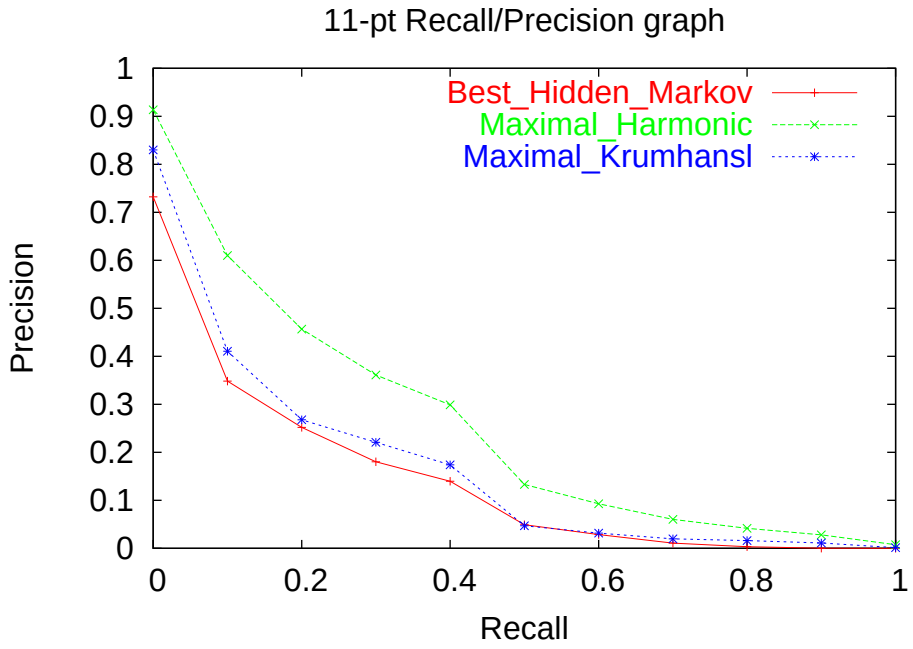


Figure A.2. Krumhansl distance measure with maximal harmonic models, variations queries

would be entirely possible using Equation A.3, as this does not require a similarity metric between members of the lexicon. On the other hand, Equation A.2 and even the hidden Markov approach demand such similarity metrics. We do not have a formal or even emperical distance measure for members of some arbitrary lexicon.

We still stand by our intuition in Section 3.2 that the set of 24 major and minor triads are, in the general case, probably going to be the best choice of a lexicon. But there might be specific cases, such as in a collection of Jazz music, where a different lexicon might yield better results. Equation A.3 gives us the flexibility to experiment with different possibilities. We have not found all the answers yet, but we have tools with which we can explore and evaluate many of the questions.

BIBLIOGRAPHY

[1] Harmonet project. http://i11www.ira.uka.de/~musik/.

[2] The naxos audio collection. http://www.naxos.com, 2002.

[3] AMNS. Nightingale music notation software, 2004. http://www.ngale.com.

[4] Arnold, Denis. Ornaments and ornamentation. In *The New Oxford Companion to Music*. Oxford: Oxford University Press, 1983.

[5] Bainbridge, David. The role of music ir in the new zealand digital library project. In *Proceedings of the first International Symposium on Music Information Retrieval (ISMIR 2000)* (http://ciir.cs.umass.edu/music2000/papers.html, October 2000).

[6] Bakhmutova, V., Gusev, V. D., and Titkova, T. N. The search for adaptations in song melodies. *Computer Music Journal 21*, 1 (1997), 58–67.

[7] Barlow, Harold, and Morgenstern, Sam. *A Dictionary of Musical Themes*. New York: Crown Publishers, 1948.

[8] Barlow, Harold, and Morgenstern, Sam. *A Dictionary of Opera and Song Themes*. New York: Crown Publishers, 1950.

[9] Barthélemy, Jerome, and Bonardi, Alain. Figured bass and tonality recognition. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 129–135.

[10] Bashour, Frederick J. "snake charmer" melody query. Email correspondence, 6 June 2001.

