Music Information Retrieval Technology

A thesis submitted for the degree of

Doctor of Philosophy

Alexandra L. Uitdenbogerd B.Sc.(UWA), Grad. Dip. Ed. (Melb), Grad Cert IT (RMIT),

Department of Computer Science,

RMIT University,

Melbourne, Victoria, Australia.

Supervisors: Justin Zobel, Hugh Williams

May 18, 2002

Declaration

This thesis contains work that has not been submitted previously, in whole or in part, for any other academic award and is solely my original research, except where acknowledged. The work has been carried out since the beginning of my candidature on 30 January 1997.

Alexandra L. Uitdenbogerd
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
May 18, 2002

Acknowledgements

A Ph.D thesis is a major undertaking that is not done in isolation. I am greatly indebted to my supervisors Associate Professor Justin Zobel, and Dr Hugh Williams, who have helped me to achieve a high quality of work.

I have been greatly encouraged by the people who have shown a strong interest in my work. I would like to thank Philippe Aigrain in particular for his encouragement.

I have received support from other academics who have been there at times of need. I'd especially like to thank Associate Professors Zahir Tari and Lin Padgham, and Dr M. V. Ramakrishna for their moral support and encouragement. I thank Sheila Howell for her kind consideration in the final stages of my work. I'd also like to thank Corinna Ng, who advised me in the early stages of my work, and Dr Gerald Harnett for his ongoing interest in my progress.

I don't know if I would have attempted this degree if it weren't for my wonderful family who raised me to believe I could achieve whatever I set my mind to, and the inspirational aunties and cousins who led the way in striving further in their careers.

This journey has not been travelled alone. I've been grateful for the companionship of other research students in the department during my studies. In particular I'd like to thank Greg Craske, and all the research students at MDS.

Some people contributed directly to the work of this thesis. I'd like to thank Abhijit Chattaraj for his excellent work developing the JAM system, and all the volunteers that spent time taking part in my experiments.

Besides my senior supervisor there is one person who has been crucial to my success in this endeavour. John Harnett has provided ongoing system administration support, occasional proof-reading, and has helped keep the household together when necessary. I remain eternally grateful for your patience and assistance.

Early versions of some of the work in this thesis were published as papers co-authored with my supervisor [132, 133]. This work was partly supported by an Australian Postgraduate Award.

Contents

List of Figures

List of Tables

Summary

The field of Music Information Retrieval research is concerned with the problem of locating pieces of music by content, for example, finding the best matches in a collection of music to a particular melody fragment. This is useful for applications such as copyright-related searches.

In this work we investigate methods for the retrieval of polyphonic music stored as musical performance data using the MIDI standard file format. We devised a three-stage approach to melody matching consisting of melody extraction, melody standardisation, and similarity measurement. We analyse the nature of musical data, compare several novel melody extraction techniques, describe many melody standardisation techniques, develop, and compare various melody similarity measurement techniques, and also develop a method for evaluating the techniques in terms of the quality of answers retrieved, based on approaches developed within the Information Retrieval community.

We have found that a technique that was judged to work well for extracting melodies consists of selecting the highest pitch note that starts at each instant.

We have tested a variety of methods for locating similar pieces of music. The best techniques found thus far are local alignment of intervals and coordinate matching based on n-grams, with n from 5 to 7.

In addition, we have compiled a collection of MIDI files, the representations of automatically extracted melodies of these files, a query set based on the extracted melodies, a collection of manual queries, and inferred and human relevance judgements. Experiments using these sets show that the type of query used in testing a system makes a significant difference in the outcome of an evaluation. Specifically, we found that manual queries performed best with a representation of the music collection that keeps a "melody" extracted from each instrumental part, and short automatically generated queries performed better when matched against a representation of the collection that, for each piece, uses the melody extraction approach described above.

Chapter 1

Introduction

Many people find that they cannot identify a piece of music of which they can only remember a fragment. A solution is to visit a music shop and ask the person behind the counter. Alternatively, a music library allows the use of a directory of musical themes to solve the problem — provided the music was written before 1975 [100]. An approach that is becoming more common is to pose the question in a newsgroup on the internet, including any contextual information such as lyrics and where the piece of music was heard, and may even include an audio file containing a recorded or sung portion of the piece. These people are engaging in content-based music information retrieval (MIR).

People trying to find the name of a piece of music are not the only potential users of MIR technology. For example, composers and songwriters often question where their inspiration has come from, forensic musicologists analyse songs for copyright infringement lawsuits, and musicians are often interested in finding alternative arrangements or performances of a particular piece.

Simple MIR technology is in use in practice. Some prototype MIR systems allow melody queries to be posed and present a collection of likely pieces as answers. Most of these search a monophonic database, that is, music in which only one note is sounded at a time. It is not clear how effective these systems are at finding suitable answers, or whether it is possible to achieve good answers to queries presented to a polyphonic database containing a wide variety of musical styles. Therefore it would be useful to study the effectiveness of MIR techniques.

Issues

Content-based MIR has distinct differences to other information retrieval tasks, leading to the need for a different approach to processing music queries. There are several factors that make the task non-trivial:

- Music is polyphonic, that is, more than one note may be sounded simultaneously.
- A query may be presented in a different key to the stored version, that is, it may start on a different note.
- There are likely to be variations between the query and answers such as repeated notes, ornamentation, and unstressed notes with different pitch. (There are similar variations between different performances and arrangements of the same piece.) Pieces of music are usually perceived as highly similar despite such differences in the notes.
- If the query is presented by singing, there are likely to be errors due to limitations in singing accuracy.
- While a query is likely to be based on the melody of a piece of music, it is not clear what the melody of a piece is: it is as obscure a concept as the "meaning" of a piece of text.
- Exhaustive matching techniques lead to combinatorial explosion.

The choices when developing a prototype MIR system are many. The types of data to work with include audio, polyphonic note data, and monophonic note data. To develop a MIR system that successfully matches a query melody to a polyphonic database of music, several problems must be solved. First, it is necessary to decide what the query should be matched against. Second, an appropriate matching method needs to be developed. Third, it is important to choose an appropriate method of evaluating the effectiveness of techniques developed.

Matching a query melody against polyphonic music is difficult with most sources of polyphonic music data, as the melody component of the music is not defined. This leaves two main approaches: match the query against any possible sequence of notes through each piece of music, or decide which portion of the music the query should be matched against and only match against that. The first approach leads to combinatorial explosion, and is likely to produce many false matches. The second approach involves a "melody extraction" phase, in which the notes that belong to the melody or theme are selected for matching. For some queries, it may be more appropriate to select other portions of the music, such as chordal information. If audio data is used instead of note data, then the problem becomes even more difficult, as it is necessary to extract features from wave-forms. Extracting melody from audio in a way that allows it to be matched against other melodies is currently not possible.

Other Work

Some MIR research has concentrated on monophonic information and others on polyphonic. Most research that uses monophonic musical information has as its source of data a collection of folk songs [42, 93, 99, 116], or a small collection of melodies [71]. A variety of techniques has been used to locate melodic matches, including dynamic programming [93, 97], n-gram-based matching [43], feature histograms [75], and state matching [93].

Some researchers have chosen to match a query melody against melodies automatically extracted from polyphonic musical data [11, 54]. Simple heuristics were used to determine the melody but there was no evaluation of the techniques. Very little has been published on the process of melody extraction. There have been attempts at splitting polyphonic music into parts with moderate success [87], but the melody extraction problem has not previously been explored in the context of retrieval. Researchers who have chosen polyphonic note data for their research [11, 34, 54, 80] use collections of files of data in the MIDI (Musical Instrument Digital Interface) file format as their source. These collections not only provide polyphonic musical data, but include music in a wide variety of styles. Other researchers have chosen to match queries against any occurrences within a collection, regardless of whether these are across musical parts or instruments [23, 34, 80], however, this approach has not yet been shown to produce effective answers, and based on our own (unpublished) experiments, would require the use of more detailed queries to produce good results.

Some researchers have tackled the problem of audio retrieval, in which similarity is determined by extracting features from wave-forms. The state of the art in this area is that it is possible to determine whether a wave-form contains music or speech, some stylistic aspects can be detected in musical recordings, the beats and therefore the tempo can be determined, but identification of actual notes within a non-monophonic piece of music is very difficult. The main problem is that it is hard to distinguish between harmonics and notes. For example if the note A at 220 Hz is played on a piano, the wave-form of that note includes harmonics at double (440 Hz — equivalent to the next highest A note), triple (660 Hz), quadruple (880 Hz) frequency and so on. Note identification is a little easier if constraints are placed on the music to be analysed, such as using an instrument with a simple timbre so that features of the instrument's typical sound pattern can be used to make sense of the wave-form. However, even so the process is not accurate. The focus of research in this field is the identification of new techniques for musical transcription from audio waves [117] and use of non-melodic features for matching, such as structural information [49, 50].

Few MIR techniques have been evaluated for effectiveness using standard information retrieval methodology. Some matching techniques were applied to a small set of pieces and are subjectively evaluated by the researchers [97, 99]. Others have used statistical measures of success [11, 41, 93], or known-item searches [43, 71]. The most comprehensive evaluation technique that has been applied was the testing of a set of 100 hummed queries against a set of 500 songs. The number of pieces that were retrieved in the top 10 and top 1 were reported for four different algorithms [71, 72].

Our Approach

In our work we have applied methodology derived from textual information retrieval to MIR. We have devised a three-stage approach for MIR consisting of melody extraction, melody standardisation, and similarity measurement. Melody extraction involves choosing melody notes out of polyphonic pieces of music to simplify the matching problem and reduce the pool of irrelevant matches. Melody standardisation converts the melodic information into a form consisting of a sequence of symbols. The symbols represent the features used for melodic matching. The similarity measurement stage calculates a similarity score based on the standardised form of the melodies being matched.

Since melody extraction is inaccurate, and since—even with standardisation—the same melody can be represented by different sequences of notes, two strings representing the same melody will not necessarily be identical. Matching must therefore be based on some measure of the similarity of query and piece. Given a standardised query string and a collection of standardised piece strings, matching involves computing a numerical score for each piece with respect to the query. Like other IR systems, the pieces can then be sorted by their score, and the highest-ranked pieces returned to the user as potential matches.

The main assumption we have made is that queries are likely to be based on the "melody" of a piece of music. We predict that the majority of monophonic queries will be of this type. It is quite possible that a query will be on some other part of the music, such as a bass line. However, as we show, techniques that involve matching against the "melody" of each part of a polyphonic piece work well and therefore can satisfy queries on non-melody parts. Our approach can also be used for some forms of polyphonic match, in which case the melody can be automatically extracted from the query before being matched against the collection. A further, more intensive matching process can then be applied to a small pool of best answers.

A second assumption is that the use of performance information, such as that found in MIDI files, will be useful in general for music queries. Most music is not available in this form, but only as recordings or sheet music. It is currently nearly impossible to generate note information from polyphonic music recordings that contain more than one instrument. However, sheet music can be transferred into note information more successfully, and note transcription technology is improving.

Music perception research provides us with some clues as to what would be perceived as melody. Typically, the highest pitch notes would be classed as the melody notes unless they are monotonous. For other notes to be identified as the melody, various compositional tricks need to be applied such as making it much louder than the accompanying notes and making use of a single timbre throughout the melodic phrase. Sometimes the pitch of soloists is slightly higher than normal to accentuate the melody against a large body of sound [61].

Findings from music perception research were a useful guide in deciding what techniques might be effective at MIR. In particular, the research into how melody is perceived has informed our approach to melody extraction. Listeners usually hear the highest pitch notes as melody notes, except when they are monotonous [51]. Pitch proximity is the most important factor in grouping notes into perceived parts [31]. These concepts have been incorporated into our melody extraction techniques and tested with human listeners. The results of our experiment on melody extraction confirm the results from music psychology, in that extracted notes that consisted of the highest pitch at any instant were deemed to be most like the melody of the pieces listened to.

The importance of melodic contour as a feature of both musical memory [36] and singers' accuracy [83] makes it a feature that ideally should be included in any melody standardisation process. We have chosen standardisation methods that retain contour information for our matching experiments.

In the final stage of our approach we applied matching techniques, namely, dynamic programming and n-gram scoring, to melody matching. Different variations on the above two methods were tested to determine retrieval effectiveness.

Our melody matching techniques were tested for effectiveness in two ways. Initially, we generated automatic queries by extracting melodies from pieces that had more than one version within the collection. These were applied using many different variations on matching methods based on dynamic programming and n-gram techniques. The results were evaluated using recall and precision methods.

Our experiments revealed that our three-stage approach can successfully answer melody queries. A technique that selected the highest pitch notes commencing at each instant, which we call "all-mono", was the most effective melody extraction technique as judged by listeners. It was also shown to be effective in melody matching. Melody standardisation techniques tested included contour, directed modulo 12, and exact interval techniques. All three of these use a relative pitch approach, allowing matching in any key. *Contour standardisation* reduces the melody to a string of characters representing "up", "down" and "same" pitch directions. This was shown to be insufficient with our collection for queries of about 10 notes. *Exact interval standardisation* retains exact pitch distance information, which, with the source data used, involves up to 255 unique symbols. *Directed modulo 12 standardisation* represents melodies as a sequence of pitch distances in which large pitch distances are reduced to an equivalent one that is less than an octave in size. This reduces the size of the representation to 25 unique symbols. Our experiments showed that there was little difference in retrieval results for directed modulo 12 and exact interval standardisations for the collection and queries used.

The local alignment dynamic programming technique and the coordinate matching n-gram technique were shown to be the best at retrieving useful answers. Retaining a melody representation of each track or channel of pieces of music was shown to be best for answering manually produced queries.

As our approach is based on note information, we recognise that this does not allow searches on all types of music. First, some music created in this century is not note-based at all, consisting of organised sound instead of melodies. For this type of music, there may not be any melody which could be used as a query. The user may need to use an audio-based technique for successful location of the work — probably by using a sample noise as a query. Second, we do not address music using scales and tunings other than the western scale of twelve notes. This is not a problem for pentatonic music (five note scale) which is easily represented, but microtonal music (music containing intervals that are smaller than a semitone) is not catered for at all. Music intended for different tunings may be represented if the scales contain less than twelve notes per octave, but will not sound as intended and thus may not match with appropriate pieces. However, as contour and relative interval size is important for matching, the disadvantage for differently tuned music may be small.

An important aspect of our work was our experimental methodology. In order to evaluate the effectiveness of different melody matching techniques, we collected query and relevance sets in two ways. First we found pieces in the collection for which there was more than one version, using the filenames and listening. This formed one relevance set. Query sets consisted of melodies that were automatically extracted from the pieces, and truncated to specific lengths. The second type of query set was created by obtaining manual queries via a volunteer who listened to pieces and played a representative melody fragment. Relevance judgements were then collected from users who listened and judged how similar pieces were to each manual query. These query and relevance sets were used to measure the quality of the answers retrieved by the different matching methods. We used two standard IR measures of retrieval effectiveness: eleven-point precision averages, and precision at ten.

The process of collecting relevance judgements revealed some issues that need to be resolved for MIR user interfaces. Some tasks are very difficult and time-consuming for users, such as comparing two unfamiliar pieces of music.

In addition to evaluating the techniques for melody matching, we tested the evaluation methodology itself, by comparing the effect of the two sets of queries and relevance judgements. We found that the two query sets gave significantly different results when used to rank melody matching techniques. In particular, evaluation with the manual set increased the measured effectiveness of techniques that treat each musical part separately during the melody extraction process. The two sets of relevance judgements were more consistent but showed some minor differences.

Organisation of this thesis

In chapter 2, we discuss background information to the field of music information retrieval. Included is a general summary about music and its standard notation, to enable understanding of the examples presented throughout this thesis. We describe the findings in music psychology that are relevant to music information retrieval, in particular, perceived melodic similarity. We discuss the needs of potential users of a music information retrieval system, including those of forensic musicologists and lay people. After presenting related research from the field of musicology, we survey researchers' work in the field of music information retrieval and discuss the methods that they have implemented.

There are several possible approaches to matching similar items. In chapter 3 we present background material on the main techniques that have been proposed for music matching: dynamic programming, n-gram matching techniques, tries, and feature histograms. Our experimental methodology is presented in 4.

In chapter 5, we present techniques for automatically extracting a melody from polyphonic music data and show results of an experiment that evaluates the techniques. This is followed by a detailed description of many potential methods for melody standardisation in chapter 6.

Using musical data processed in the manner discussed in the extraction and standardisation chapters, we present an analysis of the frequency distribution of melody n-grams in chapter 7. This is followed by detailed work on the similarity measurement process for melody matching in chapter 8, with melody matching experiments presented in 9. We present further experiments that make use of manual queries and judgements in chapter 10. We discuss future directions and conclusions in chapter 11.

Chapter 2

Background

Our approach to music information retrieval (MIR) research initially involved surveying related musicology, music psychology, and information retrieval research. Musicologists and music psychologists have explored such useful concepts as measuring music similarity and memory for music. The field of information retrieval provides a methodology for examining the effectiveness of retrieval systems. In this chapter, we cover basic music concepts needed in this thesis, and important findings in music perception and computer musicology. We introduce concepts from the field of information retrieval and survey music information retrieval research.

2.1 Music Terminology

2.1.1 Notes and Pitch

Each time a key is pressed on a piano, a bow is struck across a violin string, or air is blown across a flute mouthpiece to make a sound, we say that a note has been played. Each type of musical instrument has its own method of playing notes within a certain range. This range refers to the frequency range of sounds that can be produced from the musical instrument. Musicians do not usually discuss the frequencies of notes, but refer to them as having a particular *pitch*. For example, the note A to which an orchestra is tuned is defined to have a frequency of 440 Hz. A note with a higher frequency is said to have a higher pitch than one of a lower frequency.

2.1.2 Intervals and Octaves

The pitch distance between notes is called an *interval.* If one note has double the frequency of another, then it is said to be one *octave* above the lower note, that is, the interval between the notes is one octave. Notes that are an octave apart sound very similar to each other, which is reflected in the naming of notes. All notes that are a whole number of octaves apart have the same note name, which is usually a letter, or a letter combined with a sharp (♯) or flat (♭) symbol. For example, the note A with frequency 440 Hz is an octave lower than the note A with frequency 880 Hz.

The notes used in western music result from dividing up an octave into 12 notes that are a semitone apart. The note names used for the notes within one octave starting at C and going up are:

C C♯ D D♯ E F F♯ G G♯ A A♯ B C

C♯ is read as C sharp. It is the same note as D flat (D♭). Similarly, D♯, F♯, G♯ and A♯ can be referred to as E♭, G♭, A♭ and B♭ respectively. Note that a sharp symbol is used to raise the pitch of a note letter by one semitone and a flat is used to lower the pitch of a note by a semitone. The name that is used for a note with two possible names depends on the key of the music.

An interval between notes that is larger than two semitones is often referred to as a "leap". For example, a melody may leap from C to G at one point, which is an interval of seven semitones. In contrast, if the melody has a C followed by a D, this is not a leap but a "step".

2.1.3 Scales and Key

The octave may be divided into 12 notes, but most simple tunes don't use all of them. The term "octave" arose because usually melodies use notes from a particular scale, that is, a list of eight notes within an octave. For example, the C major scale consists of the notes:

C D E F G A B C.

The second C is an octave higher than the first C and is the eighth note in the scale.

Melodies that use these notes exclusively are said to be in the key of C major. Major scales all follow the same interval pattern: tone, tone, semitone, tone, tone, tone, semitone — where a tone is equal to two semitones. In the C major scale the first semitone is between E and F and the second semitone is between B and C. The G major scale consists of the notes:

G A B C D E F♯ G,

and has the first semitone between B and C and the second semitone between F♯ and G. We use the name F♯ because, by convention, a scale should contain each letter-name. Using G♭ would mean that we have two G letter-names and no F letter-name.

Melodies can be "transposed" from one key to another, preserving the intervals between notes of the melody. When this is done, the melody is preserved, but will sound a little higher or lower than the original.

Music that largely consists of notes from a particular key is often called "diatonic". Music that largely ignores keys is called "atonal".

2.1.4 Chords and Harmony

When more than one note is played simultaneously, the result is a chord. Each key has a set of chords that are related to it, usually created by taking every second note of a scale until there are three notes. For example, the C major chord consists of the notes C, E, and G. Harmony refers to the chord sequence used in a piece of music. Usually a piece will end on a chord that is the same as the key of the piece, for example, a melody in C major will end with a C major chord. The music that is played along with the main melody is called the accompaniment. This may consist of chords and possibly percussion, or may be a more complex mixture that includes "counter-melodies". There may be more than one musical instrument involved, in which case we use the term "part" to refer to the music for a particular instrument, as in the "violin part". Music that only consists of a melody with no accompaniment is called monophonic.

2.1.5 Common Music Notation

The Western sheet music tradition is often known as Common Music Notation (CMN). In CMN, music is written on a staff consisting of 5 horizontal lines. Notes that sound at the same time are written one below the other. Notes that occur after another note are written to the right of the note. Notes that are higher in pitch are written at a higher position on the staff. Notes are written on the staff according to a particular key and can be placed between lines or on a line. For example, the scale of C major is shown on the left in Figure 2.1. As can be seen, extra lines can be added for a note that goes beyond the range of the staff. The C major chord could be written as shown on the right in the same figure.

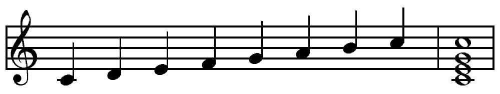


Figure 2.1: *On the left, the C major scale starting on middle C, shown in common music notation. On the right, a C major chord.*

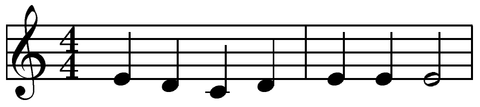


Figure 2.2: *The first phrase of Mary Had a Little Lamb shown in CMN in the key of C major.*

Each part is usually written on a separate staff. The music is normally divided into bars, each containing the same number of beats. The bars are indicated in CMN by a vertical line. The number of beats per bar and the quality of the beats is indicated at the start of the piece by the time signature. The most common time signature is 4/4, which means four "crotchet" or quarter-note beats to the bar. Figure 2.2 shows two bars of Mary Had a Little Lamb with a time signature of 4/4.

If there is to be a period in which no notes are played, one or more "rests" are placed on the staff. These indicate the duration until the next note is to be played.

2.2 Music Perception

Several aspects of music perception are relevant when implementing music database systems. First is the type of query that a user will present. In the case of someone trying to locate a half-remembered fragment of music, it is useful to understand how people remember music and in particular, how they remember melodies. Second, since most music that we hear contains both melody and accompaniment, we should determine what is likely to be perceived as melody in an accompanied musical work. Third, since many music queries will involve finding similar but not exact matches to melodies, we need to decide what similarity means in terms of music perception. We now discuss the research in these areas and the implications for music databases.

2.2.1 Music Memory

There has been much research on how people build mental structures while listening to music and on how music is remembered. Dowling [36] discovered that melody contour is easier to remember than exact melodies. Contour refers to the shape of the melody, that is, whether the next note goes up, goes down, or stays at the same pitch. He postulates that contour and scale are stored separately in memory. The musical scale is learnt through a lifetime of listening to music. As a result, subjects found it easier to distinguish between a diatonic melody and an atonal melody with the same contour, than between a diatonic melody and a copy that stays within the same scale. The easiest task of all was to distinguish between a melody and another that didn't preserve contour. This experiment only tested short-term memory.

Dowling discussed several other experiments that have been used to determine how musical memory operates. These include long-term memory experiments by Attneave and Olsen, which showed that people do distinguish between a well-known melody and inexact tonal copies of it. The ability to distinguish between melodies is discussed further in the section on melody similarity perception below.

Another situation involving music memory and melodies occurs when a melody has not been learned thoroughly. The result is often a version that differs from the original being remembered. There is little in the way of music perception research about this phenomenon, but there is evidence in musicological research [123].

Further interesting results have been found regarding the absolute nature of musical memory. For example, we remember the tempo or speed of a known piece of music and can recall it within 8% of the original speed [81]. There are similar results for pitch, showing that, people can sing their favourite rock songs at the same pitch as the recording without hearing the recording first (discussed by Levitin and Cook [81]).

The ability to recall a melody also depends on the nature of the melody itself. Melodies with "consonant accents", that is, in which the melodic, rhythmic, and stress accents coincide, are easier to remember than those where the accents do not coincide [96].

For rhythm, it has been proposed that, while listening to music, the listener establishes an internal clock and that the rhythm of the music is encoded in memory in terms of this internal clock [103]. Where no internal clock can be established due to the complexity of the rhythm, some other method of encoding must be used.

As with most music perception studies there are significant differences between the results for highly experienced musicians, people with some music experience, and those with no or little music training. Usually memory tasks are performed better by experienced musicians than those with less experience.

2.2.2 Figure and Ground

In order to extract melodies reliably from non-monophonic music files, we need to determine what a person listening to the music would perceive as the melody. Several papers have explored the way we perceive groups of notes.

Francès [51] looked at the *figure and ground* relationship for music. Several factors are considered. A musical part is heard as the figure if it is higher in pitch than the accompanying parts. However, if the upper notes are constant and the lower notes form a more interesting pattern, then the lower notes are heard as the figure. Other factors that can affect a part being perceived as the figure are its loudness and continuity. The way the music is perceived is not always constant, however. A listener can make shifts in their attention between different parts of the music. Francès observed that experienced musicians can more rapidly shift their attention between parts than non-musicians, allowing them to be more aware of the accompaniment in a piece of music.

Deutsch [31] discussed the main principles of perception of groups, originally derived from visual perception, and showed the results for the group perception of music. There are four main principles of group perception: proximity, similarity, good continuation, and common fate. The proximity principle states that we group items, that is, we see them as a unit, if they are close together. In the same way, we group items that are similar in some way, or continue in the same direction, or end together. These have been clearly shown for visual perception and apply to perception of groups of notes as well. There is a definite hierarchy amongst these principles. Proximity of notes is more important than good continuation as illustrated by the "scale illusion" experiment. If one part is descending in a rapid scale and another part ascends so that they overlap as shown in Figure 2.3, listeners perceive the upper notes as one part and the lower notes as a second part (if it is perceived at all). The amplitude or loudness of notes was found to be fairly unimportant in the perception of musical parts compared to proximity, but is also used for grouping of notes. The similarity principle for groups of notes can involve the

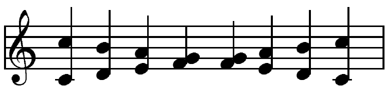


Figure 2.3: *The scale illusion. Two overlapping scales, one going up and the other going down. The listener hears the upper notes as one part and the lower notes as another part.*

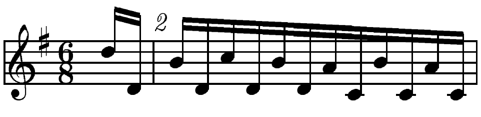


Figure 2.4: *Rapid sequences of notes in more than one frequency range are perceived as separate parts.*

timbre of the notes. Those that have a similar timbre are grouped. The timbre is less important than the proximity of notes, however, as shown by Butler.

If a melody consists of rapid notes where the alternating notes are of a different frequency range, as shown in Figure 2.4, then two musical parts are perceived. This does not occur to the same extent when the melody is slowed down. There is a considerable overlap in terms of how the music will be perceived, so that a melody could be perceived as being two musical parts or just one over a range of speeds.

2.2.3 Melody Similarity Perception

Pitch

Some aspects of melody similarity were discussed above in the section on memory: two melodies that have the same contour are perceived as more similar than those that have a different contour; those that have the same tonality and contour are perceived as more similar than those with the same contour and different tonality (atonal).

Dowling and Fujitani's experiment [38] revealed that it was hard to distinguish between two atonal melodies with the same contour compared to those with different contour. In their second experiment, it was discovered that, for familiar melodies, it is very easy to identify a well-known melody played with exact intervals (pitch distances), somewhat harder to identify one with the same contour and relative interval size, slightly harder again to identify based on contour only, and hardest of all to identify a melody of different contour but the same harmony.

Deutsch (discussed in [37]) showed that changing the octave of notes in a melody makes it hard to recognise. Dowling and Hollombe [36], and Idson and Massaro (in [37]) showed that it is slightly easier if the contour remains the same. That this task is difficult is a surprising result, when one considers that notes of the same octave are usually considered to be harmonically identical.

Contours versus intervals was further explored by Edworthy [47], who found that intervals became more important than contour for distinguishing novel melodies once the melodies exceed a certain length. It is thought that the extra notes help to establish the key to give a frame of reference for the listener.

Van Egmond and Povel [136] discovered that an exact transposition with one note altered will be considered more similar to the original melody if the note is altered "chromatically" than if it is altered diatonically. A chromatically altered note is one that has been changed by a semitone in such a way that the letter-name remains the same. For example, the note F can be chromatically changed to F$\sharp$. Key distance is another factor affecting similarity perception. A key is closely related to another if it differs in only one note. Distant keys are those with few, if any, notes in common. Melodies with the same contour and the same key are perceived as more similar than those in distant keys [9]. However, Van Egmond and Povel [136] found that, when it came to the comparison of exact transpositions, those with a greater difference in pitch — regardless of relatedness of key — were considered less similar than those with a very small difference in pitch.

The music experience of the listener affects how some melodies are perceived. Krumhansl and Shepard [77] found that musical listeners are more likely to prefer notes that are harmonically similar, whereas non-musical listeners will consider notes of a similar pitch to be similar.

Other factors

The research discussed above primarily considered pitch. The experiments that used similar melodies normalised them so that the rhythms were the same. Sundberg [126] discovered that, when listening to singers, notes that were unstressed would be perceived as being the correct pitch whereas if the same note were stressed it would not be considered correct. This seems to imply that unstressed notes should be regarded as less important when examining melody similarity.

An interesting factor in music perception is the difference between what is perceived to what is actually heard. In work on melodic and rhythmic accents, Tekman [128, 129] discovered that melodically accented notes are perceived to be louder than others. A melodically accented note is one that is approached by a leap (interval greater than two semitones). Also, the duration of a note preceding a louder note is generally perceived to be longer. A model of melodic accent has been proposed by Thomassen [130] and was considered by Huron and Royal as the best available model [69].

2.2.4 Implications for MIR Systems

The literature survey of music perception presented earlier suggests that a slightly different hierarchy of similarity holds for melodies in short-term memory to that of long-term memory. In the case of a music retrieval system, the melody queries are likely to be from long-term memory, whereas the set of answers presented to the user will be a mixture of known and unknown pieces. Another aspect that can affect the hierarchy of similarity is the musical experience of the user. For example, when comparing melodies, musicians prefer melodies with similar harmony to those with more similar contour.

We have synthesised the hierarchies shown in Figures 2.5 and 2.6 from our reading of melodic similarity perception. Figure 2.5 shows a hierarchy for melodies based on melodies that the listener knows and modified versions of them, that is, based on long-term memory. Figure 2.6 shows the hierarchy for short-term memory for melodies. The hierarchies rank the perceived similarity of melodies based on specific melodic differences. For example, in the hierarchy for short-term memory, the node representing melodies that have exactly the same intervals and pitch is shown at the top ("identical"), as they are perceived as more similar than melodies that have the same intervals but a different key.

Some of the items in the list cannot really be fully ordered since comparisons have not been made experimentally. As such, a partially ordered set is the best representation of the hierarchy. We have separated the long-term memory similarity hierarchy from the short-term memory hierarchy in our representation, however, there are other factors that affect similarity perception, such as the duration of the melodies [47], and the musical experience of the participants of an

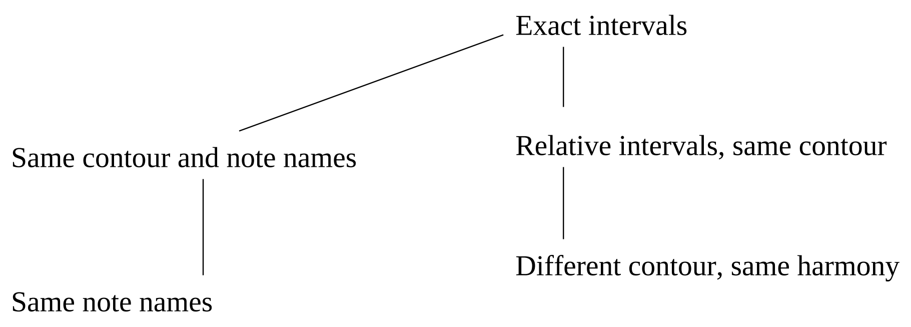

Exact intervals
Same contour and note names
Relative intervals, same contour
Different contour, same harmony
Same note names

Figure 2.5: *Hierarchy of perceived similarity for exact and modified known melodies. The highest similarity is between two melodies containing the same sequence of intervals regardless of key ("exact intervals"). Those types of melodic similarity that are not directly connected have not been compared experimentally.*

experiment [77].

For our purposes the items in the right-most branch of the short-term memory hierarchy can be condensed into one, since relative intervals are likely to be used in melody comparisons.

2.3 Computer Representation of Music

Music can be represented in many ways. It can be stored as: a wave-form representing the sound, an image of the sheet music, performance information, or sheet music layout information. Some formats, such as the draft standard SMDL (Structured Music Description Language), allow for all of these. However, the MIDI (Musical Instrument Digital Interface) standard, a commonly-used format for the exchange of music sequences, only stores performance information. Performance information consists of: when each note is played, how loudly, for how long, and by which instrument.

Other standards have been developed or adopted by musicologists for their needs, such as DARMS, which initially aimed to represent music notation, but came to be widely used for analysis. The standard was developed by Stefan Bauer-Mengelberg in the seventies [68]. A more flexibly system called Humdrum was developed by Huron [67]. This is more of a music protocol than a single representation and allows musicologists to represent the aspects of music in which they are interested.

For music type-setting, there are three related standards for use with the type-setting lan-

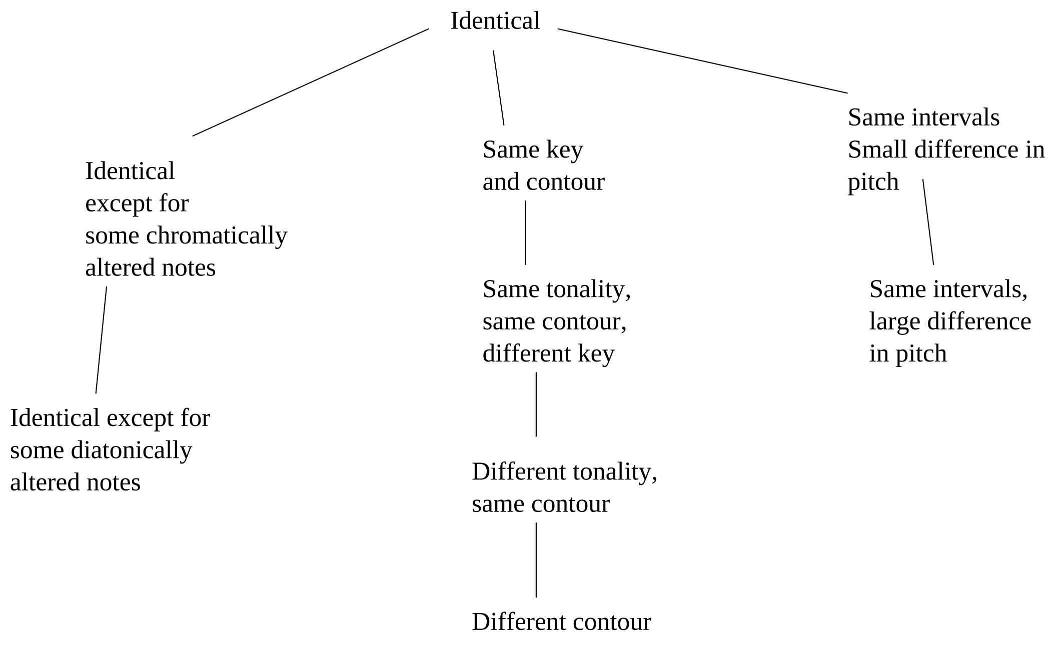

Identical
Identical except for some chromatically altered notes
Identical except for some diatonically altered notes
Same key and contour
Same tonality, same contour, different key
Different tonality, same contour
Different contour
Same intervals Small difference in pitch
Same intervals, large difference in pitch

Figure 2.6: *Hierarchy of perceived melodic similarity for unfamiliar melody fragments, based on pitch only. The items in the right-most branch can probably be placed between the "identical" and the "same key and contour" nodes, but to our knowledge there have been no experiments to confirm this.*

guage LaTeX: mutex, musictex and musixtex. These allow the type-setter considerable control over the layout of sheet music but are not intuitive for direct use by users [57]. Among file-formats for more user-friendly systems for music type-setting, the NIFF (Notation Interchange File Format) [58] standard was proposed and developed. Despite the development being sponsored by several music software companies, it does not yet have wide support.

There have been newer representations published, such as ZIPI [90], which aims to solve MIDI's shortcomings. Many extensions have also been proposed for MIDI. More recently, XML document definitions have been developed (for example [56]). Some of these, and other representations of music, are discussed in detail in "Beyond MIDI", edited by Selfridge-Field [119].

2.3.1 MIDI

The MIDI standard was initially developed for communication between musical instruments. The standard describes the hardware, speed of data transmission, and the messages that can be sent. Messages are sent whenever a MIDI event occurs.

A MIDI event represents such things as pressing or releasing a key on a music keyboard. For example, the "Note On" MIDI event occurs when a key is pressed. It consists of a channel number, representing which instrument played the note, a note number representing which key was pressed, and a velocity or loudness of the note. The note will sound until a corresponding "Note Off" event occurs. Note numbers range from 0 to 127. The note number represents the pitch of the note. For example, note 36 represents C and note 37 represents C$\sharp$.

Other events that result in MIDI messages being sent occur when controls are changed on a keyboard. For example the wheel that controls the pitch sends a pitch-bend event when the wheel is moved.

Standard MIDI File Format

The MIDI file format standard is used to represent the performance of music. It can only be used to store and play instrumental music and does not store notation information such as the division of music into bars or measures. There are facilities within the standard for storing time and key signatures in addition to other meta-information, such as title, copyright, and instrument names. The inclusion of these elements is not guaranteed however.

Standard MIDI files contain one or more tracks of MIDI events. Each event is preceded with an integer representing the time that has elapsed since the last MIDI event.

Most MIDI files contain music that is to be played on more than one instrument. Usually, each instrument has a separate track. When these files are played on a typical PC containing a soundcard, appropriate sounds are chosen to play each instrument's tracks. An extension to the MIDI standard is known as General MIDI, which specifies standard instrument sounds that are represented by specific codes that can be used in MIDI files. General MIDI drum-kits all have the same mapping of note numbers to sounds. For example, the Bass drum is associated with note 36 (C1) and the Acoustic Snare Drum with note 38 (D1).

There are two MIDI file format standards: format 0 and format 1. In format 0, all the performance information is held in one track (track 1). Each individual instrument has its own channel and events for the instrument can be identified by the channel number included in the event structure. In this case, all events are stored in chronological order. In format 1, MIDI files the instruments still have their own unique channel number, but the events for each instrument

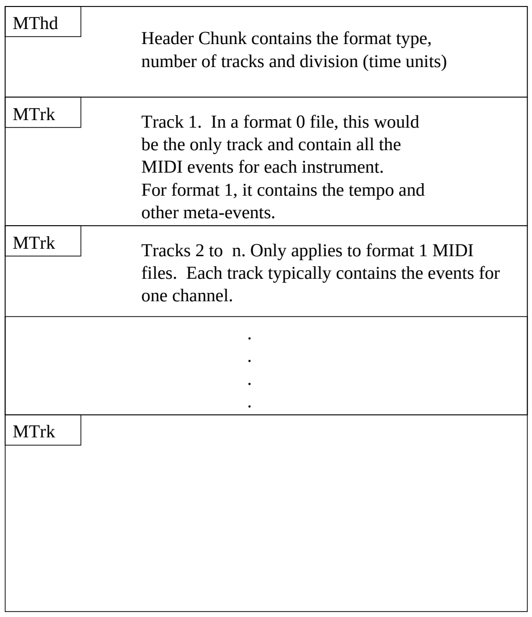

MThd
Header Chunk contains the format type, number of tracks and division (time units)
MTrk
Track 1. In a format 0 file, this would be the only track and contain all the MIDI events for each instrument. For format 1, it contains the tempo and other meta-events.
MTrk
Tracks 2 to n. Only applies to format 1 MIDI files. Each track typically contains the events for one channel.
MTrk

Figure 2.7: *The structure of a standard MIDI file*

are stored in separate tracks. In this case, the first track is usually reserved for meta-information, such as timing and instrument information. (See Figure 2.7).

The Internet has made large quantities of MIDI data available to users. One site archived MIDI files sent to a newsgroup and contained over 15 000 files. The files are largely created by fans, musicians, and employees of software companies that provide demonstration MIDI files with their product. MIDI data from the Internet is used as a basis for the experiments in our research.

2.4 Computer Musicology

Originally, musicologists used manual techniques to compare different musical works. New mathematical or computerised tools are occasionally discussed in the literature. Examples include numerical methods of comparing musical styles ([26]), using entropy as a measure of style ([73]), cluster analysis of melodies ([85]), and music chronology by seriation ([60]). The focus of these works is the application of new techniques and discovering what they reveal about the music.

Some musicologists have gone beyond mere analysis and have simulated different composers' style. For example, David Cope [25] applied pattern-matching techniques to locate a composer's "signatures", that is, patterns that occur in more than one piece. Using these segments, new compositions were created in the style of the composer using software.

Other work in the musicology realm has been in the form of collecting musical works of a particular genre and creating an index for these. Examples include the French Chanson catalogue compiled by Hudson [66], and Stinson's database of fourteenth century music ([125]). Computer programs have been developed to generate new compositions and split a piece of music into its component parts. There is also much work published about the digital representation of music.

2.4.1 Musical Style and Similarity of Music

Musicologists often perform analyses of musical works and with the advent of computer processing power, it became possible to perform larger comparative analyses of works. Several different approaches have been used to measure similarity for music. Indeed, there are different types of similarity in music. We could be measuring similarity of style, harmony, structure, melodies, rhythm, or instrumentation. In the book "The computer and music" [82] published in the mainframe era of computer use, various articles discuss the comparison of musical styles using numerical methods. Crane and Fiehler [26] discussed different metrics that can be used to compare pieces of music. A measurement may consist of a binary value, several possible discrete values, or a value in a continuous range. These various values can be combined into a single number representing the similarity of the musical works in many ways. One approach is the mean of the differences. Another is to consider each measurement to be a dimension in Euclidean space. Each musical work would be represented as a point in space and their similarity would be measured by the distance between the two points. How the measurements are applied or scaled is determined by a certain degree of trial and error. Crane and Fiehler scaled all measurements to fall between 0 and 1. They applied several approaches to calculating a single value for each of twenty French Chansons. All gave similar results. Songs by the same composer were usually grouped together but not always. In principle works can be grouped into clusters based on their relative distances.

Seriation is a mathematical technique that can be applied to a collection of works that are assumed to have a sequence in time, for example showing a composer's change in style over time or to estimate the order in which various pieces are likely to have been composed. The technique can be implemented using a dynamic programming approach, as developed by Hubert and Arabie (discussed in [60]) David Halperin [60] applied seriation to determine a sequence for a collection of songs written by the troubadours. He used such features as average pitch, average size of rising and falling intervals, ratio of rising and falling intervals, and the number of lines per song. He found that the seriation technique gave good agreement with historical evidence, with only two of the thirteen troubadours being swapped from their chronological order.

James Gabura [53] discussed various methods that have been used to measure the style of musical works, including the entropy in bits per second, number of key changes, and the distribution of interval sizes in melodies. He applied numerous measures and successfully trained a neural network to identify the composer of various works.

Logrippo and Stepien [85] used cluster analysis and factor analysis on music to analyse melodies. They clustered melodies based on the number of occurrences of notes or intervals. A further approach attempted was to determine the longest sequence of common notes in two melodies (referred to as "longest common subsequence" in our work). The length of this is expressed as a proportion of the length of the shortest melody. Any number of notes can occur between the common notes. They commented that a secondary ranking based on the length of the sequence may also be required. The resulting distances were used to develop a structure showing how the melodies are clustered. They found their techniques revealed some useful relationships in the musicological analysis of songs from Eskimo Point, Rankin Inlet and Thule.

Stech [124] produced a program to locate related melodic patterns within a piece of music. The patterns searched for were tonal answers, exact melodic patterns, rhythm matches, and contour matches. The program also located inverted melodies and retrograde (backwards) melodic patterns. The music was encoded as a sequence of pitch numbers, optionally qualified by symbols representing rhythm. Rests were also encoded. The program successfully located patterns in Dunstable's Veni Sancte Spiritus motet.

Mongeau and Sankoff [97] calculated an edit distance between melodies, including measurement of insertions and deletions, and weighting of importance of different features. They made use of both melody and rhythm in their comparisons. They tested their approach on Mozart's set of nine variations on the theme "Ah! Vous dirai-je, maman" (K. 300). Local alignment was also implemented and local similarity within Mozart's Alleluia (K. 165) was tested using this method.

In summary, these musicologists chose features of interest to them and compared different pieces of music, in order to group similar pieces, to observe the evolution of musical style, or to quantify stylistic elements of composers and their works.

2.4.2 Part Splitting and Melody Extraction

We have been unable to find much published work on melody extraction algorithm. There appears to have been no formal evaluation of different algorithms other than in our own work [132]. Blackburn et al. selected the lowest pitch notes in each track for their "navigation by musical content" system [11]. Ghias et al [54] mention that they used various extraction heuristics, one of which consisted of excluding the percussion channel of the MIDI files, which is normally channel 10. How effective these are is not clear. In recent work, Francu and Nevill-Manning [52] selected the "highest energy" note, where the energy was calculated by using a combination of the amplitude and frequency.

In work that followed on from ours, Blackburn and de Roure [12] built a part classifier, that used a set of features including average pitch and entropy to classify MIDI channels into bass, rhythm, accompaniment and lead. The techniques were evaluated by selecting melody fragments from lead parts and averaging the rank of the pieces that these fragments came from. Using the classifier to reduce the number of pieces to match against improved the rank of these pieces.

Despite the dearth of melody-extraction work, there has been some serious investigation of part splitting, that is, given the musical data of a piece of music, splitting it into likely separate monophonic parts. Marsden used perception principles, such as proximity, to split polyphonic music into its parts [87]. He concluded that it would be impossible to do so reliably with an algorithm. Similar rule-based expert-system techniques were applied to the task of transcribing lute tablature by Charnassé and Stepien [17].

2.5 Computer Musicology versus Music Perception

Theories about music and harmony have existed for hundreds of years. For example, Rameau published his "Treatise on Harmony" in 1722. About twenty years before this, acoustics pioneer Joseph Sauveur published experimental evidence of overtones [104]. Rameau's theories were initially based on a mathematical approach and only later were the physical properties of sound understood with their relation to music.

Music perception research has validated some music concepts, for example the perceived similarity of exact transpositions. There are instances, however, where music perception and a strictly musicological approach gives rise to different results. For example, as pointed out by Butler (and discussed by Deutsch [31]), Tchaikovsky's sixth symphony has the theme and accompaniment distributed between two violin parts. The theme is perceived to come from one set of instruments and the accompaniment from another, however.

This has implications for algorithms that try to extract a melody or convert a collection of notes into musical parts based on music perception. For example, the result of the melody extraction process may not be as originally organised by a composer, even if it correctly simulates how it will be perceived. Conversely, if the stored music is based on the sheet music for a composition, then the musical work may not be retrieved given the user's perception of the melody.

2.6 Information Retrieval

Information Retrieval (IR) is a well established field of research that has been in existence for longer than the computers that are an important tool for IR today. The field concerns itself with the techniques that best allow a user's information need to be satisfied [113]. When a query representing the user's information need is presented, the answer to the user's question should be produced by the information system. In today's text retrieval systems, a ranked list of documents is presented that, it is hoped, best answers the user's need. The information retrieval field has various established methods for determining the success of a retrieval technique. These rely on the concept of relevance. This is discussed in more detail below.

2.6.1 Relevance

In the field of textual IR, a user has an information need that may be expressed as a query written in English, or as a set of key words. Typically this query is presented to the information retrieval system, and a list of documents judged to be relevant to the query is displayed to the user.

IR scientists need to evaluate how well their retrieval techniques perform in terms of producing useful answers to queries. The main approach used is to determine the proportion of relevant answers. For this to succeed it is important to define what is meant by relevant. The standard approach is to use human judges to look at each potential answer and make a judgement as to whether it is relevant or not.

For example, suppose a user wants to find out whether Myasthenia Gravis is related to thyroid problems. The query could be presented to a retrieval system as a list of words and a list of documents retrieved by the system would then be presented to the user. Human judges would look at each of these documents in turn to decide whether they are relevant. A document that discusses Myasthenia Gravis would probably be classed as relevant: one that discusses thyroid problems in general but doesn't refer to Myasthenia Gravis at all may or may not be judged as irrelevant by an assessor. An article on Gravis Ultrasound soundcards would definitely be classed as irrelevant.