[11] Bello, J. P., Daudet, L., and Sandler, M. B. Time-domain polyphonic transcription using self-generating databases. In *Proceedings of the 112th Convention of the Audio Engineering Society* (Munich, Germany, May 2002).

[12] Bello, J. P., and Sandler, M. B. Blackboard system and top-down processing for the transcription of simple polyphonic music. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-00)* (Verona, Italy, December 7-9 2000).

[13] Berenzweig, Adam, Logan, Beth, Ellis, Daniel P. W., and Whitman, Brian. A large-scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of the 4rd Annual International Symposium on Music Information Retrieval (ISMIR)* (Baltimore, Maryland, October 2003), Holger H. Hoos and David Bainbridge, Eds., pp. 99–105.

[14] Birmingham, William, Dannenberg, Roger, Wakefield, Gregory, Bartsch, Mark, Bykowski, David, Mazzoni, Dominic, Meek, Colin, Mellody, Maureen, and Rand, William. Musart: Music retrieval via aural queries. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 73–81.

[15] Birmingham, William, Pardo, Bryan, Meek, Colin, and Shifrin, Jonah. The musart music-retrieval system. In *D-Lib Magazine* (February 2002). Available at: www.dlib.org/dlib/february02/02contents.html.

[16] Blackburn, Steven, and DeRoure, David. A tool for content-based navigation of music. In *Proceedings of ACM International Multimedia Conference (ACMMM)* (1998).

[17] Blackburn, Steven, and DeRoure, David. Music part classification in content based systems. In *6th Open Hypermedia Systems Workshop* (San Antonio, TX, 2000).

[18] Blackburn, Steven G. *Content Based Retrieval and Navigation of Music*, 1999. Mini-thesis, University of Southampton.

[19] Borges, Jorge Luis. Tlön, uqbar, orbis tertius. In *Collected Fictions*. Penguin Books, 1998.

[20] Burkholder, Peter. Borrowing. In *The New Grove Dictionary of Music and Musicians, 2nd ed.*, S. Sadie, Ed. Macmillan, 2000.

[21] Byrd, Donald. *Music Notation by Computer*. PhD thesis, Computer Science Department, Indiana University, 1984.

[22] Byrd, Donald. http://mypage.iu.edu/ donbyrd/ThemesAndConfoundsNoTabs.txt, July 2000.

[23] Byrd, Donald, and Crawford, Tim. Problems of music information retrieval in the real world. *Information Processing and Management 38* (2001), 249–272.

[24] Callan, J.P., Croft, W. B., and Harding, S. M. The inquery retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications* (1992), pp. 78–83.

[25] Cambouropoulos, Emilios. From midi to traditional music notation. In *In Proceedings of the AAAI'2000 Workshop on Articial Intelligence and Music, 17th National Conf. on Articial Intelligence* (2000).

[26] Camilleri, Lelio. Computational theories of music. In *Computer Representations and Models in Music*, Alan Marsden and Anthony Pople, Eds. Academic Press Ltd., 1992, pp. 171–185.

[27] Charnassé, Hélène, and Stepien, Bernard. Automatic transcription of german lute tablatures: an artificial intelligence application. In *Computer Representations and Models in Music*, Alan Marsden and Anthony Pople, Eds. Academic Press Ltd., 1992, pp. 143–170.

[28] Chen, A. L. P., Chang, M., Chen, J., Hsu, J. L., Hsu, C. H., and Hua, S. Y. S. Query by music segments: An efficient approach for song retrieval. In *Proceedings of IEEE International Conference on Multimedia and Expo* (2000).

[29] Chou, Ta Chun, Chen, Arbee L. P., and Liu, Chih Chin. Music databases: Indexing techniques and implementation. In *Proceedings of IEEE International Workshop in Multimedia DBMS* (1996).

[30] Clausen, M., Engelbrecht, R., Meyer, D., and Schmitz, J. Proms: A web-based tool for searching in polyphonic music. In *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR 2000)* (http://ciir.cs.umass.edu/music2000/papers.html, October 2000).

[31] Cleverdon, C. W., Mills, J., and Keen, M. *Factors Determining the Performance of Indexing Systems, Volume I - Design, Volume II - Test Results*. ASLIB Cranfield Project, Cranfield; Reprinted in Sparck Jones & Willett, Readings in Information Retrieval, 1966.