2.6.2 Experimental Methodology in Information Retrieval

The usual approach to testing a new retrieval algorithm is to make use of a collection of data, queries, and relevance judgements. Queries are run against the data collection using the algorithm and the results analysed by comparing results to the relevance assessments for the queries to determine retrieval success. This approach is often referred to as the *Cranfield* model, after its first use [24]. For text retrieval, standard collections have been put together to allow different research groups to test their techniques on common data, allowing comparison between results [70].

Once a set of queries have been applied to a collection, the answers can be analysed to determine how successful the retrieval technique is. Typical ways of measuring retrieval success are recall and precision [24]. Recall is measured as the total number of relevant documents retrieved divided by the total number of relevant documents. Precision is the number of relevant documents retrieved divided by the total number of documents that were retrieved. There are other measures that are also applied to test retrieval success. For example, eleven-point precision average takes the average of the precision at each decile of recall from zero to one hundred percent. Retrieval techniques can then be compared to others using the same measures of retrieval success.

To confirm that the differences in retrieval effectiveness of different techniques are statistically significant a test such as the Wilcoxon test can be applied.

2.6.3 IR Models

In developing methods of measuring similarity between queries and documents, several methods have been devised, each based on a different view of the retrieval problem. Some of the more well-known ones are the vector-space model and the probabilistic model.

The vector space model treats each unique term in documents as a dimension, and each document is described as a vector consisting of the weight, such as the term frequency, of each term. For example, for a set of terms { a, cat, dog, mat, on, sat, scat, the }, the document "the cat sat on the mat", could be represented by the vector [0,1,0,1,1,1,0,2]. The similarity between documents and queries can be determined by calculating the distance between their vectors using vector space geometry [114].

A probabilistic model has also been applied to document similarity. In this approach, a similarity formula is developed based on probability theory. Since probabilistic formulae can be computationally complex for the purpose, a simplified formula is often used that makes use of term frequencies and document frequencies [106].

2.7 Music Databases

In our discussion of music databases, we distinguish between music repositories, music database systems, and music retrieval algorithms. When unqualified, the term "music database" refers to a music collection, that is, a collection of pieces of music that has been electronically stored. We use the term "music retrieval system" to refer to a system that provides retrieval capabilities for a collection of music. The term "music retrieval algorithm" is used for the techniques employed for retrieving answers to music queries.

2.7.1 Existing Music Databases and Retrieval Systems

There are several different kinds of music database available today. Musicologists have compiled collections of music of specific genre in digital formats since at least the early seventies (for example see Lincoln [82]). Various different methods of encoding have been used for these collections. Some have used the formats discussed in the section on computer representation of music above. Others have developed representations suited to the needs associated with users of the collection. We are primarily interested in those that store musical content or databases that provide a method for searching by musical content. RISM is a catalogue that does the latter of these two. It contains information about music manuscripts and other musical works held in music libraries throughout the world. The catalogue does not contain musical content itself but does provide an “incipit” or initial melodic phrase of musical works. This allows limited content-based searching.

There are two main paper-based MIR systems available. The first is Barlow and Morganstern's Dictionary of Musical Themes [8]. In this dictionary, themes are represented in the key of C major or minor as a sequence of note names. No note durations or indications of octave are used. Despite this, only six notes are required to locate most themes, and a query length of eleven is the longest needed. Users of this dictionary are required to be sufficiently musically literate that they can transcribe music they have heard and also present it in a different key. A later book was produced by Parsons that made the task of searching for themes much easier for the lay person [100]. This book presented all themes as a sequence of up (U), down (D), and remain the same (R) characters, that is, a melody contour representation. The dictionary includes all the themes from Barlow and Morganstern, plus a large collection of popular tunes, making the total number of themes about 10,000. The author states that queries need to be sixteen notes long to distinguish all themes, but only nine are required for the popular themes. Both books exclude ornamentation in their representation of themes.

Several systems are available on the Internet for processing melodic queries, each with its own unique collection. Stinson [125] provides the ability to enter text or encoded melodic queries to locate musical works of the 14th century. The New Zealand Digital Library project provides a melodic search facility for Schaffrath's collection of folk songs [5]. Users can record a sung query and submit this as an audio file to the system. The site also provides a limited search facility for a large collection of MIDI files gathered from the Internet. Blackburn and de Roure's system [11, 108] allows users to enter a contour query to search a collection of MIDI files. The Carnegie-Mellon University “By Content Music Indexing Project” has a user interface that allows the user to move notes up and down to create a contour query of eight notes [10]. These notes are compared to the first eight notes of melodies in the collection.

Kornstadt developed a web interface for a database developed by Huron using his Humdrum language [74]. The user can enter melody queries in a variety of encodings and can restrict the set of answers with some metadata criteria. The collection consists of over 2000 monophonic themes.

A stand-alone system for “query-by-humming” was developed by Borchers et al. [14, 15], as part of an interactive music exhibit.

Technologies used for retrieving data from music repositories include standard text retrieval technology, as used by Schaffrath [116], standard semi-relational databases [45], and custom systems. We discuss in more detail the techniques that have been studied in section 2.8.

2.7.2 Music Database User Issues

There are several user-related issues in retrieval of music via a melody fragment query. In order to satisfy the user’s needs, it is necessary to define who the user is, what they want to know, and what type of results they will consider to be relevant. Once these issues are resolved, an appropriate means of establishing relevance can be defined as well as a method of evaluating the retrieval system. This section discusses the different types of users and the queries that they wish to have answered, the meaning of relevance for these users, and proposed methods of evaluating a retrieval system given these user needs.

There are several types of user for a music retrieval system: musicologists analysing collections of music, forensic musicologists who are called upon by music copyright lawyers, composers or songwriters, Internet users wishing to retrieve a specific music file, customers wishing to purchase recordings or sheet music, music librarians or music retail assistants locating music for patrons or customers, and people that are just curious about where a particular musical phrase that they remember comes from. Each of these types of users has slightly different requirements. Their queries and prior knowledge differ, as will the results that they consider relevant to their queries. The different users and their needs are discussed below. Issues related to the accuracy of their queries are also discussed.

Customers and Library patrons

Music retail customers, library patrons, and those that may assist them are usually looking for a specific piece of music of which they can recall only a fraction. The same is true for those that are merely curious to recall a piece of music that they only partly remember.

The fraction may consist of part of the sung component of a song, a fragment of the accompaniment, or may even consist of a musical part that transfers from one musical instrument to another. The user may be unable to remember more of the musical work, making an iterative querying process unlikely.

To these users, a query result will only be relevant if it is the same as the musical work that they remember. The other results may have some similarity that might lead them to be considered relevant in a general sense, but they will not be of value to the user. This difference between "topicality" and "utility" is discussed by Blair [13].

Internet users

Web users who are after a specific music file (possibly MIDI) may be able to recall significant portions of the music that they are after. The nature of MIDI files on the web is that many are archived with abbreviated song names that can be difficult to decipher. A song may not be able to be located via its name. Many files may have the title contained within the MIDI file but this is also no guarantee of success in locating a song, since the MIDI file format does not make the storage of song titles compulsory.

A Web user is likely to refine their query depending on the number of results retrieved. They may be after a specific musical work, which may, however, occur as more than one MIDI file. These files can be quite different in terms of key, arrangement and timing. There may be more than one relevant answer in this case. In terms of file location, there are likely to be duplicates of a file occurring at different locations.

Composers

Composers or songwriters may be concerned about whether their work infringes the copyright of existing works, or may merely wish to confirm the source of their inspiration. The nature of the process of composition makes it difficult to distinguish between an inspired original musical idea and a musical theme that was once heard and half remembered. It is quite easy to unconsciously re-write an existing work. If a composer's new work has a recognisable similar portion to an existing work and the composer is likely to have heard the similar prior work, then copyright infringement may have occurred.

A composer's query may consist of a fragment of melody that she wishes to verify is not in infringement. It may also consist of a complete arranged piece of music. Queries may fall somewhere in between these two extremes.

The results of a composer's query will be relevant if the melodies or arrangements are similar in terms of copyright. Copyright similarity may be different from similarity perceived by humans. Results would need to be ranked according to a hierarchy of similarity criteria. These would include the number of contiguous notes that are similar in terms of melody contour, precise intervals, rhythm contour, precise rhythm, and harmony similarity.

Composers may also be interested in different types of queries that could aid the composition process. If a melody is being harmonised or arranged and the composer would like to see how other composers handled a particularly difficult sequence of notes, a query could be used to retrieve this information. For long musical works, the location within the work would be useful to allow the user to locate the relevant part of the work. The composer may wish to add other constraints to the query such as the period or style of the work to ensure relevance.

Forensic Musicologists and Music Copyright Lawyers

Music copyright lawyers in the act of defending or carrying out litigation on behalf of composers would have similar queries to the copyright queries described above.

These lawyers call upon a forensic musicologist to present evidence regarding the similarity of two pieces of music and the possible existence of prior art. The musicologist analyses the two pieces and makes use of whatever tools are available, such as theme and incipit (see Subsection 2.7.1) indexes, to locate prior art. For most early court cases the analyses have been quite ad hoc [30], however, the methods of forensic musicology have become more formal in recent years [105]. There are two classes of results that are of interest for copyright queries: works that are copyright and those that are in the public domain. Depending on whether a case is being made to defend or prosecute will determine the query results in which the user is interested.

Musicologists

Musicologists are interested in a wide variety of aspects of a collection of music. The kinds of tasks carried out include the analysis of music, and the comparison of different pieces, composers, or music from different eras or countries. Examples of the kinds of analysis that can be performed with a set of music stored electronically and a set of tools include determining the frequency of particular musical motifs in composers' works, the shapes of melodies used, and the use of particular musical intervals in styles of music. Musicologists may wish to locate previously undiscovered features that occur in composers' musical styles. This may be used to confirm authorship of works (for example, Halperin's use of seriation [60]). Sometimes musicologists build special-purpose databases, which may in turn be used for further analysis (for example, Stinson's collection of fourteenth century music [125]).

2.7.3 Query Accuracy

Different types of users have different levels of skills when it comes to describing their query. Composers, musicologists and other musicians could be expected to prepare a query using a keyboard or notation with a good degree of accuracy. For queries comparing an entire piece of music, a MIDI file or other standard means of describing music may be used as the query. Music retail staff often have music skills and would also be able to describe queries to the system adequately.

The lay person may have some difficulty in preparing a music query. Several systems have been described that allow the user to create queries by humming or singing. Problems that occur when using this method is that people do not sing very accurately, especially if they are inexperienced singers or unaccompanied. Even skilled musicians have difficulty in retaining pitch for the duration of a song when singing without accompaniment. Programs that process this audio input need to perform pitch tracking to decide what pitch the user really meant rather than what was actually sung. Melody contour, however, is usually accurate.

This leads to the issue of user error. Some users will be very precise in their queries. Others may have errors due to the method used to enter their query. There may need to be some means of addressing the issue of helping the user prepare an accurate query to aid the retrieval process. In a large database there may be too many irrelevant pieces returned if a percentage of user error in the query has been allowed for. It may be best to give this control to the user as a series of options.

2.7.4 Music Databases in the Future

The amount of music available in digital formats will continue to increase, and automated methods will be needed for preparing data for searching by content. The majority of that music, however, will only be available in audio formats such as that found on audio CDs and MPEG files on the internet. In the future we will need to be able to convert an audio recording into discrete note data for a content-based music retrieval system to be practical for a representative collection of the world's music. This problem is yet to be solved, the difficulty arising out of the complexity of typical audio recordings. These are usually multi-timbral as well as having multiple simultaneous notes.

2.8 Survey of Music Information Retrieval Research

Several researchers have explored the problem of searching music databases for melodies. Other research exists on calculating the similarity of musical works.

Dillon and Hunter [33] discussed the location of melodic variants of pieces of music. They referred to research by Bayard, who concluded that variants of Anglo-American folk music differ more at the starts and ends of phrases than they do in the middle. The pitch of melodies were more stable across variants than rhythm. Stressed pitches were considered more important for matching melodies. Syncopated notes — notes that seem to be stressed despite being on a weak beat because they have a long duration — were not treated as stressed notes. The issue of identifying melodies that only differ in their mode, that is, start on different notes of the same scale, was also raised. Dillon and Hunter developed a set of melody representations that were based on scale numbers. For example, in one of their representations the first phrase of Mary Had a Little Lamb would be encoded as:

```
3 2 1 2 3 3 3
```

Another representation included bar numbers, by preceding each bar with a bar number enclosed in hyphens:

```
-1- 3 2 1 2 -2- 3 3 3
```

Other variations included, the representation of only the stressed pitches in the same format, and separating phrases with a "/" character. It was proposed that the melodies in the database be represented in each of these formats. Queries would then be presented in the same manner allowing exact matches to be used for searching. While it was recognised that evaluation using recall and precision measurements should occur, the lack of a body of music in the format required and lack of agreement were cited as impediments to such an evaluation.

Hawley [62] performed exact matches using a binary search on melodies that were stored as a series of relative pitches in a text format.

Some have tried to apply traditional database modelling techniques to musical data storage. Rubenstein [111] applied ER-modelling to the storage of musical notation information. Eaglestone [45] proposed extending the relational model for the storage of music. He made use of temporal database techniques in his approach. The approach allowed the maintenance of historical information about data updates. It allowed queries on note events using its extension of SQL, but was not really proposed as a MIR system in the sense of answer melody queries to find matching pieces.

Chou et al. [21] used Pat trees to index melodies. The notes of each measure of the melody were combined to determine the "chord" and this chord sequence was used as an index term. This allowed for some kinds of user input errors, and may also be useful for locating pieces with similar harmonic sequences. The music was stored in a diatonic fashion, not allowing for notes that are outside the key of the piece of music. The approach is unlikely to scale well due to the difficulties relating to the key changes in general collections of music, and the coarseness of the index terms. There was no evaluation of the effectiveness of the technique at finding answers.

B. Chen [19] built a query-by-humming system that used absolute pitch representation of pieces, and used a Euclidean distance to calculate similarity between query and pieces in the collection. There was no evaluation of the techniques used.

Hsu et al. [65] used a correlative matrix in their approach, that is, a dynamic programming algorithm to match a melody string against itself to locate all substrings that are repeated. The symbols used to represent the melody consisted of absolute pitch, however, the authors generalise the technique to other representations. The technique was tested for speed in relation to the number of notes and patterns found. In related work Liu et al. [84] applied an "RP-tree" to the task of finding non-trivial repeating patterns in a piece of music. RP-trees compress the music string into a tree. Experiments showed that the approach was faster than the use of a suffix tree or dynamic programming.

J. C. C. Chen and A. L. P. Chen [20] developed an index on rhythm for their system. The structure used was an "L-tree", which is a kind of trie and k-similar matches were used as a basis for matching. They used the rhythm of each bar as a symbol in a rhythm string to be indexed in the tree structure. Experiments examined tree-size, response time and the average number of songs containing rhythm substrings of different lengths. Rhythm strings greater than four bars in length were generally unique in a collection of 102 folk songs, and a single bar of rhythm was found in about four songs on average.

In later work by A. L. P. Chen et al. [18], sequences of pitch and duration information were indexed. Each note or sequence of notes of the same pitch was represented by a segment type, a duration in number of beats, and an interval size. The segment type represented the local contour of the melody. A variation on the suffix tree was used to index the information and found to be more efficient than a normal suffix tree for this task.

Kageyama et al. [71, 72] created indexes of their melodies, indexing from the start of each phrase, reasoning that users will usually start a query at the start of a phrase. This reduces the number of terms required to index each melody at the expense of supporting queries that start in the middle of phrases. Inexact matches were permitted. They experimented with four matching techniques and evaluated these by searching for 100 hummed queries in a database of 500 songs. The measure of retrieval success used was the percentage of queries that had the piece that the hummed melody was based on in the top 1 or top 10. The technique that worked the best under this evaluation scheme was a dynamic programming technique that used a weighted combination of pitch and duration, and allowed semitone errors.

Kageyama et al. also refer to work published in 1988 by Yamamoto, who developed a system allowing melody queries that performed exact matching only.

Downie et al. [39, 40, 41, 42, 43, 127] applied an informetric approach to MIR, that is, they analysed the statistical properties of a collection of folk-songs in order to predict the most appropriate melody matching strategy (See also, section 7.1). They listed several possible methods of representing melodies, which were analysed for term distribution, The SMART text retrieval system was used for their experiments. Based on their analysis, they concluded that the contour representation of melodies was insufficient for melody retrieval.

Ghias et al. [54] used a collection of MIDI files as data. Melodies were extracted from these files as contour strings, discussed below. These were stored in a text file and searched for matches to users' hummed queries. For a collection of 183 songs, queries of 10 to 12 notes were required to retrieve 10 percent of the collection. Ghias et al. stated that using a more fine-grained melody representation than contour, such as an alphabet of five possible pitch transitions, would be promising.

Blackburn and de Roure's approach consisted of building a system for hypermedia navigation based on melody contour [11]. In later work, they used secondary contours to improve the accuracy of searches [108]. The contours were split into n-grams of lengths twelve to fourteen. The researchers believed that the frequency of n-gram occurrences within a MIDI file was significant for matching, particularly for inexact matching. The search structure used was a tertiary tree with a branch for each possible contour symbol, that is, it was a suffix tree. Evaluation of their work consisted of testing the uniqueness of contour n-grams in the collection.

McNab et al. [91, 93, 92] converted melodies to contour strings, but also explored the effect of storing exact interval and contour and exact rhythm information. He discovered that after exact interval and rhythm, exact contour and rhythm was the combination with the greatest discriminatory power. He used a modified version of the edit distance algorithm to allow for two dimensions: pitch and duration. The techniques have been implemented as part of the New Zealand Digital Library project [4, 5].

Crawford et al. [27] surveyed possible string-matching techniques that could be applied to music matching. They list twelve types of melody matching problems, including exact matching, chord recognition and approximate matching. For each problem they list techniques that could be applied to the problem, for example, the Boyer-Moore algorithm can be applied to the exact matching problem.

Crochemore et al. [29] presented an algorithm to solve a rather unusual matching task, to locate evolutionary sequences in music or molecules. This algorithm finds sequences of substrings within a string that have an edit distance of (say) one between adjacent pairs of strings.

Salosaari and Järvelin [112] explored the possibility of using n-grams of lengths two to four for melody matching. The similarity measure used was the number of n-grams in common between the query and document divided by the number of n-grams in the query. They tested their approach on a set of themes from a Bach fugue. They classed nine of the seventeen themes as relevant and the rest as irrelevant. They concluded that the longer n-grams gave better results but missed some answers. They doubted whether n-grams would be effective in large-scale music databases.

Ó Maidín [99] calculated the pitch difference between two melodies by aligning them in time. For this approach to work, both melodies must be in the same key. A statistical theorem allowed the most appropriate transposition one of the pieces to be matched to determine the smallest possible pitch difference between melodies. Ó Maidín also applied weights so that longer notes had more influence on the matching process. He tested the technique on a set of 419 Irish tunes and reported that it worked well on this set at identifying matches. The matching method required melody fragments to be the same length. We believe that the technique could be extended to pieces of different lengths by using a windowing technique.

Pollastri [102] implemented a pitch extraction algorithm for extracting the melody from a monophonic audio source. In addition he applied dynamic programming to the task of matching melodies extracted from audio. To test his approach he used a set of audio files that were versions of a melody that included or excluded ornamentation, expression, or errors. He concluded that the matching approach worked but was uncertain how well it would succeed with shorter melody strings and large collections.

Melucci and Orio [94, 95] implemented a phrase-based MIR system that used several different kinds of melody "normalisation", including pitch normalisation, so that phrases start on the same note, and duration normalisation that makes use of the greatest common divisor of the durations. Evaluation of their approach was carried out using a collection of 419 pieces of classical music, and a set of automatically generated incipits of these pieces of varying lengths. They found that the answers to a query usually contained a significant number of pieces by the same composer.

Kosugi et al. [75, 76] developed a MIR system that uses a feature vector consisting of tone distribution and tone transition features. Tone distribution consists of the amount of time spent at each pitch relative to some base pitch. Tone transitions were defined as musical intervals relative to positions within a bar. The researchers found that the combination of the two features gave better results than just using one feature. The tone transition feature was almost as successful as the combined features. The measure of retrieval effectiveness used was the rank of known-item searches for a set of 186 hummed queries in a collection of 10,069 melodies.

Tseng [131] tested a music index consisting of n-grams of lengths two and three. Both relative and absolute pitch representations were used. It was discovered that 3-grams were better than 2-grams for retrieval.

Lemström et al. initially used fuzzy intervals and tries for music matching [78], but later changed to matching against polyphonic musical data using bit-parallel techniques [80]. This is discussed further in Chapter 3. This team also experimented with combining pitch and duration into a single value for matching [79].

Rolland et al. [107] are developing a system that takes hummed queries as input and uses a similarity measurement method that takes into account structural properties of the music and different types of error. Like Mongeau and Sankoff, they have applied a different weight depending on the specific note substitution made. To our knowledge there was no evaluation of the similarity measurement technique in terms of retrieval effectiveness.

Dovey [35, 34] presented the theory behind a brute-force approach to locating matches of melody fragments in a collection of music. As the technique involved absolute pitch, queries needed to be transposed into each of the 128 possible starting pitches for matching. He presented a means of ranking answers based on the level of similarity to the query. Examples were presented that showed where the techniques succeed and fail.

Clausen et al. [23] built an index on absolute pitch of notes quantised to a small time unit, so that rhythm and stress were incorporated into the matching process. Searching for answers to a query involved searching at each possible starting pitch. They proposed a technique that would limit the cost of this search by building a set of smaller indexes. This is discussed in more detail in Chapter 3.

Francu and Nevill-Manning [52] built an index on 200 millisecond fragments of each channel, excluding any such fragments that would be extremely difficult to sing, and used this as the first stage in the matching process. The second stage involved a more comprehensive matching of the query against the pool of potential answers retrieved via the index. The matching process extends the idea of Ó Maidín in that the key in which the piece and query match with the minimum difference in pitch is determined. The approach was evaluated by observing its effectiveness at retrieving pieces given a sung query. The system was shown to be more effective in answering queries than six musically untrained listeners.

Pickens [101] explored the use of n-grams on a collection of folk-songs. He applied Bayesian inference networks using unigrams and bigrams. He found that the techniques were quite successful for known-item searches consisting of seven or twelve note incipits on the collection.

Sonoda and Muraoka [122] represented melodies with relative pitch and duration. They used a combination of "short DP matching", that is, using dynamic programming to match substrings of length n between the query and potential answers. Once the top 100 results were retrieved, full dynamic programming matching was applied. Evaluation of the system was in terms of speed, space consumed by the index, and accuracy. They used 120 manual queries to test their system with a database of 10,000 randomly generated melodies and 206 real tunes. Accuracy was stated as being 92.5%, where success in the known-item search was defined as being retrieved as the top ranked item.

Other methods explored include the encoding of whether a note is stressed or not, (for example, Schaffrath [116], Bakhmutova et al. [6, 7]). Stress may be important for ranking similar melodies, since experience suggests that those melodies that only differ on an unstressed note are more similar than those that differ on a stressed note. Both Schaffrath and Bakhmutova et al. used diatonic information, that is, the notes were numbered according to the note number in the scale or key of the music. Schaffrath stored the music in the STAIRS text retrieval system. He successfully used the combination of the retrieval system and his collection of folk songs to perform statistical analyses of folk songs from different countries (for example Germany and China). Bakhmutova et al. used their method of calculating melodic similarity to locate variants of Russian and French folk songs.

2.9 Summary

In this chapter we introduced the various fields that form the background to this thesis, that is, music, music psychology, information retrieval, musicology, and music information retrieval. We also discussed the needs of potential users of MIR systems.

Music perception research informs us of some factors in determining similarity between melodies. It also provides us with clues on how simultaneous musical parts are perceived. This may aid the development of better algorithms for separating a performed musical work into its parts and extracting a melody.

Some approaches have been applied by musicologists to analyse and compare pieces of music. In addition, various methods have been implemented for MIR systems. Most of these haven't been evaluated for effectiveness using IR evaluation techniques, and those that do have used known-item searches or a set of variations on a theme as their basis for evaluation.

We discuss matching techniques in more detail in Chapter 3. Their application to music is described in Chapter 8.

Chapter 3

Matching Techniques

Since our aim is to be able to search for melodies in a music collection, we have surveyed techniques for matching and searching. Many techniques have been proposed for music matching, including dynamic programming [93, 97, 133], n-gram techniques [40, 133], bit-parallel techniques [80, 93], suffix trees [21, 78], indexing individual notes for lookup [23], feature vectors [76], and calculations that are specific to melodies, such as the sum of the pitch differences between two sequences of notes [99]. Several of these techniques use string-based representations of melodies.