[32] Crawford, Tim. Personal communication, 2000-2003.

[33] Crawford, Tim. Email correspondence, 7 September 2001.

[34] Crawford, Tim. Email correspondence, January 2004.

[35] Crawford, Tim, Iliopoulos, Costas S., and Raman, Rajeev. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology 11* (1998), 73–100.

[36] Crochemore, Maxime, Iliopoulos, Costas S., and Pinzon, Yoan J. Computing evolutionary chains in musical sequences. *Electronic Journal of Combinatorics* (2000).

[37] Dannenberg, Roger B. A brief survey of music representation issues, techniques, and systems. *Computer Music Journal 17*, 3 (1993), 20–30.

[38] David Meredith, Kjell Lemstrm, and Wiggins, Geraint A. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research 31*, 4 (2002), 321–345.

[39] Dixon, S. E. On the computer recognition of solo piano music. *Mikropolyphonie 6* (2000).

[40] Doraisamy, Shyamala, and Rüger, Stefan M. An approach toward a polyphonic music retrieval system. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 187–193.

[41] Dovey, Matthew. An algorithm for locating polyphonic phrases within a polyphonic piece. In *Proceedings of AISB Symposium on Musical Creativity* (Edinburgh, April 1999), pp. 48–53.

[42] Dovey, Matthew. A technique for "regular expression" style searching in polyphonic music. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 179–185.

[43] Dovey, Matthew, and Crawford, Tim. Heuristic models of relevance ranking in searching polyphonic music. In *Proceedings of Diderot Forum on Mathematics and Music* (Vienna, Austria, 1999), pp. 111–123.

[44] Dowling, W. Scale and contour: Two components of a theory of memory for melodies. *Computers and the Humanities 16* (1978), 107–117.

[45] Downie, J. Stephen. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text.* PhD thesis, University of Western Ontario, Faculty of Information and Media Studies, July 1999.

[46] Foote, Jonathan. Arthur: Retrieving orchestral music by long-term structure. In *Proceedings of the 1st International Symposium for Music Information Retrieval (ISMIR)* (Plymouth, Massachusetts, October 2000). See http://ciir.cs.umass.edu/music2000.

[47] Freitag, Dayne, and McCallum, Andrew Kachites. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction* (1999).

[48] Futrelle, Joe. Three criteria for the evaluation of music information retrieval techniques against collections of musical material. In *The MIR/MDL Evaluation Project White Paper Collection (Edition #2)* (http://music-ir.org/evaluation/wp.html, 2002), J. Stephen Downie, Ed., pp. 20–22.

[49] Ghias, A., Logan, J., Chamberlin, D., and Smith, B.C. Query by humming - musical information retrieval in an audio database. In *Proceedings of ACM International Multimedia Conference (ACMMM)* (San Francisco, CA, 1995), pp. 231–236.

[50] http://www.gracenote.com.

[51] Hoos, Holger H., Renz, Kai, and Görg, Marko. Guido/mir - an experimental music information retrieval system based on guido music notation. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 41–50.

[52] Hsu, J. L., Liu, C. C., and Chen, A. L. P. Efficient repeating pattern finding in music databases. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)* (1998).

[53] http://en.wikipedia.org/wiki/DNA_motif#Discovery_of_DNA_motifs. Wikipedia: The free encyclopedia, 2003.

[54] http://www.musedata.org. The musedata collection, 1999. Center for Computer Assisted Research in the Humanities (Stanford, CA).

[55] Huron, David. Humdrum file format. http://www.music-cog.ohio-state.edu/Humdrum/.

[56] Iliopoulos, C.S., Kumar, M., Mouchard, L., and Venkatesh, S. Motif evolution in polyphonic musical sequences. In *Proceedings of the 11th Australasian Workshop on Combinatorial Algorithms (AWOCA)* (University of Newcastle, NSW, Australia, August 2000), L. Brankovic and J. Ryan, Eds., pp. 53–66.

[57] Iliopoulos, C.S., Lecroq, T., Mouchard, L., and Pinzon, Y. J. Computing approximate repetitions in musical sequences. In *Proceedings of Prague Stringology Club Workshop PSCW'00* (2000).

[58] Jin, Hui, and Jagadish, H. V. Indexing hidden markov models for music retrieval. In *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)* (IRCAM - Centre Pompidou, Paris, France, October 2002), Michael Fingerhut, Ed., pp. 20–24.

[59] Junglejane−ga. Origin of the bell tower tune, 2002. Posted 10 April 2002 to Google Answers (http://answers.google.com).

[60] Kalt, Thomas. A new probabilistic model of text classification and retrieval. Tech. Rep. TR98-18, Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 1996.

[61] Klapuri, A. P. Automatic transcription of music. Master's thesis, Tampere Univeristy of Technology, 1998.

[62] Kornstädt, Andreas. Themefinder: A web-based melodic search tool. *Computing in Musicology 11* (1998), 231–236.

[63] Krumhansl, C. L. *Cognitive Foundations of Musical Pitch.* Oxford University Press, New York, 1990.

[64] Lavrenko, Victor, and Pickens, Jeremy. Polyphonic music modeling with random fields. In *Proceedings of ACM Multimedia Conference* (Berkeley, CA, November 2003).

[65] Lee, W., and Chen, A. L. P. Efficient multi-feature index structures for music data retrieval. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases* (2000).

[66] Leman, Marc. Tone context by pattern integration over time. In *Readings in Computer-Generated Music*, D. Baggi, Ed. Los Alamitos: IEEE Computer Society Press, 1992.

[67] Lemström, Kjell, and Laine, P. Musical information retrieval using music parameters. In *Proceedings of the International Computer Music Conference (ICMC)* (Ann Arbor, MI, October 1998), pp. 341–348.

[68] Lemström, Kjell, Laine, Pauli, and Perttu, Sami. Using relative interval slope in music information retrieval. In *Proceedings of the International Computer Music Conference (ICMC)* (Beijing, China, October 1999), pp. 317–320.

[69] Lemström, Kjell, and Tarhio, J. Searching monophonic patterns within polyphonic sources. In *Proceedings of the RIAO Conference* (College of France, Paris, April 2000), vol. 2, pp. 1261–1278.

[70] Lindsay, Adam T. Using contour as a mid-level representation of melody. Master's thesis, MIT Media Lab, 1996.

[71] Liu, C. C., Hsu, J. L., and Chen, A. L. P. Efficient theme and non-trivial repeating pattern discovering in music databases. In *Proceedings of IEEE International Conference on Research Issues in Data Engineering (RIDE)* (1999).

[72] Logan, Beth, and Salomon, Ariel. A music similarity function based on signal analysis. In *IEEE International Conference on Multimedia and Expo* (Tokyo, Japan, August 22-25 2001).

[73] Manning, Christopher D., and Schütze, Hinrich. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.

[74] Marolt, M. Transcription of polyphonic piano music with neural networks. In *Information technology and electrotechnology for the Mediterranean countries. Vol. 2, Signal and image processing : Proceedings, MEleCon 2000, 10th Mediterranean Electrotechnical Conference* (Cyprus, May 29-31 2000).

[75] Marsden, Alan. Modelling the perception of musical voices. In *Computer Representations and Models in Music*, Alan Marsden and Anthony Pople, Eds. Academic Press Ltd., 1992, pp. 239–263.

[76] Martin, K. A blackboard system for automatic transcription of simple polyphonic music. Tech. Rep. 385, Perceptual Computing Section, MIT Media Laboratory, 1996.

[77] McNab, Rodger J., Smith, Lloyd A., Bainbridge, David, and Witten, Ian H. The new zealand digital library melody index. In *D-Lib Magazine* (May 1997). Available at: www.dlib.org/dlib/may97/meldex/05witten.html.

[78] Melucci, Massimo, and Orio, Nicola. Musical information retrieval using melodic surface. In *Proceedings of ACM Digital Libraries* (Berkeley, CA, 1999).

[79] Meredith, D., Wiggins, G.A., and Lemström, K. Pattern induction and matching in polyphonic music and other multi-dimensional datasets. In *the 5th World Multi-Conference on Systemics, Cybernetics and Informatics* (Orlando, 2001), pp. 61–66.