Besides their use for text, string matching techniques have been applied to a range of different types of data. The data to be matched is usually represented as a sequence of symbols which may be printable characters. An example is the representation of genomic data, which usually consists of sequences of the characters C, G, T, and A, representing the four nucleotide bases found in DNA. A further example is the representation of speech as a sequence of phonemes. Melodies can be represented by a sequence of notes, or musical intervals stored as a string, therefore string matching techniques are potentially useful for melodic matching.

In this chapter we survey different types of matching techniques that have been proposed for music and string databases. We also discuss a few other matching techniques that are not entirely string-based, but have been applied to music matching problems. We describe both exhaustive techniques and indexing approaches for both exact and inexact matching. We then discuss the potential of these techniques for MIR systems. The techniques that we have explored in our experiments are discussed in Sections 3.4 and 3.5.

3.1 Terminology

A *string* is a sequence of symbols, usually characters. A *pattern* p is the string of characters that is being searched for within a body of *text* t. In this chapter we assume the pattern consists of m symbols and the text n symbols.

String matching can be *exact* or *inexact*. String *searching* usually refers to locating exact matches within the text, whereas *matching* is a more general term. All these matching problems are part of the more general group of problems known as *pattern matching*. The pattern matching problem domain includes matching two-dimensional structures and trees, in addition to strings [1]. For inexact matching the problem is expressed in different ways: find all matches that have up to k errors, and use a scoring approach to measure how similar strings are. The types of errors that occur are *mismatches*, where two characters being compared are different, *inserts* and *deletes*, where a character is inserted in one string to align it with the other string. As inserting in one string has the same alignment effect as deleting from the other string, the two types of error are defined as *indels*. A *gap* is a contiguous sequence of inserts. Consider the example of matching "caterwaul" with "call". If the first "l" in "call" is a mismatch against the "t" in "caterwaul", then there is a gap of length five consisting of the letters "erwau". Each of the letters in the gap is an indel.

Exhaustive matching techniques scan the entire text for matches to the pattern. The pattern is often preprocessed to make the matching process more efficient, but the text itself is not. The alternative to exhaustive matching involves preprocessing the text to make an index. These techniques we have termed *indexing techniques*.

3.2 Exact Matching

Some MIR systems use exact matching techniques for query processing [71]. However, inexact matching is generally more useful for MIR. Some techniques that have been developed for exact matching have been modified for inexact matching, but others are not easily modified. Here we briefly discuss existing exact matching techniques, but explain the bit-parallel method of Baeza-Yates and Gonnet in more detail, as this is the basis of several techniques for inexact matching [146], and has been extended for use in MIR systems [80, 91].

Several of the most efficient algorithms for finding exact matches are the Boyer-Moore-

Horspool, Rabin-Karp [118], and Baeza-Yates and Gonnet [2] algorithms. The brute-force approach to exact matching commences comparison of the pattern at each location in the text. This requires $O(mn)$ character comparisons, though in practice the cost is less since checking the pattern at a particular position can stop as soon as a mismatch occurs. The Knuth-Morris-Pratt approach to string matching improves on the brute force technique by first analysing the pattern to determine where to resume matching when a mismatch occurs, eliminating the need to back-track. As a result, the algorithm has a maximum of $O(m + n)$ comparison operations. The Boyer-Moore algorithm improves on this technique by matching from right to left within the pattern so that larger skips are possible. When the characters at each end of a pattern are examined, the largest skip that can be made is determined. The algorithm was further improved by Horspool for long patterns [2]. Rabin and Karp applied a hashing technique in their approach. At worst the algorithm is $O(mn)$ but this is highly unlikely [118]. More recent work in exact string matching was surveyed by Crochemore [28].

For short patterns, particularly those of length less than five, the Shift-Or bit-parallel approach suggested by Baeza-Yates and Gonnet is faster than the Boyer-Moore family of techniques [2]. It is also easier to adapt to inexact matching than the other algorithms mentioned above [146]. This approach performs at its best when the pattern to be matched is smaller than the word-size of the machine it runs on. It uses $O(n)$ time.

A bit-pattern is created for each character in the alphabet representing the positions at which the character occurs in the pattern. This bit pattern is used to calculate the state at each position in the text during matching. Moving from one position in the text to the next involves applying a shift-left operation to the previous state and then using logical Or of the current text character's bit pattern with the state bit pattern. A complete match is found once the most significant bit of the state is set to zero. For example, consider the pattern "a·cat", the text "ta·gat·a·cat" and the alphabet {'a', 'c', 'g', 't', '·'}. An array T is created that contains bit patterns as shown in Table 3.1. The letter 'a' occurs at positions one and four in the pattern, therefore the value for the array element representing 'a' contains 10110, which is set to zero at positions one and four.

A state variable keeps track of the matching process. Table 3.2 shows the state values at each stage of the matching process. The state is initialised to 11111. Zeroes in the state bit pattern show the length of matches, so the 0 in the state value occurring under the third character of

		T				
i	char	t	a	c	·	a
0	a	1	0	1	1	0
1	c	1	1	0	1	1
2	g	1	1	1	1	1
3	t	0	1	1	1	1
4	·	1	1	1	0	1

Table 3.1: *Preprocessing for the Baeza-Yates and Gonnet algorithm. A bit pattern representing the occurrence of each alphabetic character in the pattern is stored in array T. Columns one and two are the array index and characters of the alphabet respectively. A 0 is stored at positions where the character occurs in the pattern ("a·cat" shown in reverse order).*

Text	t	a	·	g	a	t	·	a	·	c	a	t
$T[x]$	01111	10110	11101	11111	10110	01111	11101	10110	11101	11011	10110	01111
State	11111	11110	11101	11111	11110	11111	11111	11110	11101	11011	10110	01111

Table 3.2: *Matching process for the Baeza-Yates and Gonnet algorithm. The bottom row shows the state of the matching process at each position in the text. When a zero exists in the state, then there is a partial match at that point. A zero in the most significant bit indicates a complete match between the pattern and the text.*

the text (11101) indicates a match of length two ("a·").

Lemström et al. used a variation of Baeza-Yates and Gonnet's algorithm for music matching [80]. They applied it to absolute pitch matches against a polyphonic music collection. Thus the algorithm had to be modified to handle data that was not really a string. Their approach was to modify the main matching process to allow a pattern character to be within a set of pitches that occur at a particular instant in the text. The technique consisted of exact matching of pitches and was modified to allow transposition invariance, that is, matches to a melody query would be found in the database regardless of the key in which the query is presented.

3.3 Inexact Matching with k Errors

The bit-parallel techniques described above have been extended to solve the matching problem with k errors. Baeza-Yates and Gonnet showed extensions to their algorithm that allow k mismatches and a few class-based matches, such as “don’t care” symbols, and ranges of characters [2]. The mismatch problem is dealt with by allocating more than one bit per position and using a sum instead of logical Or to calculate the state in each iteration.

Wu and Manber extended the algorithm to allow inserts and deletes in addition to mismatches. The algorithm uses k bit arrays, where k is the number of permitted errors. The algorithm can be extended to have different penalties for inserts, deletes or mismatches by making a single instance of these count for more than one error. For example, if deletes are to have twice the penalty of mismatches or inserts, then a delete equates to two errors. The algorithm is $O(kn)$ in time [146].

McNab et al. [91] applied Wu and Manber’s technique to music matching. In the preprocessing stage they created two matrices, one to represent pitch and the other to represent duration. For their work, all note durations were quantised to 16th notes (semiquavers). They found that the state-matching approach was less discriminating than the dynamic programming approach based on Mongeau and Sankoff, but much faster. However, their tests for discrimination involved using a single value for the number of errors, regardless of the pattern length. As a result, short queries returned the entire collection.

Ghias et al. [54] used Baeza-Yates and Perleberg’s matching algorithm (described in [3]), which is $O(n + R)$ time in the general case, where R is the number of positions where the text and pattern match [3], but $O(mn)$ in the worst case when the pattern and text are entirely made up of the same repeated character [54]. For matching with k errors, the algorithm makes use of a pattern partitioning technique. In practice the algorithm is faster than that of Wu and Manber. Ghias et al. applied the algorithm to contour strings representing melodies in a database of MIDI files.

3.4 Dynamic Programming

Dynamic programming is the name given to an approach that evaluates small subproblems that can be combined to produce answers to a larger problem without revisiting the small problems.

The class of string matching techniques based on dynamic programming use a scoring matrix to find regions of similarity in data.

Dynamic programming-based string matching in its most basic form involves creating a two-dimensional array that stores the results of comparisons between two strings. Unlike techniques that locate occurrences of a pattern, the purpose of dynamic programming is to calculate a score that states how similar (or different) the two strings are. This similarity can be local — the score of the best matching substrings — or global — the overall quality of the match. Whether the matches are local or global, the alignment of the strings can be determined once the two-dimensional array has been filled.

The technique was originally developed independently by about nine different researchers [115], including Needleman and Wunsch, and Sankoff, although the algorithm originally published by Needleman and Wunsch ran in cubic time [59]. The use of an edit distance to determine similarity between strings was first proposed by Levenshtein and it is often referred to as the Levenshtein distance. The edit distance comprises of a count of inserts, deletes, and substitutions required to make one string match another exactly. The combination of the dynamic programming technique and a distance measure is frequently used for the comparison of DNA strands [59], text, musical data [93, 97], and audio signals [50, 115].

3.4.1 Example Using Longest Common Subsequence

One of the dynamic programming family of techniques calculates the longest common subsequence (LCS) in two strings. As an example, consider the strings "the·fact·that" and "the·fat·cat". The longest common subsequence would consist of the characters "the·fat·at". There are two possible alignments that result:

the·fac**t·**th**at**

the·fa-**t·**c-**at**

and:

the·fac**t·**th**at**

the·fa-**t·**-c**at**

where "-" represents an insertion, and the common subsequence is shown in bold.

A formula for calculating the score for longest common subsequence is shown below:

$$a[i,j] = max \begin{cases} a[i-1,j] & i \geq 1 \\ a[i,j-1] & j \geq 1 \\ a[i-1,j-1]+1 & p(i) = t(j) \text{ and } i,j \geq 1 \\ 0 & \end{cases} \tag{3.1}$$

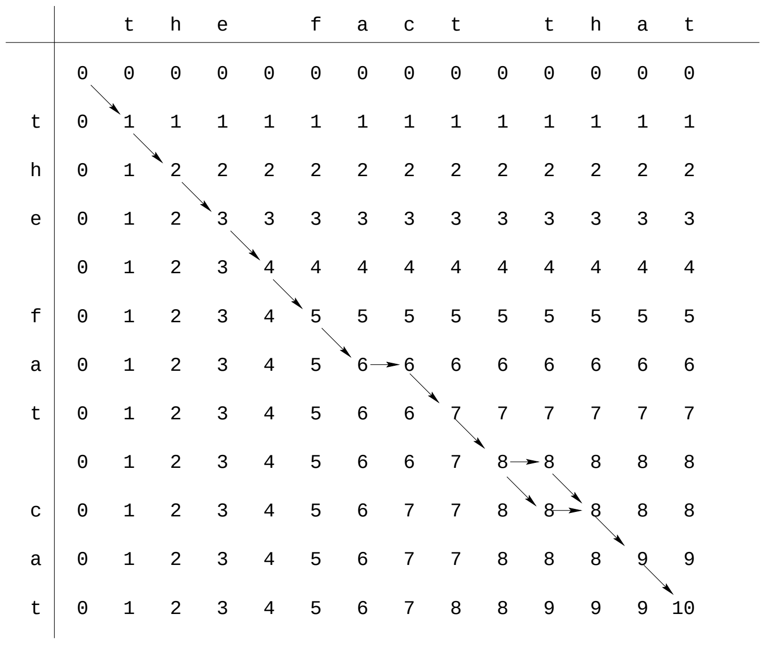
where a represents the array, p the pattern, and t the text. If each common character found in sequence contributes a score of 1, then the score produced by the technique for the example would be 10. Figure 3.1 shows the values in the array as a result of applying the formula. For example, at position $[1,1]$ the first character of each of the strings align and are equal, therefore the value stored at the position is 1 $(0+1)$. The number in the bottom right corner of the array is the length of the longest common subsequence of the two strings, that is, 10. The actual subsequence can be determined by tracing backwards through the array.

LCS can be implemented by creating an $n \times m$ array and using a nested loop to fill it with the values according to the formula, and completes in $O(mn)$ time.

3.4.2 Dynamic Programming Variations

The longest common subsequence algorithm adds one to the previous score if the current two characters match and otherwise uses the previous score. The formulation has no penalty for gaps between matching symbols. There are other kinds of scoring technique that have been developed for use with dynamic programming. Most of these vary the weights applied when contributing to the score at each point in the matrix. Variants include global alignment, local alignment, and longest common substring.

Global alignment (also known as Needleman-Wunsch alignment [59]) typically assigns a positive score for a match and negative scores for mismatches, inserts and deletes (indels). A commonly used set of weights is 1 for a match, -1 for a mismatch, and -2 for an indel.



		t	h	e		f	a	c	t		t	h	a	t
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t	0	1	1	1	1	1	1	1	1	1	1	1	1	1
h	0	1	2	2	2	2	2	2	2	2	2	2	2	2
e	0	1	2	3	3	3	3	3	3	3	3	3	3	3
	0	1	2	3	4	4	4	4	4	4	4	4	4	4
f	0	1	2	3	4	5	5	5	5	5	5	5	5	5
a	0	1	2	3	4	5	6	6	6	6	6	6	6	6
t	0	1	2	3	4	5	6	6	7	7	7	7	7	7
	0	1	2	3	4	5	6	6	7	8	8	8	8	8
c	0	1	2	3	4	5	6	7	7	8	8	8	8	8
a	0	1	2	3	4	5	6	7	7	8	8	8	9	9
t	0	1	2	3	4	5	6	7	8	8	9	9	9	10

Figure 3.1: *LCS array created when matching the strings "the fact that" and "the fat cat". At position* $[1, 1]$ *the first character of each of the strings align and are equal, therefore the value stored at the position is 1* $(0 + 1)$*.*

$$a[i,j] = max \begin{cases} a[i-1,j]+d & i \geq 1 \\ a[i,j-1]+d & j \geq 1 \\ a[i-1,j-1]+e & p(i)=t(j) \text{ and } i,j \geq 1 \\ a[i-1,j-1]+m & p(i) \neq t(j) \\ 0 & i,j=0 \end{cases} \tag{3.2}$$

where d is the cost of an insert or delete, e is the value of an exact match, and m is the cost of a mismatch.

Local alignment, also known as Smith-Waterman alignment, uses a similar technique to global alignment but does not allow a score to become negative. The final score for the match is the maximum of each of the local maxima. This represents the best match between substrings of the two strings.

$$a[i,j] = max \begin{cases} a[i-1,j]+d & i \geq 1 \\ a[i,j-1]+d & j \geq 1 \\ a[i-1,j-1]+e & p(i)=t(j) \text{ and } i,j \geq 1 \\ a[i-1,j-1]+m & p(i) \neq t(j) \\ 0 & \end{cases} \tag{3.3}$$

It is also possible to use dynamic programming to calculate the longest common substring by setting the score in the array position to zero as soon as a mismatch or indel occurs. The largest value in the array then represents the length of the longest common substring. As with the other methods above, it is possible to trace back through the array to determine the substrings that have matched.

Another variation in the use of dynamic programming is the application of different scores for *gaps*. A gap occurs when a long run of inserted characters occurs during matching. For example, if the word "birthday" were to be matched against the word "bay", there would be a gap of five characters. For some applications the penalty for long gaps would not be as heavy as would occur if the normal indel penalty is used for each character position. This approach has been applied to genomic matching [59].

Another extension to the matching process has been the application of a substitution matrix that is used to determine the weights to be applied for any particular pair of characters that are being compared. One of the substitution matrices used in genomics is the PAM matrix (discussed by Gusfield [59]). This contains values based on how frequently a particular DNA component is substituted for another in highly similar sequences of DNA. Mongeau and Sankoff [97] applied a similar approach to melody comparison, in that they used different weights depending on which notes were being matched. They penalised dissonant melodic intervals more than consonant (pleasing) ones on the basis that consonant substitutions are more likely than dissonant ones.

3.4.3 Other Issues

An advantage of dynamic programming techniques is the ability to define the way in which strings are to be compared. It is possible to reward symbols occurring in sequence regardless of intervening symbols by using longest common subsequence, restrict to exact matching, or fine-tune weights for mismatches and indels in order to suit the types of data being compared.

There are two disadvantages of dynamic programming-based string-matching techniques. For long strings they use a large amount of memory ($O(mn)$). Also, being $O(mn)$ in speed, they may be slow compared to an indexed approach or a technique optimised for matching with k errors.

Storage requirements can be reduced by only storing the current and previous rows of the array, which is feasible because for typical use of dynamic programming the value in each cell only depends on those on the previous row or previous column. If alignment of the strings is also required, then a recursive algorithm to determine how the strings have been matched can be implemented (described by Gusfield [59]).

There are techniques that address the speed of the DP algorithm, making it subquadratic [64]. However, a technique that has been more useful in practice is to reduce the number of cells in the array that are calculated. Techniques that do this limit the number of consecutive errors, or the total number of errors, which reduces the proportion of the array that must be calculated [140], thereby reducing the time required.

3.5 Indexing

MIR systems could benefit from an indexing approach to speed up melody queries. Several researchers have developed indexes for melody retrieval systems [11, 23, 76] while others have created indexes on non-melodic features of music [138]. An indexing approach consists of a search structure and a set of information about the location of occurrences of each index term. Here we discuss index terms and different types of index structure.

3.5.1 Index Terms

In textual databases, each word occurring in documents is indexed. Looking up the word in the index would result in a list of documents containing the word being retrieved. For melodic data there are no words as such. Individual notes [23] or melodic phrases [71] could be indexed. If the index terms are individual notes, many look-ups would be required for a single query. Further difficulties need to be solved when answering queries that are inexact because query terms may not be present in matching pieces. Using melodic phrases, however, limits the start position of a melodic query.

Another possibility for melody indexing is to use *n-grams*. An n-gram is a set of n adjacent symbols in a stream of text or other symbols. For example, the 3-grams (tri-grams) of the phrase “the·cat” are:

```
the
he·
e·c
·ca
cat
```

An n-gram index contains all possible n-grams for documents in the collection.

Sometimes the terms to be indexed are “features” of complex data. Using a feature histogram involves choosing certain features of the item to be indexed and creating an array of values for each item. An index is created of these items. Special structures such as r*-trees are used to cater for the multi-dimensional nature of the index terms. These structures are typically used for image retrieval, in which case the features include colour and texture. As colour histograms can be of fairly high dimensions, summary features such as the average colour are used in the first instance to filter potential answers [147]. For music retrieval, features that have been used are a “tone distribution” and “tone transitions” relative to a base pitch [76]. When evaluated, the combination of the two features was shown to be more effective than when used individually, with 75% of melodies being retrieved in the top five answers.

3.5.2 Index Structures

There are several potential methods for indexing the terms that occur in a string database. Two components are required: a search structure and method for locating the term in the index, and a set of information about the locations of terms in the collection. Potential indexing structures include arrays, hash indexes, and various tree structures, such as binary trees, B-trees, and tries.

In some situations the best approach for looking up terms is to have an array of sorted terms and to apply a binary search. This is applicable when the set of terms is sufficiently static and small enough to be kept in memory [144].

Hashing involves computing the address of a record based on the user's query. Storage and retrieval using hashing is O(1) if records are stored without overflow in the hash-table. This approach is appropriate for data that is relatively static and where range queries or ordering are not needed. Hashing in which order is preserved has been applied to genomic data [139]. A perfect hashing function was applied that involved a simple mapping from a character to a component of a key. Wilbur and Lipman used a formula that makes use of the small character set for genomic data by creating an array of size s^n where s is the size of the alphabet and n the n-gram length. For example, characters in the four character set for DNA: 'A','C','G', and 'T', may be represented by 00, 01, 10, 11 respectively when calculating the key. A 4-gram "AACG" would then have the key 00000110, that is, 6.

Another search structure that has been proposed for string matching is the suffix tree, a tree that has each node representing a character in a string, and all suffixes of a string are stored in it. There are several formulations of this type of tree. The *trie* is used for storing multiple strings, using the individual characters in sequence as look-up terms. The Patricia tree is another variation, that stores the next string position that needs to be checked in internal nodes to determine which path in the tree to follow.

Chou et al. [21] used the Patricia tree to store chordal representations of melody notes in each bar of music. Lemström et al. [78] applied a suffix-trie to a specific depth and suffix trees for the remainder of the index structure to their melody search problem. Tries were not used for the entire structure due to the space cost. Lemström et al. have since changed their approach to one that extends Baeza-Yates and Gonnet's algorithm [80] as a means of locating matches.

In addition to their application to text and music, suffix tree algorithms have been extended to other types of data such as matrices [55]. A new generalisation of the trie, called the *burst* *trie* has recently been explored in detail [63]. In this structure, trie nodes are combined with containers of another type. A container is "burst" into a new trie node once some criterion is met. The structure has been shown to be more efficient for certain types of data than several other commonly used structures. In particular it was shown to be very efficient for music n-grams of length seven using our *all-channels* data-set (see Chapter 5).

A common approach to indexing is to use B-trees (or similar) as a search structure. This class of trees allows reasonably efficient ($O(logN)$, where N is the number of records) retrieval of records and is guaranteed to have a balanced structure. The B-tree was used by Chou et al. [21] to implement their PAT-tree of chordal representations of notes.

In addition to providing a fast way of locating a term's record, the search structure must provide access to information about the term's locations in the database. The structure that has been shown to be the most efficient for information retrieval is known as an *inverted list* [153]. This consists of an index of all terms in the collection. For each term in the index there is a list of documents that contain the term. The information contained in the list for each term-document pair varies depending on the application requirements, but may include the number of occurrences and the position within each document that the term occurs.

Clausen et al. [23] used inverted lists for the music searching problem. Individual notes were stored as ordered pairs of quantised start time and MIDI pitch number. Queries and answers were required to have the same metrical position within a bar to be considered a match. A restricted form of inexact matching was proposed for systems using this type of index, consisting of sets of alternative notes, but the use of unrestricted inexact matching was not discussed. The size of the index for a database of 12,000 classical pieces containing 327 Mb of MIDI data and over 33 million notes, was 110 Mb when uncompressed and 22 Mb when compressed with Golomb encoding. This index requires queries to be in the same key and octave, or the index to be searched 128 times. A set of smaller indexes was proposed that makes use of the "Chinese Remainder Theorem" to reduce the number of searches required for matching occurrences of a query in any key.

3.5.3 Index Heuristics

One technique used to reduce the size of an index is called *stopping*. This involves removing the most frequently occurring words from the index. For example the word "the" occurs in virtually every text document, so a query on it would not be useful. If the list structure used is proportional in size to the number of documents containing the word, then the list for very frequent words would be much larger than those for more useful query terms.

Zobel and Dart [151] built n-gram indexes as part of a two-phase approach (coarse-and-fine search) to locating matches. A search involved looking up all n-grams of the query in the index and retrieving the lists of answers that contained any of these n-grams. The list of answers was ranked based on the number of n-grams each answer had in common with the query. The best k answers were then examined closely using a distance measure. They found that the combination of an n-gram index and a simplification of Ukkonen's n-gram-based measure was very effective for matching personal names and spell-checking.

The coarse and fine approach was applied to very large databases of nucleotides by Williams and Zobel [142, 141]. Stopping and various data compression techniques were used to reduce the size of the index. The indexes built were tested against the current method of retrieving matching nucleotide sequences and out-performed it in speed.

Once index terms are looked up, they must be applied in some way to calculate the similarity between the query and the set of potential answers. The similarity measures applied to the words occurring in documents could be used with n-grams. We discuss some of these in detail in Chapter 8.

3.6 Summary

Many techniques have been proposed for music matching. In addition to music matching techniques, much can be learned from the matching techniques applied to other types of collection, such as text, genomic, and speech data.

The notes in a melody string can be represented in a variety of levels of detail. They can include information about the pitch, duration and stress of notes or just the pitch or rhythm. The alphabet size for music strings depends on the representation chosen, but unlike genomic databases, and more like text, the distribution of symbols within a collection is not even. Music strings often represent relative pitches instead of absolute ones so that the representation is key independent. This allows matching of music that is played or stored in different keys. However, matching in different keys can also be achieved with indexes that store notes as absolute pitch by searching the index multiple times, or using a set of multiple indexes such as those proposed by

Clausen et al. [23]. We discuss the representation of music as strings in more detail in Chapter 6.

Disregarding the representation and formulae used for measuring similarity, matching techniques fall into four main categories, exhaustive exact matching, exhaustive inexact matching, indexed exact matching, and indexed inexact matching. For reasons elaborated in Chapter 2, techniques that support inexact matching are more useful for MIR purposes. Exhaustive techniques such as dynamic programming and state matching are useful if they can provide greater precision than practical implementations of indexing techniques. Of the indexing techniques proposed, there is no evidence of a best approach; however, for n-grams, burst tries or a form of hashing may be an efficient choice.

In this chapter we focused on the algorithms and data structures that are behind matching techniques, and described how these have been applied to music matching. Further details about how the similarity of melodies is measured is discussed in Chapter 8.

Chapter 4

Methodology for Music Retrieval Research

The aim of music information retrieval research is to build effective MIR systems that answer queries satisfactorily and rapidly. There are various areas that require study to establish the best techniques for MIR, including music perception, music similarity measurement, and efficient musical query processing. We establish a framework for the development and testing of MIR techniques, with particular attention to retrieval effectiveness. We demonstrate this framework by developing several techniques for melody matching, and applying the evaluation techniques described in this chapter. Our approach to music similarity measurement, making use of results from music perception research. Within the domain of music similarity measurement, there are many variables to be addressed, such as the portion of the music that should be used for matching, how should these portions be compared, and how these methods of comparison can best be implemented.

Deciding which portion of the music should be compared is partly an efficiency consideration, as an exhaustive comparison of all aspects of all pieces of music results in combinatorial explosion. Moreover, such an approach may result in the user being swamped with poor matches. Sensible choices for the design of algorithms can be made through the use of information about how people perceive music. Regardless of the choice, the approach would need to be validated. Not only should the efficiency of the approach be examined, but the effectiveness in producing answers to queries should be measured. In this chapter we describe techniques used to evaluate IR systems and evaluation approaches in other existing MIR research. We then present the methodology we have chosen for our MIR research and our experiences in obtaining the data necessary for the evaluation process. Finally we discuss the applicability of our methodology to MIR in general.

4.1 Evaluating Information Retrieval Systems

Information retrieval systems require testing to determine how successful they are at retrieving answers that a user perceives to be relevant. In the context of text retrieval, such testing is achieved with a collection of documents and an associated collection of queries. For each query, there is a set of relevance judgements, where users have evaluated each answer to determine whether it is relevant to the query. Relevance must be defined before such evaluations can proceed. There is some variation in how this can be defined [13]. A document can be considered relevant if it answers the information need represented by the query, or if it discusses the same topic, regardless of whether it actually answers the user's specific question.

To determine how effective a system is at answering users' queries, various measures can be applied. The best-known of these are recall and precision. Recall is calculated as the proportion of the relevant documents that have been retrieved, expressing how complete the answer set is. Precision is the proportion of retrieved documents that are relevant, expressing how accurate the answer set is. These can be combined by, for example, averaging precision at 0%, 10%, 20%, ... , 100% recall [144]. Other measures of retrieval performance are also used: one can compare the rank of the first relevant answer, the rank of the first irrelevant answer, and so on. In addition to determining the recall or precision of the results and comparing these to those of other retrieval techniques, it is usually necessary to check the statistical significance of the results, with a test such as the Wilcoxon test [150].

The above tools allow different similarity measurement techniques to be compared in a consistent manner. However, a further issue is that measured retrieval effectiveness tends to differ, depending on the data collection used as a basis for comparison [151]. The TREC conferences address this problem by having many competing research teams apply their techniques to the same collections.

4.2 Evaluation of MIR systems

As with other IR systems, MIR systems need to be objectively evaluated to determine whether they produce useful answers to queries posed by users. MIR systems can be evaluated in a similar manner to other IR systems. As with traditional IR, we need to decide what is a query, what is an answer, and what is meant by relevance, that is, what constitutes similarity.

The usual definition of an ad hoc music query is a melody fragment consisting of a sequence of notes. These could be presented by, for example, singing, playing on a keyboard, using a graphical representation, or by entering a sequence of symbols [10, 11, 71]. Some researchers have built systems that try to answer non-melodic queries [20, 34, 80]. The information need of the user may be that: they half-remember some piece of music and would like to know what it is; they wish to locate an on-line version of some favourite song; they wish to find other performances of a given piece of music; they wish to find out the source of the inspiration for their composition; or they wish to find web-sites that are violating copyright restrictions on a given piece of music. For legal queries, users may wish to check if works are similar enough to be in breach of copyright.

From a user's perspective, an answer to the first kind of music information need is relevant if it is the same as the piece of music that they half-remembered. For the copyright search, relevant answers are those that are similar enough to be considered so in a court of law. Similarity measurement is likely to require a basis in knowledge about how we assess musical similarity. From a practical perspective, we would need to be able to measure relevance of answers to a set of queries in order to test our MIR systems.

Some researchers have evaluated the retrieval effectiveness of their systems, using a variety of methods:

Mongeau and Sankoff [97] used two compositions by Mozart as their test data. Cluster analysis was used to show the similarity of the different variations. This was based on the scores obtained by comparing pairs of variations using their dynamic programming algorithm. The results were evaluated by comparing the results to the authors' intuitive notions of the similarity of the different variations and theme used.

Kageyama et al. [71] used one of the melodies within the database as a query and measured success in terms of the rank the melody received in the set of answers. This is sometimes called a "known-item search". This approach showed that their dynamic programming approach successfully retrieved answers to queries based on songs in the collection of 500 songs. The database itself was a monophonic collection of melodies.

Downie and Nelson [44] used normalised precision to evaluate a wide range of n-gram representations of melodies in a collection of 9354 folksongs. Downie used a form of known-item search approach in his evaluation of techniques. In addition, analysis of the uniqueness of intervals and n-gram occurrence was used to make recommendations for n-gram use [42].

Other researchers have also tested the expected uniqueness of melody fragments of specific lengths in different representations [11, 41, 93], thereby providing an estimate of the query length required for exact match. Estimates vary depending on the representation used and the size of the collection.

However, these approaches do not address the issue of whether matches would be found to be similar by a listener, nor the issue of whether close but inexact matches can be found. There is a danger in relying only on uniqueness-based evaluation as it may be easy to construct a "similarity" measure that discriminates well between melodies in the database but doesn't retrieve good answers to queries.

4.3 Our MIR Methodology

Our methodology is based on the traditional IR approach of using a collection, a set of queries, and lists of relevant pieces for each query. We apply various evaluation measures to the query results, in particular eleven-point precision averages and precision at k items retrieved, although other measures listed above are also useful.

4.3.1 Testing Melody Extraction

In addition to our study of melodic similarity measurement, we have examined techniques for extracting the melody from a polyphonic piece of music. Our methodology for evaluating these techniques involved two stages. In the first stage our aim was to determine which algorithm performed best at extracting melodies as perceived by listeners. This was done by presenting pieces of music to listeners along with the melodies extracted by each algorithm. Each listener ranked the extracted melodies for each piece.

The second stage in evaluating melody extraction involved using it in our melodic similarity measurement experiments to determine how well the techniques would work in practice. Extraction technique was one of several variables tested in our experiments that evaluated similarity measurement. The results of the melody extraction experiments are discussed in Chapter 5, while in Chapter 8 we discuss the effect of the different extraction techniques on similarity measurement.

4.3.2 The Music Collection

In our work we have used a collection of MIDI files downloaded from the Internet. We chose MIDI files as a readily-available source of polyphonic musical data. Non-standard MIDI files were excluded from our collection but duplicates were retained. The collection as used for similarity measurement experiments consists of 10,466 MIDI files. MIDI files do not represent all music, for example, some styles of music from the twentieth century that are not really note-based cannot be represented well in this format. Music that uses tunings that differ from those in standard Western music also presents problems. However, the collection includes a wide variety of music genres and pieces of greatly varying length. Most earlier retrieval experiments used monophonic collections of one or two types of music [44, 71, 93].

4.3.3 First-Stage Queries and Relevance

As an initial source for queries, we selected queries and relevance judgements from the collection by assuming that alternative arrangements or performances of a piece chosen as a query were relevant. Eighty-one pieces were located in a variety of genres that had more than one distinct version within the collection. Versions were detected by locating likely pieces of music via the filenames and then verifying by listening to these pieces. These pieces were only considered to have distinct versions if there were obvious differences in the arrangement, such as being in a different key, using different instruments or having other more obvious arrangement differences, such as in the rhythm, dynamics, or structure.

This set of pieces was used to measure retrieval effectiveness with the standard techniques of eleven-point precision averages and precision at k pieces retrieved. It was used to to determine which matching techniques were most successful and to yield a pool of answers that could be judged for relevance for later experiments. Whether these relevance judgements correspond to — or yield the same ranking of techniques as — human relevance judgements was at that stage an open question.

One of the arrangements in each set of arrangements of a piece of music was randomly chosen to be an “automatic” query. All arrangements of the piece were assumed to be relevant and all other pieces assumed to be irrelevant, thus giving “automatic” relevance judgements.

The automatic queries and relevance judgements were then used to evaluate matching algorithms. The answers retrieved by the best algorithms could then be used as a basis for collecting real relevance judgements.

4.3.4 Obtaining Manual Queries

Having established some basic results using the automatic queries and judgements, it is valuable to use manual queries and judgements. In particular, it is necessary to be able to present the listener with a set of answers to judge as similar or not similar to melody queries. To produce a meaningful set of relevance judgements, the pool of pieces presented to the listener should include a good proportion of potentially relevant answers. The answers found by the algorithms judged to perform well using the automatic queries and judgements are an excellent starting point.

Music retrieval systems may receive queries from a variety of types of user and will be of varying accuracy. For MIR research, there are several potential sources for manual queries. One could collect actual queries posed at music libraries and shops. One could ask a group of volunteers to note down whenever they are reminded of a bit of melody that they cannot put a name to. One could randomly select portions of music pieces and transcribe these as queries. The queries we have collected are created by a musician via an electronic keyboard. Naturally these queries will not be the same as those that are hummed, but may be more like those presented by a music librarian or composer. The advantage of our approach is that we were able to control the selection of pieces to match our pool of pieces with multiple versions.

We gathered manual queries by asking an expert musician to listen to a given set of pieces for which we had located multiple versions in the collection, and to play the melody of these pieces. This musician has perfect pitch and can reproduce melodies that she hears on a keyboard. Each MIDI file that she listened to was from the same set of files used for the automatic queries. Each of these files was randomly selected from the list of different versions of the same piece. This collection of MIDI files was presented to the expert in sequence. She was given the following instructions:

> Please listen to the pieces of music presented to you. After listening to each piece, you are to play a portion of the piece's melody that is long enough (but no longer) to make it recognisable to someone who knows the piece.

The expert was permitted to listen to the piece as often as she wished. She then played a keyboard rendition of a portion of the melody, which was recorded via MIDI to a music sequencing program. For each piece of music it was noted whether the expert knew the piece of music by name, whether it sounded familiar but could not name it, or that it was unknown.

Discussion

There were some problems encountered with the chosen music collection. One problem was that some versions of pieces didn't actually contain what most people would consider to be the melody of the piece. This happened for about two of the pieces that the expert was asked to transcribe. In these situations parts of the accompaniment were chosen as the melody. It was interesting to note that this particular musician would not consider themes that occurred at the start of the piece to be "the melody". The melody was typically chosen to be something that started after the introduction had finished. With the MIDI representation of songs the location of the start of "the melody" did not have the usual cues found when a voice sings the song. As a result, some choices for the commencement of the melody were quite different to that which would be chosen by someone who was familiar with the piece of music. The musician we used is a professional accompanist. We suspect that this focus on the melody as something that starts after the introduction could be a result of this focus of the soloist providing the melody and the accompaniment being purely of a background nature. It would be interesting — but beyond the scope of this thesis — to explore this aspect of the definition of melody further. The use of a single volunteer for the generation of melody queries may have introduced some loss of generality, particularly given the different interpretations of the word "melody". The melody starting points chosen by the volunteer would be significant for the matching process if the database "melodies" to which they are compared are truncated, however, in our melody matching experiments they are not. In general, the notes that were selected still had much in common with those determined by other listeners when comparing melody extraction algorithms (see Chapter 5), that is, they tended to be from parts that were either higher in pitch or entropy.

For some pieces that were unknown to the expert, multiple listenings were required. Also, some mental fatigue may have affected consistency across the set of queries, with later queries being less carefully prepared. The expert was only available for one session of two hours making it difficult to avoid any fatigue that could arise.

This process resulted in a set of 30 manually produced queries. The number of notes in the queries varied from a minimum of 15 to a maximum of 88, with the mean number of notes being 31.9 and the median 28. The expert's queries tended to become longer as she progressed through the set. Collecting 30 queries took approximately two hours.

4.3.5 Manual Relevance Judgements

As mentioned above, our initial definition of whether an answer was relevant was "if it is a version of the same piece of music that the query came from". This was a good first approximation, and describes answers to one type of query. However, to further test queries that are looking for highly similar pieces of music, a different approach is needed. Our approach was to collect relevance judgements for melody queries and to use these to evaluate MIR techniques.

In order to collect judgements from several volunteers in a methodical manner, we developed software that displays pieces that judges were to listen to, and that collected the judgements. The tool developed for this purpose has a web client interface that stores inputs from judges in a database. A web client was chosen because it provides portability and accessibility. The system can be used by any web-browser capable of handling JavaScript and a plug-in for playing media files in MIDI format. The query and answers appear with individual media panels that have standard music player functionality, such as play, stop, pause, and volume.

For each query, the judge is presented with a media panel for the query itself and a number of answers to the query, each with its own media panel. The judge is allowed to listen to the query and answers as many times as they wish, after which they are required to identify:

- For the query, whether the judge:
  - recognises the piece of music and can name it, or
  - recognises the piece of music but cannot name it, or
  - cannot recognise the piece of music.
- For each answer, whether it:

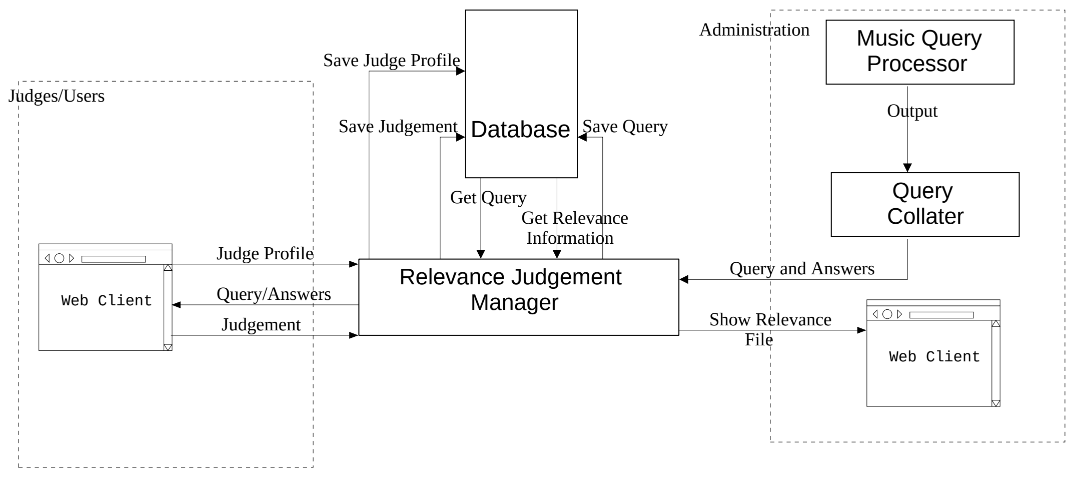


Figure 4.1: *Components of our Judgement Acquisition and Management JAM system for collection of music relevance judgements.*

– is the same as the query or,

– is different to the query.

Inputs from the judges are stored in a database, along with the time taken by the judge to complete each set of answers and queries. An extension of the system is an interface for administration, that allows:

- Managing the set of queries presented to judges.
- Generating relevance files based on judgements received on particular queries.
- Viewing statistics on judgements.

The system is shown in outline in Figure 4.1.

Before assessing the queries, the judges were presented with the following instructions:

> You are about to listen to a collection of pieces of music that may or may not be similar to the given melody that we shall call the "query melody". For each piece, you are to make a judgement as to whether it has the same or very similar melody to the query melody. Please choose the "same" or "different" button as appropriate

> for each piece. You may listen to the query melody and other pieces again if you wish.

The judges were asked to state whether each of the returned pieces was relevant or not. In generating the file of manual judgements, we assumed an answer was relevant if at least one judge marked it as being the same as the query.

To best simulate queries provided by users, we collected human judgements for the set of manual queries provided by our expert. The queries created were fed to the music retrieval system using each of the main promising techniques. The top 20 answers from each method were pooled and presented to the expert user to be judged for relevance.

Results

To create a set of MIDI files to be judged, the top ten answers were selected for each manual query using our "all channels" version of the database and three similarity measures. In addition to these answers, the files considered relevant by inspecting those with similar names to the original were added to the collection to be judged. Twenty-eight of the thirty queries were judged, each query having one or two assessors. The judgements were made by six judges in total. Most of the sets of judgements (30 of 43) were made by those who were musically unskilled or who had only a basic level of musical skill.

We compared the content of the two sets of relevance judgements and found that only five of the twenty-eight queries had identical sets of assessments. Four of the queries in the manual set of judgements were judged to have no relevant answers. In some cases it was clear that the automatic set was missing some answers that would have been found upon a more thorough search of the collection. For example, the automatic set of relevance judgements contained three pieces that matched Auld Lang Syne, but, the combination of applying our search techniques to locate matches and collecting relevance judgements found an extra two pieces, one of which did not have an obviously similar filename. In most cases the intersection of relevant pieces was greater than the difference, but in two cases there were no pieces in common. Further detail about the two sets of judgements is in Table 4.1.

Assessing musical relevance proves to be quite difficult for the listener in some situations. We observed that when the piece on which the query was based was known to the listener, then the judgements were easy to make, with the listener not needing to listen to more than

Total number of manual queries with judgements	28
Queries with identical sets of matches for both kinds of relevance	5
Manual queries with no relevant files in the manual relevance set	4
Queries in which 100% of the manual set matches are automatic matches	9
Queries in which 100% of the automatic set matches are manual matches	11
Queries in which there is no intersection (excluding empty queries)	2
Queries with less than 50% overlap with either set (excluding empty queries)	4

Table 4.1: *Statistics about automatic and manual relevance judgements. The manual relevance judgements were collected using six judges listening to manual queries and a collection of MIDI files. The "automatic" set was collected by listening to MIDI files that had similar file names and confirming whether they were versions of the same piece of music.*

a few seconds of the answer. When the query is not recognised, the listener needs to listen to the query repeatedly when comparing it to the answers, especially where the answers are also unfamiliar. It is an extremely difficult task, especially for those who are not musically trained or experienced. The accuracy of judgements on queries that are not known is likely to be less than that on known queries. We believe that in some cases a "relevant" rating was given, not because the piece was similar to the query, but that it was similar to other answers in the answer set.

A further difficulty is that the MIDI files were presented as is. The user could listen to the entire MIDI file or as much of it as they wished. Sometimes users would listen to the first part of the file and not hear the part that was relevant to the query, a particular problem when the matching piece was minutes or tens of minutes in length. It would have been helpful to mark up the slider bar on the on-screen MIDI player to indicate which region in each piece had best matched the query, but to do so would have been difficult with the tools and time available to us.

To obtain better relevance judgements, participants should judge pieces with which they are familiar and be presented with the portion of the answer that matched the query. Restricting the participants to skilled musicians may improve the quality of the relevance judgements at the expense of representing the range of abilities of potential users of MIR systems. However, despite these shortcomings, we believe that the relevance judgements we obtained allow evaluation of an MIR system with reasonable confidence, as illustrated in the chapter discussing experiments with manual queries.

4.3.6 Second-Stage Melodic Similarity Experiments

Once manual queries and judgements are obtained these can be used for further experiments on melodic similarity. We used the same approach as that of our earlier similarity experiments. The manual queries and judgements were compared with the earlier queries and relevance data to see if differences in comparing algorithms were apparent. The results of these experiments are reported in chapter 10.

4.4 Summary

In this chapter we have discussed the basics of the techniques for evaluating IR systems. We have discussed approaches used by other researchers in the MIR field and followed this with a description of our own approach. We recognise that there are some limitations in terms of types of music that can best be handled using the MIDI collection we have chosen as our data-set, but that it has the widest range of styles of music in any collection being used for evaluation of music retrieval to date. The method of query generation used in our first-stage experiments is only a first approximation to manual queries. However, this may be a useful approach for some types of real query, such as when matching pieces that are presented as a whole, rather than a melodic fragment. We then collect manual queries and relevance judgements to better simulate the information needs of potential users of MIR systems. These form the basis of our second-stage evaluation experiments. In terms of a general approach to MIR research, our model can be applied to the evaluation of MIR techniques, regardless of the collection chosen and the type of query.

Chapter 5

Melody Extraction

Melody extraction is the first of three phases that make up our approach to music matching. The purpose of a melody extraction technique is to identify sequences of notes that are likely to correspond to the perceived melody. Given the volumes of music available online, it is not practicable to do this by hand. The effect is to reduce the musical data to be searched to that which the users are most likely to present as queries, which helps to increase the precision of the melody matching process. Our aim, in this chapter, is to describe the melody extraction techniques that we have explored, and report on an experiment that tested the effectiveness of each of these techniques.

5.1 Melody Extraction Issues

The data upon which we base our experiments is a collection of MIDI files, which usually contain note events for each instrument in separate channels. Some MIDI files contain information such as bars and keys but this is not guaranteed. The individual instruments occurring in a MIDI file can be playing chordal parts, consisting of several simultaneous notes; there may be more than one "counter-melody" occurring simultaneously; and the perceptible melody can move from instrument to instrument. That is, the melody may not be present in any individual part, thus it is far from clear which sequence (or sequences) of notes should be used in the comparison. Additionally, some "notes" are actually percussion and therefore have no pitch. Extraction of all sequences would lead both to combinatorial explosion and large numbers of sequences with little perceived connection to the original music; indeed, most such sequences would sound like no more than a random series of notes. For example, the music fragment at Figure 5.1A has 22 instants at which notes commence. If it were a monophonic sequence, that is, only one note occurring at a time, then there would be $\binom{22}{10} = 646646$ possible subsequences consisting of 10 notes. However, as there are several notes occurring simultaneously, the number of subsequences is considerably larger. Moreover, if rhythm is ignored, we can find in the example the first seven notes of the Camptown Races and Silent Night, the first six notes of Frère Jacques, Danny Boy, and the choruses of Just Because and I Was Made for Lovin' You, the first five notes of Mary Had a Little Lamb, Go Tell Aunt Rhodie, The First Nowell, Good King Wenceslas, Oh Little Town of Bethlehem, Hey Jude, Au Clair de la Lune, Yankee Doodle, and Oh Susannah, and the first four notes of many other tunes. This list of songs is by no means exhaustive and clearly illustrates the need for selectivity in the matching process for meaningful results to be found.

There does not appear to have been much research on the problem of extracting a melody from a piece of music. Several systems have been built that use a simple approach to melody extraction. For example, the approach of Ghias et al. [54] was to ignore percussion and apply other simple heuristics (not described in their paper) to collect melodies. Blackburn and de Roure implemented a "melody" extractor that retained the lowest pitch notes of each channel for matching [11].

Researchers have discussed the splitting of polyphonic-style music into its parts using rules, largely based on pitch proximity and usually for a set of music with a fairly uniform style [17, 87]. Marsden et al. applied the proximity principle in a rule-based approach, and attempted to refine the rules to split the parts of a Bach fugue [87]. Charnasse and Stepien [17] allocated notes to parts when transcribing German lute tablature. Their approach was also rule-based, mainly using pitch proximity. The number of parts was estimated from chords in the music and the actual parts were produced by processing musical chunks delimited by chords that contain the maximum number of voices. Similar psychoacoustic-influenced approaches are used when analysing sound wave-forms. In this case, the components being grouped are parts of a sound, with the aim of separating different sources of simultaneously occurring sounds. [48, 89]

According to Francès [51], the notes with the highest pitch are usually heard as the melody, unless they are monotonous. As discussed earlier, the most important factor in the grouping of musical notes is proximity in pitch [31]. Other principles that apply are timbre and volume; however, these are less important than proximity.