[80] http://www.midi.org.

[81] Mongeau, M., and Sankoff, D. Comparison of musical sequences. *Computers and the Humanities 24* (1990), 161–175.

[82] Monti, Giuliano, and Sandler, Mark. Pitch locking monophonic music analysis. In *Proceedings of the 112th Convention of the Audio Engineering Society (AES)* (Munich, Germany, May 10-13 2002).

[83] http://www.moodlogic.com.

[84] Narmour, E. *The Analysis and Cognition of Basic Melodic Structures*. The University of Chicago Press, Chicago, 1990.

[85] Pardo, Bryan, and Birmingham, William. Chordal analysis of tonal music. Tech. Rep. CSE-TR-439-01, Electrical Engineering and Computer Science Department, University of Michigan, 2001.

[86] Pardo, Bryan, and Birmingham, William P. Algorithms for chordal analysis. *Computer Music Journal 26*, 2 (2002), 27–49.

[87] Paulus, Jouni, and Klapuri, Anssi. Measuring the similarity of rhythmic patterns. In *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)* (IRCAM - Centre Pompidou, Paris, France, October 2002), Michael Fingerhut, Ed., pp. 150–156.

[88] Pickens, Jeremy. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *Proceedings of the 1st International Symposium for Music Information Retrieval (ISMIR)* (October 2000). See http://ciir.cs.umass.edu/music2000.

[89] Pickens, Jeremy, Bello, Juan Pablo, Monti, Giuliano, Crawford, Tim, Dovey, Matthew, Sandler, Mark, and Byrd, Donald. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)* (IRCAM - Centre Pompidou, Paris, France, October 2002), Michael Fingerhut, Ed., pp. 140–149.

[90] Pickens, Jeremy, and Crawford, Tim. Harmonic models for polyphonic music retrieval. In *Proceedings of the ACM Conference in Information Knowledge and Management (CIKM)* (McLean, Virginia, November 2002).

[91] Ponsford, Dan, Wiggins, Geraint, and Mellish, Christopher. Statistical learning of harmonic movement. *Journal of New Music Research* (1999).

[92] Ponte, Jay Michael. *A Language Modeling Approach to Information Retrieval*. PhD thesis, University of Massachusetts Amherst, 1998.

[93] Prather, Ronald E. Harmonic analysis from the computer representation of a musical score. *Communications of the ACM 39*, 12 (1996), 119. See: Virtual Extension Edition of CACM.

[94] Purwins, Hendrik, Blankertz, Benjamin, and Obermayer, Klaus. A new method for tracking modulations in tonal music in audio data format. citeseer.nj.nec.com/purwins00new.html.

[95] Rabiner, Lawrence R. Tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (February 1989), 257–286.

[96] Rand, William, and Birmingham, William. Statistical analysis in music information retrieval. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 25–26.

[97] Raphael, Christopher. Automated rhythm transcription. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 99–107.

[98] Raphael, Christopher. Automatic transcription of piano music. In *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)* (IRCAM - Centre Pompidou, Paris, France, October 2002), Michael Fingerhut, Ed., pp. 15–19.

[99] Raphael, Christopher. Personal communication, January 2003.

[100] Raphael, Christopher, and Stoddard, Josh. Harmonic analysis with probabilistic graphical models. In *Proceedings of the 4rd Annual International Symposium on Music Information Retrieval (ISMIR)* (Baltimore, Maryland, October 2003), Holger H. Hoos and David Bainbridge, Eds., pp. 177–181.

[101] Reiss, Josh, and Sandler, Mark. Beyond recall and precision: A full framework for mir system evaluation. In *The MIR/MDL Evaluation Project White Paper Collection (Edition #2)* (http://music-ir.org/evaluation/wp.html, 2002), J. Stephen Downie, Ed., pp. 58–63.

[102] Roads, Curtis. Grammars as representations for music. In *Foundations of Computer Music*, Curtis Roads and John Strawn, Eds. Cambridge, MA: MIT Press, 1985, pp. 403–442.