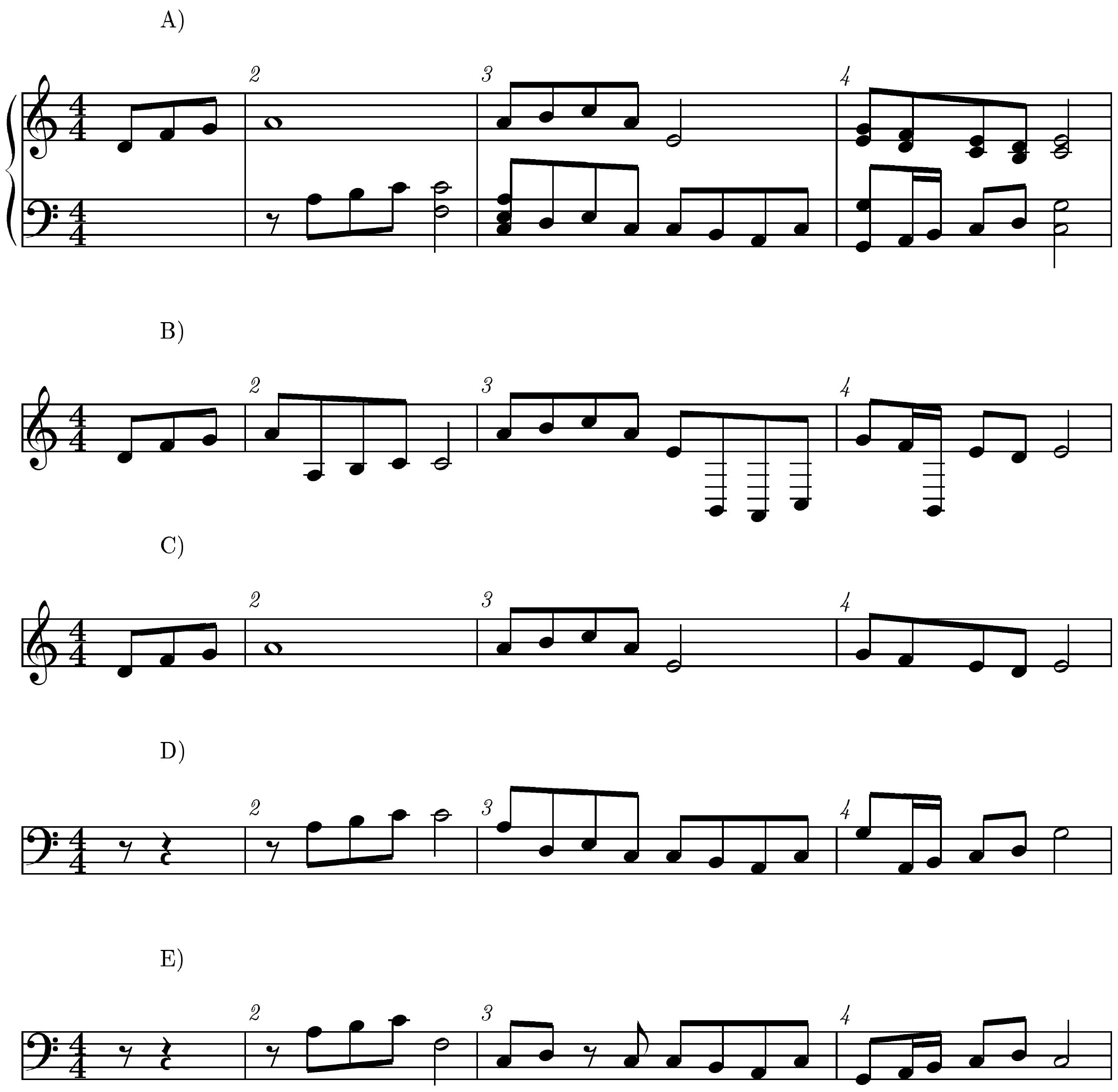


Figure 5.1: *Melody extraction techniques. The example fragment of music at A) has a melody consisting of the highest notes in the top staff. In B), with all-mono melody extraction, the melody is captured in addition to intervening notes. At C), the top-channel method is illustrated. In this case, the melody is perfectly extracted. At D) the entropy-channel method has determined that the lower part is more interesting and has selected the highest notes starting at any instant within that channel as the melody. At E) entropy-part has selected notes based on their proximity, and the part with the highest entropy was chosen as the melody.*

5.2 Our Techniques

We tried several approaches to automatic melody extraction that make use of music perception principles: the highest musical part is usually perceived to be the melody, and notes that are close together in pitch are usually considered to belong to the same musical part. As melodies are likely to be less repetitive or monotonous than accompanying parts, we tested the use of first-order predictive entropy as a means of predicting which part contains the melody. This calculation is in effect based on the probabilities in a simple Markov model with one state for each note. In highly repetitive tracks, such as some forms of accompaniment, each note is reliably predicted by its predecessor, giving a low entropy, whereas in more varied tracks the entropy is high.

We developed four techniques for extracting melodies from pieces of music, which we have named all-mono, entropy-channel, entropy-part, and top-channel. All the methods made a single pass through the note stream to select the melody notes, ignoring the note events from channel 10 as these are percussion events in standard MIDI files. In some circumstances, different methods generate identical melodies, most often when the music consists of a melody with a simple chordal accompaniment below it, with the chords occurring at the same time as melody notes. The four algorithms are described below. The effect of each algorithm is illustrated in Figure 5.1.

5.2.1 All-Mono

The all-mono technique combines all the note events from every channel into a single stream of events. At each instant in which there were notes starting, the highest-pitched of these notes was selected as a “melody” note.

This results in many overlapping notes, as a note that is sustained in one track covers the start of other notes. Extra notes are often included in the melody, as illustrated by Figure 5.1B. In this example, there are two channels in the original piece shown at A, each represented by a separate staff in the score. The melody consists of the highest notes in the upper staff. The all-mono technique has selected all the melody notes in addition to accompanying notes that occur whilst a melody note is sustained.

There is an advantage as well as disadvantages to the all-mono technique. As the melody is often the highest part in pitch, the majority of the melody will be selected. In the case where a melody is not the highest part, there is the possibility of still collecting some of the melody with the algorithm. One disadvantage of the technique is that it will rarely select the melody cleanly from a piece of music. Another is that it will not select the melody at all if it is lower in pitch than other parts.

For the purpose of evaluation in our experiment, note lengths were truncated so that the resulting melody was monophonic.

5.2.2 Top-Channel

The top-channel algorithm makes use of the structure of MIDI files (tracks and channels) by examining each channel separately. Each channel is processed by applying the all-mono algorithm to it to produce a melody for the channel. The channel melody with the highest average (mean) pitch is then chosen as the melody. Each note in the channel melody is given equal weighting in the calculation, regardless of duration.

The algorithm, illustrated in Figure 5.1C, works well for the example but in practice sometimes wrongly identifies a high accompanying part. Channel-based techniques such as top-channel also fail to select the entire melody if it moves from one part to another.

5.2.3 Entropy-Channel

The entropy-channel technique also makes use of the channel structure of the MIDI file. It applies the all-mono technique to each individual channel of the piece, then the channel with the highest first-order predictive entropy is selected as the melody.

Entropy measures the amount of “information” in a message by calculating the average number of bits needed to represent each symbol of the message. In our case the message is a sequence of notes. The entropy was calculated in the following manner: The pitch of each note of the melody was represented as a MIDI note number between 0 and 127 inclusive. The formula for the first-order predictive entropy [86] used was:

$$-\sum_i p(x_i|x_{i-1}) \log(p(x_i|x_{i-1})) \tag{5.1}$$

where each x_i is a note.

A sequence of repeated notes would have a lower entropy than a sequence with greater variation. When a sequence of notes is monotonous, the listener tends to focus on a part of the music that is more interesting, regardless of the relative pitch of the parts [51]. We suggest that entropy can be used to measure how interesting a musical channel or part is in order to select the melody of a piece of music. In the example the lower part had the greatest entropy and so was chosen.

5.2.4 Entropy-Part

The entropy-part technique uses timing and pitch proximity information to split the music into parts. The notes from each channel were analysed in turn and allocated to an appropriate part. If the current note occurred at the same time as those in the existing parts then a new part was created. Otherwise the note was allocated to the part that was the most similar in pitch, using the most recent note in each part to make that decision.

The part with the highest entropy is then chosen as the melody. The splitting algorithm compares the current note to previous notes in the piece. Unfortunately, if there are multiple notes occurring at precisely the same time, as happens with non-natural performance information, then the algorithm may allocate notes in unexpected ways. This is shown in the example in Figure 5.1E, where the part selected goes from middle C to F instead of back to middle C, because in the note stream the F occurred before the second middle C. If the simultaneous notes C and F were reversed in the note stream a rather different part would be produced. The rest in the third bar occurs because the E has been allocated to the part that contains the previous E at the start of the bar.

5.2.5 Discussion

In the example in Figure 5.1, the melody was correctly selected by the top-channel method. In general this is not the case, however, as our experiment reveals.

The methods described here form a foundation for future melody extraction research. All of them are fairly simple. It remains an open question as to whether significant improvements can be gained by more sophisticated algorithms applied to a wide variety of musical works. An improvement in part-splitting would be to group notes that occur at the same time and then allocate them to the appropriate part. Each method could make use of simple heuristics based on duration and other factors to improve their performance. Combining average pitch and entropy information is also likely to improve results. These four methods are, however, the basis of our first and only experiment in determining what users judge to be the best approach.

In our melody matching work we also make use of the melody extracted from each channel, thereby removing the need to select which channel is *the* melody. The name we have given to this approach is “all-channels”. If the all channels approach turns out to be the most effective, then there would be no need to use entropy or average pitch as part of the melody selection procedure.

5.3 Experiment

From our analysis of the needs of music databases users and from well-known results in music perception, we believe that each of these algorithms has potential as a method of extracting melodies from music. However, application of these methods to actual music shows that they can produce very different results. We applied the four algorithms to ten MIDI files, representing a range of styles and methods of organisation within a file. Each of our eight listeners was presented with the original MIDI file and the four automatically-extracted melodies for each of the ten files.[1]

The melodies generated for each piece of music were presented in random order. The listeners were able to play the pieces as often as they wished, and sometimes only played a small portion of the melodies if that was sufficient to make a decision. The extracted melodies were ranked from one to four according to how well they represented the melody of the original piece. The listeners were asked to consider the presence of extra notes and absence of melody notes in their evaluation, and were permitted to give melodies equal ranking.

5.3.1 Results

The sum of the ratings for each algorithm for each piece are shown in the table at Table 5.1. Low scores indicate a high rating. The range of possible scores is 8 to 32. As can be seen from the results, most algorithms performed well for some pieces and badly for others. The only algorithm to work consistently well was algorithm one (all-mono) —the most naive of the

[1]The MIDI files are available at http://www.mds.rmit.edu.au /~sandra/melexp.

	Algorithm				Best	Worst
	all-mono (am)	top-chan. (tc)	entropy-ch. (ec)	entropy-part (ep)	Alg	Alg
1	18, 0.70/0.13	30, 0.00/0.00	14, 1.00/1.00	17, 0.76/1.00	ec	tc
2	10, 1.00/1.00	24, 0.00/0.03	19, 0.16/0.47	21, 0.16/0.47	am	tc
3	9, 1.00/1.00	32, 0.00/0.00	20, 0.13/0.19	19, 0.13/0.19	am	tc
4	14, 1.00/1.00	14, 1.00/1.00	13, 1.00/1.00	32, 0.35/0.55	ec	ep
5	11, 1.00/1.00	22, 1.00/1.00	13, 1.00/1.00	13, 0.93/1.00	am	tc
6	16, 0.99/0.34	12, 1.00/1.00	16, 0.91/0.38	32, 0.00/0.00	tc	ep
7	14, 0.99/0.69	12, 1.00/1.00	26, 0.00/0.00	24, 0.00/0.00	tc	ec
8	22, 1.00/1.00	14, 1.00/1.00	16, 1.00/1.00	25, 0.43/0.93	tc	ep
9	13, 1.00/1.00	31, 0.11/0.59	21, 0.02/0.03	20, 0.02/0.03	am	tc
10	13, 1.00/1.00	14, 0.78/1.00	32, 0.22/0.25	19, 0.30/0.49	am	ec

Table 5.1: *Results of ranking melody extraction algorithms. The first number in each algorithm column is the sum of the rankings given for the melodies by the 10 evaluators. The second number is the proportion of notes in common with the melody selected as the best by the evaluators, expressed as a ratio of number of common notes over the number of notes in the best melody. The third number is the ratio of the number of common notes over the number of notes in the melody generated by that algorithm. For example, the most successful algorithm for piece number one was judged to be the entropy-channel algorithm. The number of notes it had in common with the all-mono algorithm divided by the number of notes in the entropy-channel algorithm melody equals 0.70. The number of common notes divided by the number of notes in the all-mono algorithm melody is 0.13.*

methods we considered. Interestingly, for pieces 5 and 8 the first three algorithms produced identical melodies but were ranked somewhat differently. For piece number 5 ("Scotland the Brave") it is our opinion that the entropy-part algorithm performed the best, as accompanying notes at the start and during the piece were removed. This difference was not detected by the evaluators, however.

Sometimes music is perceived differently if it is recognised. In this experiment, however, the algorithm rankings were similar regardless of whether or not the music was recognised. The music experience of the evaluators did not reveal any clear-cut trends in algorithm preference. The least skilled group (three people) and the most skilled group (three people) ranked algorithm one as the best whereas the middle group (two people) ranked algorithm three the best overall.

In conclusion, choosing the highest pitch notes starting at each instant (all-mono) usually best selects the melody of a piece of music, but garners extra notes. The other three methods can select a part which has no notes in common with the melody of the music; as our example shows, there can be almost no notes in common between the melodies generated by these algorithms. In some cases, this involves the selection of a part with very few notes in it, sometimes widely separated in time. In the case of top-channel, this may happen with a brief high accompaniment, whereas with entropy-channel or entropy part a sequence of seemingly random — therefore high entropy — notes may be selected.

It may be possible to improve the melody extraction process. The first algorithm often included notes that have a much lower or higher pitch than the majority of the notes. In some circumstances these could be removed. The difficulty is knowing whether these outlying notes are indeed extraneous, or the only notes from the melody that have been extracted from the piece. The part extraction algorithm could possibly be improved by considering all notes that start simultaneously as a group. We speculate that combining entropy with average pitch and other information may yield more reliable results when selecting the melody part. However, others who have attempted more sophistication in part splitting have also found it difficult to achieve reliability [87].

5.4 Summary

Several algorithms that extract melodies from polyphonic pieces of music were presented. Each one was based on aspects of simultaneous notes perception. The simplest of these, that selects the highest pitch note starting at any instant worked the best compared to the other more sophisticated algorithms when evaluated by listeners. None of the algorithms can reliably extract melodies from all types of music, but this is likely to be impossible. We speculate that better results could be achieved with the combination of the factors used in our experiment, namely pitch proximity, entropy, and pitch height, and possibly with more detailed analysis of the musical data. However, as the simplest algorithm worked the best in our experiments, it is not clear that significant gains could be achieved by more sophisticated methods when applied to such a wide variety of music.

Chapter 6

Melody Standardisation

The aim of melody standardisation is to produce a normalised form so that different performances of the same piece of music can be readily compared. By selecting the most important features of melodies for matching we also reduce the storage space required and the time taken to search for answers to a melody query. More importantly, we ensure that the most relevant answers to a query will be retrieved from the collection.

Some kinds of melodic information are more important than others for comparing melodies. It has been found that pitch is the most important for human comparisons, so a standardisation method should include pitch in its representation if the matching is to be successful. Extra features such as rhythm are also helpful. It is possible that the best standardisation technique may vary depending on which similarity measurement method is chosen. In this chapter we define various methods of melody standardisation and discuss their advantages and disadvantages.

There are many possible methods of melody standardisation. They can purely represent the pitch of successive notes or combine other features such as duration and stress. We discuss the possible pitch representations in detail below and after that the other features that can be used in the representations of melody. Most of the methods discussed in this chapter have been used by other researchers. In this thesis our experiments focus on the contour, directed modulo and exact interval methods discussed in Subsections 6.1.1, 6.1.7, and 6.1.5 respectively.

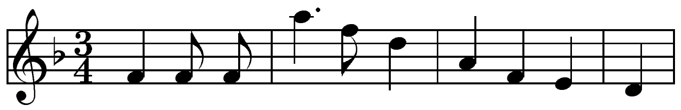


Figure 6.1: *Example melody fragment that contains a leap of more than an octave (from Domine Deus by Mozart, K427).*

6.1 Representing Pitch

When representing the pitch of each note of a melody, we can do so in several ways: absolute pitch, relative to the key, or relative to the previous note. The other main factor in representing the pitch is the precision of the pitch representation. A further variation on pitch representation is whether use is made of the fact that notes repeat every octave. Table 6.1 lists the various combinations and the terminology that we will use for the methods in our discussion. We discuss each of the methods in turn below.

6.1.1 Contour

A melody's contour represents a melody's shape in terms of pitch direction only. The contour representation of the melody fragment shown in Figure 6.1, using U for up, D for down and S for same pitch, is:

SSUDDDDDD

Contour representation has the advantage that singers usually get the contour of a melody right but usually don't sing the intervals accurately, an important consideration when queries are sung. A melody query would need to be quite long for relevant answers to be found, however. In particular, two melodies can be represented by the same contour string yet have no perceived similarity. For example, the two phrases of Twinkle Twinkle Little Star have the same contour (and rhythm) as those of the second movement of Haydn's Surprise Symphony, yet the melodies are quite different.

Method Name	Symbols	Relativity	Direction	Comments
contour	3	previous note	yes	
extended contour	5	previous note	yes	
Cn	n	previous note	yes	Downie's nomenclature for various classifications, based on frequency of occurrences, so high precision for small intervals and low precision for large ones.
absolute pitch	128	none	yes	Based on MIDI note numbers
exact interval	255	previous note	yes	Difference between MIDI note numbers
modulo interval	12	previous note	none	All intervals converted to the equivalent interval less than an octave
directed modulo	23	previous note	yes	As above, but interval direction is kept, eg. "up two semitones" is different to "down two semitones"
key-relative	12	key note	none	Requires key information
directed key-relative	23	key note and previous note	yes	Requires key information
base-tone-relative		a note within the bar	yes	Kosugi et al.'s method [76]
scale-independent	7	key note	none	Allows tonal matches in different scales, such as minor to major
directed scale-independent	15	key note	yes	Tonal matches in different scales preserving contour
fuzzy extended contour		overlap for intervals that could be considered large or small		
secondary contours	3	current note−2	yes	Used in addition to contour to increase precision

Table 6.1: *Different methods of melody standardisation.*

Interval	Representation			
	C7		C15	
	up	down	up	down
0	a	a	a	a
1	C	c	B	b
2	B	b	C	c
3	C	c	D	d
4	D	d	E	e
5	D	d	F	f
6	D	d	G	g
7	D	d	G	g
8	D	d	G	g
9	D	d	G	g
10	D	d	G	g
11	D	d	G	g
12	D	d	G	g

Table 6.2: Downie's C7 and C15 melody classifications

6.1.2 Extended Contour

Extended contour is a more fine-grained approach to melody representation. In this version of extended contour, we distinguish between large and small intervals with a different symbol. For example, we could use U for a large interval upwards, u for a small one, and similarly use D and d for large and small downward intervals respectively. A decision needs to be made regarding the classification of intervals as large and small. It is clear that step intervals of one or two semitones are small and intervals that are greater than five semitones are large. If we use the musical concept of steps and leaps, then all intervals of three or more semitones would be classed as large intervals. If we used the entropy-maximising classification approach of Downie (discussed below and elsewhere [41]), then the same decision would be reached, as intervals of one or two semitones are the most frequently occurring intervals in melodies. In this case, our example melody would be represented as:

SSUDDDDdd

The advantage of this technique is that it still allows for inaccurate singing, but would be more discriminating when searching a database of melodies. When considering pitch errors of singers, however, there may still be some mis-classification of intervals as singers often have an error of up to one semitone [83].

6.1.3 Cn Classifications

Downie [41] produced a set of representations of melodies based on pitch intervals only. The name given to each representation was the letter C followed by the number of unique symbols used in the representation. Where there was no restriction on the number of symbols, CU was used. He performed informetric analyses of a collection of 9354 folk-songs and based his classifications on the frequency of each interval. The main classes analysed were C3 (contour), C7, C15, and CU. The representation of the intervals is shown in Table 6.2.

C7 is interesting in that intervals of one and three semitones are represented by the same symbol. Zero and two semitone intervals are each uniquely represented and all intervals that are at least four semitones are represented by the same symbol.

6.1.4 Absolute Pitch

Absolute pitch standardisation represents the exact pitch of each note, rather than the relationship between notes. Our example melody may be represented using absolute pitch standardisation by using the MIDI note numbers:

65 65 65 81 77 74 69 65 64 62

As another example, the song "Mary Had a Little Lamb" is encoded as shown in Figure 2.2 when using common music notation. Using numbers to represent the pitch (for example, MIDI note numbers) we could represent it as:

64 62 60 62 64 64 64

It may not be very useful to represent the data as an absolute pitch, since the same melody can be played or sung at different pitches. When this happens there would be very few symbols in common. There are some published techniques that use absolute pitches with some success, however [99, 35, 23].

6.1.5 Exact Interval

In the exact interval representation the number of semitones between successive notes is used to represent the melody. For example:

0 0 16 −4 −3 −5 −4 −1 −2.

represents the example melodic fragment shown in Figure 6.1.

Exact interval representation has the advantage that it is one of the most precise ways of representing the pitch of a melody. As such it is more likely to contribute to high-precision matching of melodies.

All three methods have the problem that an error in a pitch can result in two consecutive errors in a matched melody when using string-based pattern matching techniques. For example, if a query melody contained a G instead of an F in bar 3, its representation would be:

0 0 16 −4 −3 −5 −2 −3 −2.

which has two adjacent symbols that differ from the original melody, despite only one note being different.

6.1.6 Modulo Interval

Another possible method of representing the melody is to store the relative pitches reduced to the scope of one octave, which we refer to as "modulo-12 intervals". This may be of use from a musicological perspective, or when looking at the harmony of a musical work, but loses information about the aspect of melody that people remember the best — the contour. If used, it would need to be combined with a search on contour before presenting answers to the user.

6.1.7 Directed Modulo

Directed modulo-12 intervals retain the direction information but reduce any intervals greater than an octave to the harmonically similar interval that is no more than an octave in size. This method has 25 distinct symbols. The Mozart melody fragment would be represented as:

0 0 4 -4 -3 -5 -4 -1 -2.

Exact interval representation provides for more accurate matching, however the directed modulo-12 interval method allows a smaller representation.

6.1.8 Key-Relative

The standardisation method that we have dubbed key-relative represents all notes relative to the key of the melody. In this version, the number zero represents the key or tonic note, and we represent notes by the number of semitones distance from the tonic note. This representation is equivalent to remembering the letter names of notes. The example melody extract would be encoded as:

3 3 3 7 3 0 7 3 2 0

since the melody is in D minor. If encoded in the related key F major it would be represented as:

0 0 0 4 0 9 4 0 11 9

This highlights one of the difficulties of this approach, namely, that the numbers used depend on the key note selected. In terms of melody matching, the user presenting a query is unlikely to indicate which key the melody is in. Since this is the case, we would need to either determine this key and risk missing matches that are identical but shifted to a different part of the key, or ensure that all possible matches are examined regardless of key.

This particular key-relative representation has further problems in that it doesn't preserve contour information. Directed key-relative standardisation, discussed below, retains contour information.

6.1.9 Directed Key-Relative

In this representation the direction of the note transition is recorded in addition to the pitch relative to the key note. This can be done by using positive numbers to represent going up to the note and negative numbers when the melody moves down. It is also necessary to encode the fact that sometimes the pitch remains the same. To demonstrate this in a consistent manner with the examples above, we use an extra symbol "." to represent a repeated note and use the number of semitones from the key note to represent all other notes. In addition we assume that we can associate a sign with the number zero. The example melody fragment would then look like the following:

3 . . 7 −3 −0 −7 −3 −2 −0

This representation captures both contour and information about the individual pitches. It still has the problem that a melody fragment that is represented in a different portion of the scale may not be located, but, unlike the interval-based representations, a single incorrect note only causes a single symbol to differ.

6.1.10 Base-Tone-Relative

Kosugi et al. [76] proposed storing melody fragments as pitch transitions relative to a "base tone". As each bar of notes was stored separately, they selected a pitch within the bar as the base tone and all other notes were represented relative to it. A candidate for the base tone could be the most frequently occurring pitch in the bar, or possibly the first pitch in the bar. In their representation rhythm was also a represented. The base-tone-relative representation of the first two bars of "Mary Had a Little Lamb" when stored relative to the first pitch would be:

$0\ -2\ -4\ -2\ 0\ 0\ 0$

and relative to the most frequent pitch in the bar:

$2\ 0\ -2\ 0\ |\ 0\ 0\ 0$

where | represents a bar line, indicating where a change in base tone has occurred.

6.1.11 Scale Independent

For independence of scale, a different approach could be used for representing the melody. Sometimes we may wish to match the same melody without considering whether it is a major or minor key. In this case we could represent the note's position in the scale. "Mary Had a Little Lamb"'s first phrase would be represented thus:

3 2 1 2 3 3 3

where 1 is the first, or key note of the scale. This would also match "Mary Had a Little Lamb" played in a minor key, where E is replaced with E♭. The interval representation in this case would be:

$-1\ -2\ 2\ 1\ 0\ 0$

As with the other representations that are relative to the tonic or key note, a decision needs to be made about how to represent pitches that exceed the scale. In this case, we apply the same technique as the key-relative technique shown above, namely, just represent the position in a single scale octave of seven unique notes. For example, the example melody would be represented as:

3 3 3 5 3 1 5 3 2 1

A scale-independent technique was used by Schaffrath [116] and others to allow matching of folk music melodies regardless of the scale used.

6.1.12 Directed Scale Independent

Directed Scale Independent standardisation adds the contour information to the representation. This can be done in the same way as for the Directed Key-Relative technique. That is, when the melody rises to a note, the note is represented as positive; when the melody falls, the note is represented as negative. In this case, repeated notes can be represented by zero. The example melody would be represented as:

$3\ 0\ 0\ 5\ {-3}\ {-1}\ {-5}\ {-3}\ {-2}\ {-1}$

In this way it is fairly clear which notes follow the starting note. The only time the pitch is unclear is in the case of leaps of greater than one octave.

6.1.13 Fuzzy Extended Contour

This technique was first reported by Lemström et al [78]. It involves allowing an overlap in the range of intervals that are classed as large or small. This ambiguity is used in the matching process, so that intervals in the range will be accepted as a match against other intervals in the same direction regardless of whether they are large or small, whereas an interval classed as small can only match other small or ambiguous intervals. The same case applies for large intervals.

6.1.14 Multiple Contours

De Roure and Blackburn [108] reported the use of secondary contours in their implementation. This consists of comparing each note, not with the previous note, but with the one before that.

Using a combination of a contour and a secondary contour index gave greater precision without needing to decide how to divide up extended contour representations.

6.1.15 Other Pitch-based Methods

Some methods try to capture both key and pitch information. For example, the note C♯, at least with modern tuning, is the same as D♭, however, in the musical context of keys they are different notes. Hewlett developed a base-40 system of pitch representation that allocates a different number to each possible modification of a note letter (discussed by Selfridge-Field [120]). C♭♭ and C♯♯ are included as well as the single and double sharps and flats for all other notes.

Our approach was to combine modulo-12 with direction, using a representation that has +12 for a leap up of an octave and −12 for a leap down. Any intervals greater than one octave would be mapped to the corresponding interval within an octave. For example, a leap up of 17 semitones from A to D would be replaced with a leap up of 5 semitones from A to D.

6.2 Other Features

Other features that are sometimes used for melody matching include rhythm and stress. Rhythm can be encoded in several ways: as an exact duration, a contour, or in a more detailed relative manner.

Exact duration holds many difficulties for melody matching as it is quite likely that a presented query and set of answers differ in their tempo or speed. Ideally, matching should consider the relative durations of notes.

Rhythm contour merely states whether the current note is longer than, shorter than, or the same duration as the previous note. This information is not always as straightforward as it may seem. If information derived from an actual human performance is used, differences in duration of notes tends to vary, even for notes that are intended to be the same. It may be necessary to have a cut-off threshold or similar approach to handle this case. Kageyama et al. [72] had a simple approach for rhythm: if the current note was less than double, or more than half of the previous one in duration, then it was encoded as the same duration. All others were classed as different in duration. This was not expressed in the encoding of the melody itself but in the matching process.

Other relative duration approaches that have been tried include examining the ratio of the durations of successive notes [79] and performing a kind of normalisation of durations by replacing a note of two beats with two notes of one beat, so that the weight of the match is increased for longer duration notes [97] when matching.

A feature that is related to rhythm is the encoding of rests. A separate symbol can be used to represent a rest in a melody. This can be useful for some rhythmic tunes and also for delineating the start and end of melodic phrases. Rests can be encoded in a similar manner to notes, however they do not have a pitch and cannot be compared in pitch to the previous note. In our experiments reported later on the use of rests in matching, it usually reduced the quality of answers. However, the minimum size of rests may have been too small for the method to have been useful.

Stress is another feature that is sometimes encoded. It usually represents the note's position within a bar. If a note falls on the first beat of the bar, it has a strong stress. In 4/4 time, the third beat of the bar has a medium stress. Similarly, in 6/8 time the fourth beat of the bar has a medium stress. All other notes have a weak stress. In playing music, musicians conform to this pattern of stresses to emphasise the rhythm of the music, making the first note of the bar slightly louder than the others. Sometimes the dynamics of a piece of music change this pattern slightly, particularly with the introduction of syncopation, that is, accents at normally unstressed positions within the bar, and also in certain styles of music where syncopated rhythms are typical. The three or possibly more levels of stress can be encoded for each note in a melody. On their own they are fairly meaningless, but could be used in combination with other features, as discussed in the next section.

6.3 Combining Features

For greater precision, the combination of pitch and rhythm can be used. As some notes are more important in matching than others due to the stress on the notes, a stress component could also be included.

In its simplest form, pitch and rhythm encoding can be achieved by having an ordered pair for each note. For example, the phrase from "Mary Had a Little Lamb" could be encoded as:

(64,1) (62,1) (60,1) (62,1) (64,1) (64,1) (64,2) if exact pitch and duration (in beats) were used. Similarly if stress were also encoded, a triple would represent each note:

(64,1,2) (62,1,0) (60,1,1) (62,1,0) (64,1,2) (64,1,0) (64,2,1)

In this case, we have encoded the standard text-book relative stresses for notes within a bar in 4/4 time: strong, weak, medium, weak.

The above methods may seem to be two and three dimensional representations, but, depending on the level of detail required, they could easily be encoded for matching into a single symbol. For example, the use of modulo-12 intervals with direction and rhythm contour can be encoded using three ranges of ASCII characters: the lower case could be used for intervals that have a longer duration, upper case can represent the same duration as the previous note, and the non-alphabetic printable characters can represent shorter duration notes.

6.4 Summary

It has already been discovered that pitch and especially the contour of melody is the most important feature for humans when comparing melodies. Rhythm alone is not sufficiently discriminating, for example, consider the melodies of “Twinkle Twinkle Little Star”, “Old MacDonald” and “Mary Had a Little Lamb”: all three commence with identical rhythms. It is possible that pitch contour and rhythm may be a useful combination, especially considering the abilities of the average person in presenting a query. For more refined matching the exact intervals between notes is important, with some allowance for the less important notes, that is, the unstressed ones.

While it seems to be important to include not only contour information but more precise pitch information, the best approach to use is not clear. The transition-based methods such as extended contour, secondary contours, and intervals all have the disadvantage of possibly having two adjacent symbols incorrect when a single note is different between two melodies that are being compared. However, all these methods are flexible in that melodies can be easily compared without needing to worry about transposing melodies to the same key in some way. Absolute pitch representation requires comparisons in all keys and octaves [34], or a compromise in the precision of the match [99]. Methods that are relative to the key, such as the scale-independent and key-relative methods described above, do not scale well with pieces containing modulations to different keys.

Stress may be important in removing answers that are less likely to be relevant. In particular, some answers may have the same pitch and rhythm contour, but may be perceived as unrelated due to being shifted to commence at a different position within a bar, or in a different time signature. However, it is probably unnecessary to include stress for melody matching for the current music collections available for research. In our experimental work we focus on three standardisation techniques, contour, directed modulo-12, and exact interval. These are tested in conjunction with similarity measurement techniques in Chapter 9.

Chapter 7

Musical Data

Implementers of IR systems sometimes analyse the data in order to predict the most appropriate indexes to build [145]. The extent of the analysis can be merely determining the alphabet size and the number of distinct terms to be indexed, or may involve a more detailed analysis of the data's properties, including its distribution. In this chapter we discuss the difference between musical data and other types of IR data, describe the musical data that we use for our experiments, and examine the term distribution of that data.

7.1 Informetrics

The study of the statistical properties of information is sometimes known as informetrics. One of the aims in the use of informetrics for information retrieval is to suggest likely techniques that would be applicable to a collection of data. Wolfram [145] lists some types of statistics about a collection that can be used to inform IR design. These include term distribution, term co-occurrence, and database growth. In addition to the above, other features that can describe a collection of data used for retrieval include the type of term, alphabet size, the frequency of each character in the alphabet, and the range of document lengths within the collection.

Knowing the term frequency distribution can help determine the best structure for an index, for example, to determine whether stopping, that is, the removal of the most frequent terms, is applicable. Informetrics has mainly been applied to improve the performance of search techniques, and to determine the size of indexes [145].

The query resolution techniques that are most appropriate to an information retrieval system depend on the nature of the data, the queries, and relevance. Relevance is harder to quantify, but for music, genomic, and some types of text searching, such as spell-checking and name searches it may be possible to characterise it to some extent.

Once reduced to sequences of characters, melodic data can be described by various parameters such as alphabet size and distribution, allowing comparison with other types of data collections used in retrieval such as text and genomic data collections. Like melodic data in this form, the textual representation of genomic data is actually a reduction that excludes some information.

If musical data resembles other forms of data for which information retrieval techniques are already successful, then the same techniques could be applied to music retrieval. However, a retrieval technique that is effective on one collection may not be as effective on even another collection of the same type of data [152], so experiments to determine effectiveness are still necessary.

7.1.1 Textual Data

For text, there are two main approaches to searching: searching for strings in the manner of a pattern matcher and searching for documents that answer an information need. Various types of term have been explored for use in string matching or document retrieval, the most common being characters, n-grams of characters and words. Sometimes, sequences of words are also used, typically of length two. The alphabet size for characters is anywhere from 27 to approximately 100 for English text, depending on whether case, punctuation, and extra symbols are used. The frequency of each character can vary from negligible to over 17% (for spaces) in an alphabet of 94 characters [143].

If the terms are words, then the number of distinct terms rises to tens or hundreds of thousands or more, depending on how a word is defined [143]. Whether characters or words are used as the basic unit, there are different frequencies for each different unit. There are also different frequencies with which pairs of characters or words occur adjacent to each other. In document retrieval, search engine designers make use of the fact that words in a query are likely to occur in relevant documents, but less likely to occur in irrelevant documents. Words that occur in most documents are not useful for discriminating between relevant and irrelevant documents, so they are either not used or contribute in a very minor way to the ranking process.

A simple model that approximates the distribution of words in text is Zipf's law [149], which states that the rank multiplied by the frequency is approximately constant, that is, the distribution of words in text is approximately hyperbolic. There have been better models proposed, such as the Zipf-Mandelbrot function [109] which improves the fit for terms at each end of the distribution. For n-grams of text, however, the models are a poor fit. Using a model based on the random division of the probabilities of each character that occurs works somewhat better but the fit is still fairly poor [143].

7.1.2 Genomic Data

Genomic data is typically expressed as a sequence of symbols representing nucleotides or amino acids in strands of DNA. There is no unit equivalent to a word as such, but each nucleotide or amino acid is represented by a letter. There are only four letters in the nucleotide alphabet: C, G, T and A. In addition, there are eleven standard wildcard characters that are used to represent parts of a DNA sequence that are not fully known (Described by Williams [140]). Alternatively, amino-acid sequences can be used, each of which is made up of three adjacent nucleotide symbols, in which case there is an alphabet size of 20. The frequency distribution of both these representations is fairly flat. Indeed, some researchers have concluded that protein is incompressible [98]. However, Mantegna et al. discovered that the non-coding part of DNA follows the Zipfian distribution better than coding DNA (discussed by Rousseau [110]). Indexing has only recently been used successfully for genomic data [142] and n-grams are the chosen index terms. The small alphabet size would result in high repetition within genomic sequences, particularly with short n-grams.

7.1.3 Musical Data

Musical data can be represented in many ways. Although some researchers choose to retain all notes for matching in their approach to music retrieval, it is likely to be valuable to extract melodic data from the source.

Melodies can be expressed as a sequence of symbols representing the pitch of each note or the pitch distance (interval) between notes. Alternatively, notes or intervals can be represented as ordered pairs consisting of pitch and rhythm. In some research [7, 99], stress is also included in the note representation. The pitches themselves can be represented in many ways: exact pitches, relative pitches or intervals, pitches relative to the key of the melody, or a reduced-precision representation such as melody contour instead of exact intervals. The alphabet size varies with the representation method used. If melody contour is used, the alphabet consists of just three symbols representing whether the melody goes up, goes down, or stays the same in pitch. If exact interval standardisation is used as defined in Chapter 6 for an automatically extracted set of melodies, the interval representation alone may include a range of up to 255 values. Rhythm information would increase this further. As with genomic data, there is a high likelihood of repetition within a melody, as repetition is a common technique in musical composition.

As an extracted, standardised melody can be represented as a text string, suggesting that text-based techniques may work well for melody retrieval, but the best approach needs to be determined via experiment.

7.2 The Music Collection

There are many collections of musical data. Most of these are monophonic, such as the Essen collection of folk songs and Parson's collection of themes originally published in book form.

The main source of polyphonic musical data is the Internet's accumulation of MIDI files, contributed by many people. Unlike the data collections prepared by musicologists, the Internet MIDI data is prepared in an uncontrolled manner. The files do not always fully conform to the standard format, making it difficult to assume that certain features will be present in each MIDI file.

Our focus in this research has been the querying of polyphonic music for melodies using musical note information as opposed to audio information. Therefore we chose to use the MIDI data available to us as a basis for our experiments. In this chapter we examine the data to provide a picture of the characteristics of musical performance data. We apply one of our melody extraction techniques to the data and show the frequency distributions of substrings of that data in the hope that it provides an insight into how melody retrieval should be implemented.

7.3 A Survey of Musical Data Analyses

Much has been published on the nature of musical data. Zipf [149], argued that music follows the same distribution as text and showed this behaviour in reference to one or two specific musical works. Dowling [36] counted the frequency of different interval sizes in a collection of 80 Appalachian songs and discovered that the smallest intervals were the most common. Zaripov [148] looked at the frequency distribution of parts of melodies, concentrating on a collection of Russian folk melodies. In addition, there has been work in the musicology [73] and fractal [121, 46] fields regarding entropy and music.

Bainbridge [5] used gzip to estimate the repetitive nature of musical data. He concluded that the entropy is 1.6 bits per note, and that 18 notes would be required to uniquely identify a melody in their collection of about 99,000 MIDI files if pitch was the only information used. The other statistics determined about the collection are that on average files contained 7.4 channels, each containing an average of 700 notes. The average duration was about 5 minutes.

Downie [42] analysed the folk-song database that was also used by McNab [93] and concluded that there were no likely candidates for removal as stop-words.

7.4 An Analysis of Melodies Extracted from MIDI Files

We ran a series of experiments to determine the distribution of the n-grams for a body of musical works. The works initially used were MIDI files downloaded from the Internet. Several sub-collections were used:

- Approximately 500 MIDI files of a mixed variety gathered from several WWW locations;
- Approximately 15 000 files from an ftp site that collected all MIDI files posted to a particular newsgroup; and
- A subset of the 15 000 MIDI files consisting of all those MIDI files that contained only one track, with this track only containing one channel. This ensured that all notes were likely to have been played on one instrument only and so the melody extraction process would produce one melody per MIDI file. There were 209 files in this collection.

The melodies were extracted from each MIDI file as a contour string, a modulo-12 interval string or an exact interval string. Where there was more than one track in the MIDI file, a melody

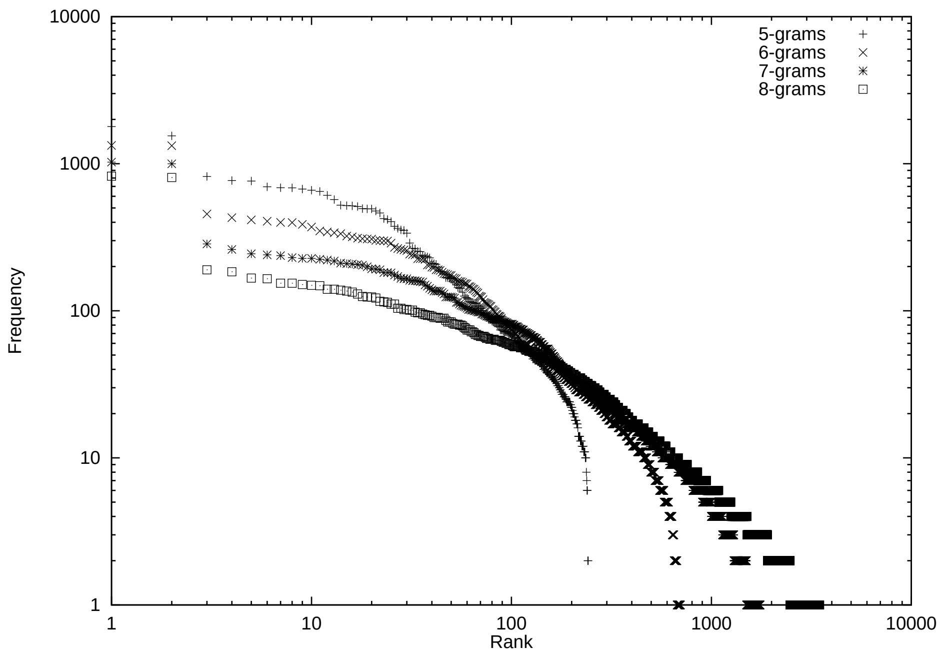


Figure 7.1: *Graph of the n-gram frequency distribution of melody contour strings extracted from 209 single-track single-channel MIDI files.*

string was extracted for each track and treated as if it were a separate melody. The collection of melody strings was then processed to produce a rank-frequency distribution of the n-grams, for values of n between 1 and 13. The results were then examined and graphed. Some of the graphs obtained from the data are shown in Figures 7.1 and 7.2.

To determine how many melodies contain each n-gram the melody strings were re-processed to produce another rank-frequency distribution that ignored the number of times n-grams occur within a piece of music. For example, an n-gram that occurs in all 2 697 tracks would get a frequency score of 2 697, regardless of the number of occurrences within each melody string.

We then determined the number of tracks expected to be retrieved by an average query by calculating the median point when n-grams were ranked and observing the number of tracks in which the n-gram at the median point occurs. This result was graphed for n-gram sizes from 1 to 12.

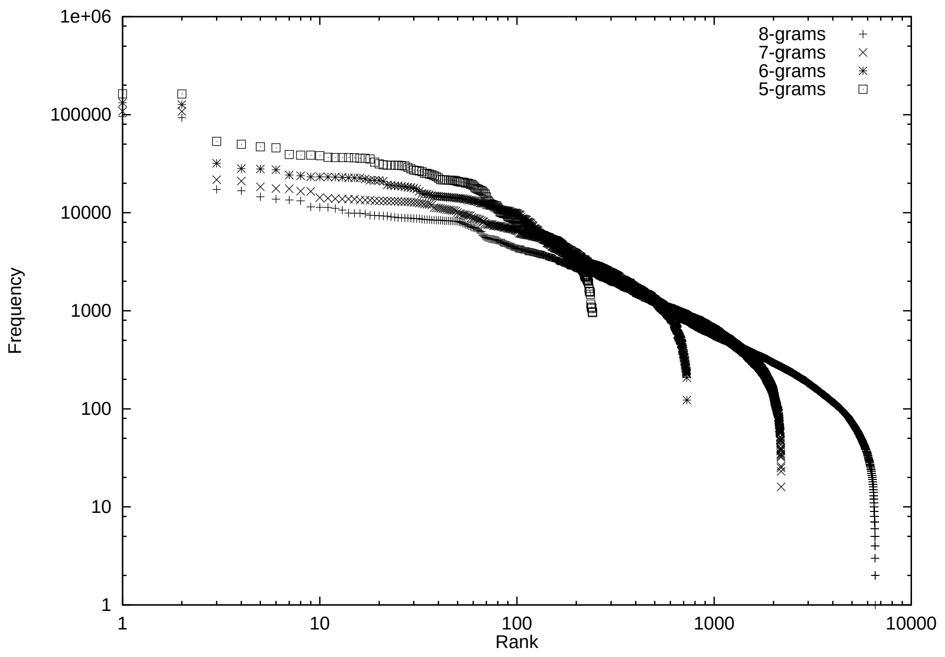


Figure 7.2: *Graph of the n-gram frequency distribution of melody contour strings extracted from 11270 MIDI file tracks.*

7.4.1 Results

Figures 7.1 and 7.2 show that the data follows a distribution similar in shape to that of text n-grams, as shown in Witten's paper on text models [143]. The distribution is quite far removed from the kind described by Zipf's function. The shape remains similar for different choices of n. The distribution of n-grams using other standardisation methods (not shown) were similar in shape, but the individual n-grams were less frequent. This indicates that melodic information behaves in a similar way to text and similar techniques could be used to optimise retrieval of it. Figure 7.3 shows the distribution of the contour n-grams found in a collection of 500 MIDI files. These files contained a total of 2 697 tracks that were each processed individually to extract their melody n-grams. Many of the n-grams are very common, occurring in over one third of the tracks. These would be poor at providing answers to queries.

Figure 7.4 shows the relationship between n-gram size and the expected number of tracks to be retrieved for a database size of 2 697 and contour standardisation. To retrieve less than 10

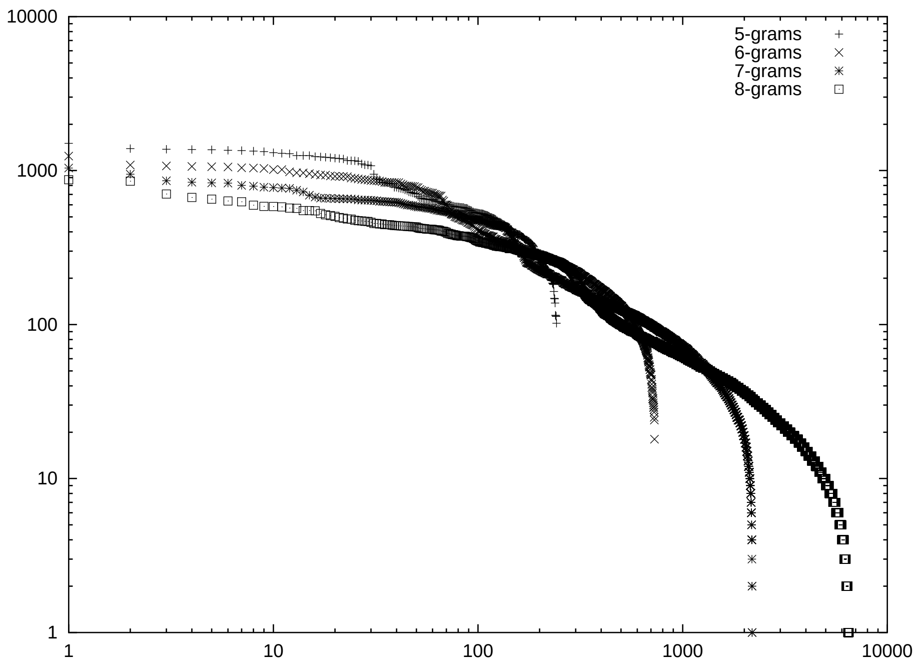


Figure 7.3: *Graph showing the number of tracks (out of a total of 2 697 tracks) containing each contour n-gram for the collection of about 500 files.*

tracks from among 2 697 requires a query consisting of 12 notes on average. A query of 6 notes would result in about 330 tracks with an exact match being returned. In the collection of nearly 15 000 files (not shown), about 1 220 tracks out of about 65 000 tracks would be retrieved by a query of the same length. Clearly, a longer string of notes would be required in queries, or greater precision in the search for melodies.