[103] Rolland, P.-Y., Raskinis, G., and Ganascia, J.-G. Musical content-based retrieval: An overview of the melodiscov approach and system. In *Proceedings of ACM International Multimedia Conference (ACMMM)* (Orlando, FL, 1999).

[104] Salton, Gerard. *Automatic Text Processing*. Addison-Wesley, 1989.

[105] Schaffrath, Helmut. Representation of music scores for analysis. In *Computer Representations and Models in Music*, Alan Marsden and Anthony Pople, Eds. Academic Press Ltd., 1992, pp. 95–109.

[106] Schenker, H. *Free Composition*. Longman, New York, 1979.

[107] Selfridge-Field, Eleanor. Conceptual and representational issues in melodic comparison. *Computing in Musicology 11* (1998), 3–64.

[108] Shalev-Shwartz, Shai, Dubnov, Shlomo, Friedman, Nir, and Singer, Yoram. Robust temporal and spectral modeling for query by melody. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)* (August 11-15 2002), pp. 331–338.

[109] Sheh, Alexander, and Ellis, Daniel P. W. Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the 4rd Annual International Symposium on Music Information Retrieval (ISMIR)* (Baltimore, Maryland, October 2003), Holger H. Hoos and David Bainbridge, Eds., pp. 183–189.

[110] Shifrin, J., Pardo, B., Meek, C., and Birmingham, W. Hmm-based musical query retrieval. In *Joint Conference on Digital Libraries* (Portland, Oregon, July 14-18 2002).

[111] Shifrin, Jonah, and Birmingham, William P. Effectiveness of hmm-based retrieval on large databases. In *Proceedings of the 4rd Annual International Symposium on Music Information Retrieval (ISMIR)* (Baltimore, Maryland, October 2003), Holger H. Hoos and David Bainbridge, Eds., pp. 33–39.

[112] Shmulevich, I., and Coyle, E. J. The use of recursive median filters for establishing the tonal context in music. In *Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing* (Mackinac Island, MI, 1997).

[113] Shmulevich, I., Yli-Harja, O., Coyle, E., Povel, D.J., and Lemström, K. Perceptual issues in music pattern recognition - complexity of rhythm and key find. *Computers and the Humanities 35*, 1 (2001), 23–35. Appeared also in the Proceedings of the AISB'99 Symposium on Musical Creativity.

[114] Stolcke, Andreas. Entropy-based pruning of backoff language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop* (Lansdowne, Virginia, February 8-11 1998).

[115] Stone, Kurt. *Music Notation in the Twentieth Century: A Practical Guidebook*. New York: W. W. Norton, 1980.

[116] Tseng, Yuen-Hsien. Content-based retrieval for music collections. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)* (Berkeley, CA, 1999).

[117] Tzanetakis, George, Essl, Georg, and Cook, Perry. Automatic musical genre classification of audio signals. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)* (Indiana University, Bloomington, Indiana, October 2001), J. Stephen Downie and David Bainbridge, Eds., pp. 205–210.

[118] Uitdenbogerd, A.L., and Zobel, J. Manipulation of music for melody matching. In *Proceedings of ACM International Multimedia Conference (ACMMM)* (1998), ACM, ACM Press.

[119] Uitdenbogerd, A.L., and Zobel, J. Melodic matching techniques for large music databases. In *Proceedings of ACM International Multimedia Conference (ACMMM)* (Orlando, Florida, USA, Oct. 1999), ACM, ACM Press.

[120] Xu, Jinxi, Broglio, J., and Croft, W.B. The design and implementation of a part of speech tagger for english. In *University of Massachusetts Technical Report IR-52* (1994). http://ciir.cs.umass.edu/publications/index.html.

[121] Zhai, Chengxiang, and Lafferty, John. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the ACM Conference in Information Knowledge and Management (CIKM)* (November 5-10 2001), pp. 403–410.

[122] Zhai, Chengxiang, and Lafferty, John. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)* (September 9-13 2001), pp. 334–342.

[123] Zhu, Yongwei, and Kankanhalli, Mohan. Music scale modeling for melody matching. In *ACM Multimedia* (Berkeley, California, November 4-6 2003).