In an examination of the n-grams of the collection of 10,466 pieces used in our other experiments (directed modulo standardisation of melodies extracted using the all-mono method), the most frequently occurring n-gram was a series of repeated notes. For $n = 4$ this n-gram occurred 589,700 times, in 7,960 pieces. The next most frequent 4-gram occurred 23,606 times in 5,045 pieces. Of the 279,841 possible 4-grams using this type of melody standardisation, 181,404 occurred in the collection. For $n = 5$ the repeated-note n-gram occurred 254,949 times in 4,903 pieces, with the next most frequent n-gram occurring 14,742 times in 1,514 pieces. The collection contained 1,325,963 of the possible 9,765,625 n-grams. Only when $n < 3$ are all n-grams present

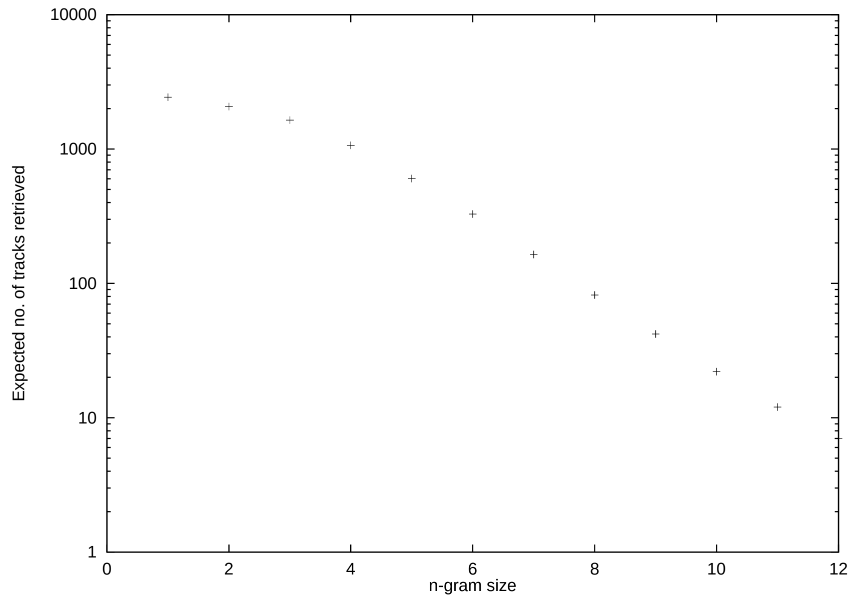


Figure 7.4: *Graph showing the number of tracks expected to be retrieved given the contour n-gram size for the database containing 2 697 tracks.*

in the collection. Figure 7.5 shows a rank-frequency plot of the number of melodies containing each n-gram.

7.4.2 Discussion

The data analysis above suggests that using melody contour as the only basis for preparing answers to a query would be insufficient as databases grow larger. Despite this, it may still be useful to index contour n-grams for answering queries by inaccurate singers, particularly if in addition a secondary contour is used [108]. Contour representation's usefulness for music queries would depend on the typical length of queries, however. Contour n-grams can be compressed to a greater extent than an exact interval n-gram but their use would reduce the precision of answers to queries.

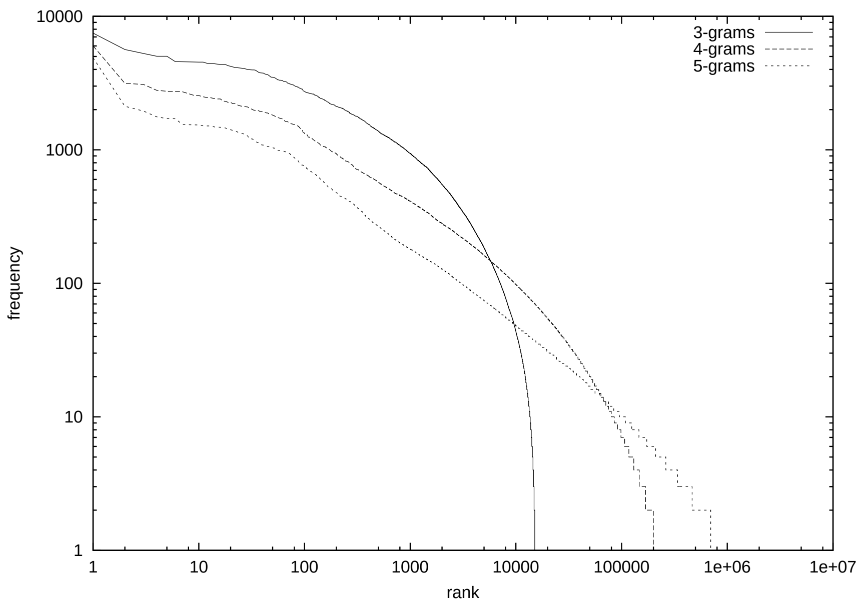


Figure 7.5: *Graph showing the number of pieces (out of a total of 10,466) containing each directed modulo n-gram for the all-mono database.*

Stopping is a technique that is often used with the indexing of terms for retrieval. It involves omitting the most frequently used terms from the index in order to reduce the index size. This can usually be done without affecting retrieval effectiveness because the most frequent terms are not useful in discriminating between documents. The data analysis above shows that some terms are very frequent, particularly for contour representation. This suggests that stopping would be a useful tool for reducing the size of an n-gram index. For richer representations of melody than contour the frequency of terms is not so large, however, there is one n-gram that is very frequent in this collection regardless of the standardisation method used, the one that represents a sequence of notes of the same pitch.

Since a large number of answers could be retrieved from a music database which would be time-consuming to check, ranking of results will be essential. Once a set of melodies is retrieved using an n-gram index, they could be examined more closely with a fine search. The index terms could consist of n-grams of some kind, where the individual symbols represent note intervals. This n-gram-based coarse-and-fine search approach has been applied to text for spell-checking and personal name searching [151], as well as to genomic data [142].

7.5 Summary

Musical data, once in the form of a standardised melody, resembles text in many ways, including the frequency distribution of n-grams. Some of the techniques used for text, such as n-gram indexes and stopping could be applied to music in order to better search for matches to a user's melody queries. A melody contour query, however, would need to be fairly long in order to retrieve a small number of answers as databases grow larger.

Despite some similarities between textual, genomic, and melodic data it is not clear that methods that work for one data type will work for the others. For example, ranking genomic data by using a TF-IDF similarity measure (explained in Chapter 8) has been shown to be less effective than a simple count of distinct common n-grams [140], whereas the reverse is true for text retrieval [113]. Nevertheless, our results reveal a similarity to the distribution associated with text n-grams and suggest a likely query length needed for good answers with the shown melodic representations.

Chapter 8

Similarity Measurement

There are many candidate methods for measuring the similarity of melodies. Some were originally applied to fields such as text retrieval and genomic database searching. Ranking of text documents is usually carried out by calculating a similarity measure based on the frequency of words within the document and in the text database as a whole. It is not clear what would be considered a word or term in a melodic document and whether the same techniques would be useful for melody databases. If n-grams are used as melodic words, their distribution is somewhat different to that of words in text, and techniques that work best for melodies are not the same as those for text, as shown by our experimental results in chapter 9.

In genomic database searching, protein sequences are compared using a dynamic programming technique known as local alignment. More recently, n-gram indexing has been applied as part of a two stage approach to matching [151], with local alignment being the basis of the second stage. As discussed in chapter 7, and shown by our experiments in chapter 9, this approach may be applicable to searching music databases.

In this chapter, we discuss the two main approaches that are the focus of our work, namely, dynamic programming and n-grams. Experiments that test these techniques are described in chapter 9.

8.1 Adapting Dynamic Programming to Music Matching

If a melody standardisation technique is used that represents the melody as a sequence of symbols, that is, a string, then we can apply string matching techniques to compare melodies.

One established way of comparing strings is to use edit distances. This family of string matching techniques have been widely applied in related applications including genomics and phonetic name matching [142, 151].

There are three main types of edit distance matching techniques that we consider here: longest common subsequence, local alignment, and longest common substring. Another approach that could be used is global alignment. However, this is less likely to be useful when comparing a fragment of a melody to a full piece of music as it measures global similarity.

8.1.1 Longest Common Subsequence

With the *longest common subsequence* (LCS) technique, the query is matched against pieces with no penalty for gaps of any size between the matching symbols. LCS has potential for this task because melody extraction often yields additional non-melody notes. In essence, LCS counts how many symbols occur in sequence in both strings. The similarity score is the largest such common subsequence. For example, consider the sequence of intervals below for the first phrase of Beethoven's Fifth Symphony, represented as the number of semitones, with positive numbers meaning an increase in pitch.

$0\ 0\ {-4}\ 2\ 0\ 0\ {-3}$

Suppose this is to be matched against the start of "Au Clair de la Lune", represented as:

$0\ 0\ 2\ 2\ {-2}$

By visual inspection it can be seen that they both contain the subsequence:

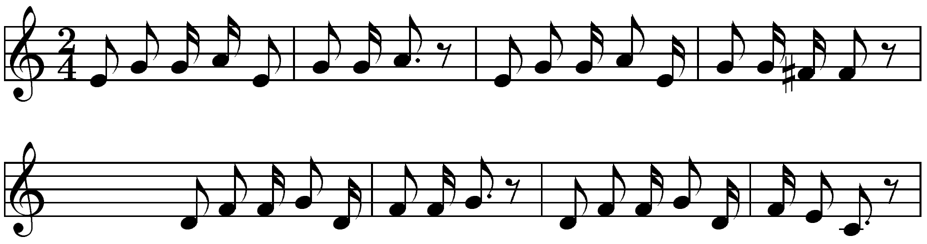
$0\ 0\ 2$

Consider what would happen in the case where one note of Beethoven's Fifth was incorrectly played. For example, consider the representation below:

$0\ 0\ {-4}\ 2\ 0$ -2 ${-1}$

Immediately there are two symbols that are different even though the melodies only differ by one note. A perfect score for comparison of the original melody fragment to itself would have been 7 (that is, seven common symbols in sequence). With a single wrong note, the score reduces

A) "Année Passée"



B) "Rum and Coca Cola"

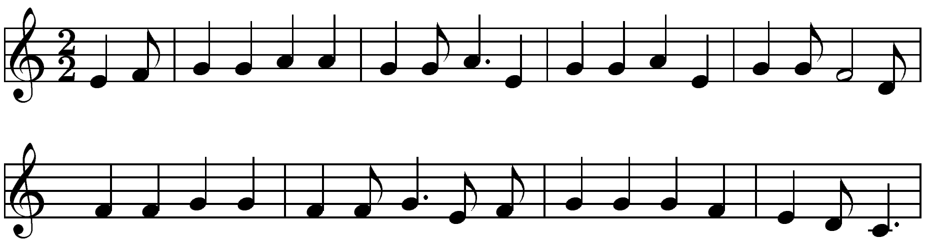


Figure 8.1: *Melody extracts of the songs "Année Passée" and "Rum and Coca Cola".*

to 5. A melody of the same length with one symbol altered has only the note occurring before the changed symbol in common with the melody. All notes following it are shifted in key. For example, the melody string below has the first four notes in common. This fragment could also receive a score of 5.

$0\ 0\ {-4}\ 0\ 0\ 0\ {-3}$

In this case, the relationship between the notes after the mismatch is totally different to that of the original melody fragment.

The above melodic fragments do not have much in common other than that the starting note is played evenly three times. In general, pieces with significant similarity would have more notes in common and therefore would have longer subsequences in common. To see what happens with a longer example, consider the two songs "Rum and Coca-Cola" and "Année Passée", the composers of which were involved in a copyright infringement case [30]. The melodies are shown in Figure 8.1. The songs have a lot in common melodically, but are not identical. The melody extract of “Année Passée” is represented by the following interval sequence:

3 0 2 -5 3 0 2 -5 3 0 2 -5 3 0 -1 -1 -3 3 0 2 -5 3 0 2 -5 3 0 2 -5 3 -1 -4

The first 21 notes of the “Rum and Coca-Cola” extract are represented as follows:

1 2 0 2 0 -2 0 2 -5 3 0 2 -5 3 0 -2 -3 3 0 2

When LCS is applied, a common subsequence (0 2 0 2 -5 3 0 2 -5 3 0 -3 3 0 2) of length 15 is found.

A dynamic programming approach can be used to calculate the length of this maximal subsequence [64]. A two-dimensional array is filled according to the formula where a represents the array, q and p represent the query melody string and piece to match against respectively, array index i ranges from 0 to query length, and index j ranges from 0 to piece length:

$$a[i,j] = max \begin{cases} a[i-1,j] & i \geq 1 \\ a[i,j-1] & j \geq 1 \\ a[i-1,j-1]+1 & q(i) = p(j) \text{ and } i,j \geq 1 \\ 0 & \end{cases} \tag{8.1}$$

The array values for the above example are shown in Figure 8.2.

Our initial hypothesis was that the longest common subsequence may be useful for matching with extra or omitted notes. As can be seen in the examples above, the technique may work well for long matches. However, there is the potential for many false matches, particularly for short queries. Its usefulness would depend greatly on the methods of melody standardisation used. It may be more suited to representations such as the *key-relative* group described in the chapter on standardisation. In these methods, there is no loss of key information when a note is omitted or added in a match.

8.1.2 Local Alignment

For local alignment, dynamic programming is used to determine the region of best match for two strings. The technique involves calculating a score, incorporating a penalty for mismatches, insertions, and deletions. The resulting score represents how good the best local alignment is, and the positions at which to best align the two strings.

For example, using the two melody fragments above, we would fill a two-dimensional array of values according to the following formula, where a represents the array, q and p represent the query melody string and piece to match against respectively, array index i ranges from 0 to query length, and index j ranges from 0 to piece length:

$$a[i,j] = max \begin{cases} a[i-1,j]+d & i \geq 1 \\ a[i,j-1]+d & j \geq 1 \\ a[i-1,j-1]+e & q(i)=p(j) \text{ and } i,j \geq 1 \\ a[i-1,j-1]+m & q(i) \neq p(j) \\ 0 & \end{cases} \tag{8.2}$$

where d is the cost of an insert or delete, e is the value of an exact match, and m is the cost of a mismatch. In this example we used $d = -2$, $e = 1$ and $m = -1$.

The resulting array for our example melody fragments is shown in Figure 8.3. The best alignment commences with a common substring (3 0 2 −5 3 0 2 −5 3 0). This is followed either by an indel (insert or delete) and a mismatch, or vice versa, representing the (−1 −1) substring matching against (−2). The alignment ends with the common substring (−3 3 0 2).

As occurs with the longest common subsequence method, the interval representation causes a single different note to be represented as two different adjacent symbols. However, in contrast with longest common subsequence, local alignment rewards adjacent symbols and has a significant penalty for gaps when matching. As a result, a long run of exactly matched symbols will not be greatly penalised by a double mismatch, as seen in the example above.

8.1.3 Longest Common Substring

The longest common substring of two strings can also be found using dynamic programming techniques. In this case, the algorithm does not permit indels or mismatches to contribute to a match. Instead, scores are reset to zero whenever the two strings do not match. Matching the correct and incorrect Beethoven's Fifth fragments described above would result in the same scores as for local alignment. When matching "Année Passée" to "Rum and Coca-Cola", the resulting score is 9, from the longest common substring (0 2 −5 3 0 2 −5 3 0). The advantage of this method is that there are no incorrect matches on pitch due to key shifts from mismatches and indels.

8.1.4 Thresholded Alignment

The technique that we have named *thresholded alignment* is a variation of local alignment. The technique gives a score of zero to matches that are less than k characters long. This technique aims to reduce the number of trivial matches. In the case of Beethoven's Fifth matched to Au Clair de la Lune, with a threshold of four, the score would be zero. Matching "Année Passée" to "Rum and Coca-Cola" has a score of six, as scores for matches start to be accumulated once a substring match reaches a length of four.

8.1.5 Cumulative Weight Matching

Another variation explored in our experiments increases the weight to be applied each time a new adjacent symbol is found in a match. The aim of this technique is to reward long matches. For example, with an arithmetic cumulative factor of 1, and a match weight of 1, the first match in a contiguous set of matches would contribute 1 to the score, the second adds 2, the third 3, and so on, so that the score for three consecutive matches, as occurs in the Beethoven's 5th example and its incorrectly played version, results in a score of 6. As soon as a mismatch or indel occurs, the cumulative weights are set back to zero.

8.1.6 Other Dynamic Programming Approaches

Some of the above approaches are basic techniques that have already been applied to other types of string-based information. Musical information is multi-dimensional, however, so the techniques can be extended to capture more information for comparison. For example, Mongeau and Sankoff extended their edit distance computations to use two dimensions—pitch and rhythm [97]. The pitches are relative to the key of the melody. A set of weights was devised to distinguish between consonant intervals (pleasing intervals such as octaves, and those that make up a major chord) and dissonant intervals (intervals that sound unpleasant together, such as semitones). Other techniques used were fragmentation and consolidation. This involves matching a single long note to repeated short notes with only a small penalty score. The method was tested on two compositions by Mozart and for this small data-set, successfully grouped musical themes and their variations. However, the example piece, unlike many pieces, remains in one key. The technique does not locate all occurrences of a theme or melody when a piece modulates to a different key [137].

Later, Kageyama et al. [71] applied a simplified approach that combines pitch with rhythm contour, each with a different set of weights in their dynamic programming-based algorithms. The technique was shown to rank highly the melody on which hummed queries were based, in a database of 500 melodies.

In chapter 11, we discuss another technique that compensates for the double errors that occur when a single pitch is incorrect.

8.2 N-grams and Music Retrieval

Melody strings can be broken down into n-grams, or substrings of a given length n for matching. In information retrieval, n-grams are sometimes used as terms for matching. The calculation of similarity is based on the existence or frequency of terms in the query and in the collection of documents. In the case of melodies, n-grams taken from the string representation of a melody can be used as terms for computing similarity for ranking.

The length of a typical melodic query is likely to affect the usefulness of n-grams. We expect that the first attempt at a melody query by a user will consist of the notes of a single phrase. This may be as little as four notes but is often considerably longer. The chosen length for n-grams not only affects the answers retrieved, but places a limit on the minimum length of queries that can be processed. However, queries that are shorter than typical n-gram lengths are unlikely to be successful with most matching methods on a substantial music collection. As our chosen melody standardisation methods represent note transitions instead of notes, the number of notes represented by an n-gram is one more than the n-gram size. For example, an n-gram of length five, such as the contour n-gram SSDUS, represents six notes.

In the experiments described in chapter 9 we tested several ways of computing a similarity score using n-grams: coordinate matching, sum common, the Ukkonen measure, and TF-IDF. These are illustrated below using a contour representation.

8.2.1 The Sum Common Measure

The *sum common* measure was calculated as follows (shown using notation from Zobel and Moffat [152]):

N-grams	SSUDDDDDD	SSUDDDUDDD	Sum	Coord.	Ukkonen	IDF	TF-IDF
SSU	1	1	1	1	0	1.31	1.31
SUD	1	1	1	1	0	1.12	1.12
UDD	1	2	1	2	1	1.03	1.03
DDD	4	2	2	1	2	1.09	2.18
DDU	0	1	0	0	1	1.03	1.03
DUD	0	1	0	0	1	1.02	1.02
			6	4	5		7.69

Table 8.1: *The 3-grams of two similar contour strings and the resulting similarity scores. The "sum common" (Sum) method gives a score of 5, coordinate matching (Coord.) a score of 4, and the Ukkonen measure a score of 5. Using some IDF figures based on the all-mono melody collection, the TF-IDF score for the pair is 7.69.*

$$S_{q,d} = \sum_{t \in \tau_{q,d}} f_{d,t} \tag{8.3}$$

where $\tau_{q,d}$ is the set of distinct n-grams that occur in both the query q and the melody d, $f_{d,t}$ is the frequency of n-gram t in the melody. That is, for every distinct n-gram in the query we counted the number of occurrences of the n-gram in the stored piece.

As an example, consider 3-grams and the contour representation of the Mozart melody example from chapter 6, SSUDDDDDD, and compare it to a slightly different contour string, SSUDDDUDDD. The frequency of each of the 3-grams is shown in Table 8.1. The score is the sum of the frequencies of the 3-grams in the melody that also occur at least once in the query, in this case 6.

8.2.2 The Ukkonen Measure

The Ukkonen measure [135] is a difference measure, that is, it counts the number of n-grams in each string that do not occur in both strings. To make it consistent with our other measures, we negate it. The formula used is:

$$S_{q,d} = -\sum_{t \in \tau} |f_{q,t} - f_{d,t}| \tag{8.4}$$

where τ is the set of possible n-grams, $f_{q,t}$ and $f_{d,t}$ represent the frequency of term t in the query and document respectively.

In the example shown in table 8.1, the Ukkonen measure results in a score of -5, which is the sum of the differences between query and piece shown in the "Ukkonen" column of the table.

8.2.3 The Count Distinct Measure

Also known as coordinate matching, the *count distinct* measure ignores term frequency and just counts the number of distinct n-grams that occur in both strings. The formula is:

$$S_{q,d} = |\tau_{q,d}| = \sum_{t \in \tau_{q,d}} 1 \tag{8.5}$$

where $\tau_{q,d}$ is the set of distinct n-grams that occur in both the query and the melody.

For example, in the example shown in table 8.1, the query and melody have four distinct n-grams in common. The *count distinct* measure ignores the fact that some of these n-grams occur more than once in the query and melody.

8.2.4 The TF-IDF Measure

There are many variations of the TF-IDF similarity measure, so called because they include the *term frequency* (TF) and the *inverse document frequency* (IDF) in some form. They all combine information about the frequency of terms in the documents and the frequency of a term in the collection as a whole. A term that is very common in a collection, such as the word "the" in a text collection or a sequence of repeated notes in a music collection, will not be very discriminating as a dominant component of a similarity measure. Important terms for searching tend to be less frequent. For this reason similarity measures of this class multiply weights by the IDF or a related factor which increases the weight of important, less frequent terms and decreases the weight of frequent, less important ones.

TF-IDF is the family of similarity measures based on summing the weights of the n-grams in the piece that match n-grams in the query. In one formulation, the weight of an n-gram is determined by its frequency in the piece and by the reciprocal of the number of pieces containing it. The formulae we use for TF-IDF, using the same notation as defined above, are:

$$S_{q,d} = \sum_{t \in \tau_{q,d}} \frac{f_{d,t} N}{f_t + 1} \tag{8.6}$$

and

$$S_{q,d} = \sum_{t \in \tau_{q,d}} = f_{d,t} \log(\frac{N}{f_t + 1}) \tag{8.7}$$

where f_t is the number of pieces in the collection that contain the n-gram t, and N is the number of pieces in the collection . One is added to the denominator to avoid division by zero in the implementation.

As an example, consider 3-grams and the contour representation of the Mozart melody example, SSUDDDDDD, and compare it to a slightly different contour string, SSUDDDUDDD. Using some IDF figures based on the *all-mono* melody collection, the TF-IDF score for the pair is 7.69.

8.3 Efficiency Considerations

When a query is presented to an information retrieval system, the query is transformed to allow it to be applied to the index or to allow straightforward comparison with items in the database. There are essentially two approaches to query evaluation: search the entire collection or look up terms in an index. A third approach consists of using a combination of these two. Typically, the use of an index makes searching much faster than searching directly through the collection. Improvements can be made in direct searching of a collection by using a compact representation of the most salient features for successful query evaluation and using the fastest available algorithms.

Dynamic programming is a technique that allows strings to be compared and the best matches found, by aligning strings and producing a score representing how similar (or different) they are. When applied to a string database, such as a collection of strings representing melodies, the entire collection is searched to find the best matches. The naive technique uses a significant amount of storage space, in that a two-dimensional array is kept in memory. If the strings being compared are very long then this can be a significant factor. The actual processing time is O(mn). Again, if the amount of text to be processed is large then this can be slow. There are techniques published that reduce the space and speed of this approach (for example, see Apostolico and Galil [1] for an overview of efficient serial and parallel dynamic programming algorithms) however the use of an index to speed up processing would be preferable.

An indexing strategy that has been shown to be effective for personal name searches, spell-checking [151] and genomic database searching [142] combines a coarse search using an n-gram index with a fine search of the results retrieved by the index. For genomic data searching, Williams [140] constructed frames that are bounded by n-gram matches in the strings being compared, and then a dynamic programming technique is used to compare the best of a pool of matches. Whether this is necessary for music is not really clear. Our results have shown that n-gram indexes work about as well as dynamic programming for manual queries in particular for our data set. Regardless of whether dynamic programming is needed, indexing is achievable and therefore desirable. We discuss approaches to indexing n-grams for music below.

8.3.1 Music Index Terms

There are many candidates for music index terms. There are some aspects that need to be considered before deciding on the appropriate index term.

First, a music query will not necessarily begin at the start of a melody. It may, however, begin at the start of a musical phrase, as suggested by Kageyama et al. [71]. Queries may possibly be chordal or may even cross from one musical part to another. Indeed, it may be a part of the melody that is being queried, or a part of the accompaniment.

We assume that melodic queries will be used, possibly with the melodies coming from any musical part. To allow for queries that start anywhere within a melody, the indexing of n-grams is a useful approach. In this case, the symbols represent absolute or relative pitches and possibly other features of each note.

8.3.2 Building an N-Gram Index

When using n-gram terms, an approach to indexing that has been used for genomic data involves a hash index with pointers to a list of occurrences of the n-gram [139]. This is particularly useful for genomic data as the hash index can be quite small despite using a simple perfect hashing function that maps each of the four characters to two binary digits. If a simple contour representation is being used for representing melody, then a similar approach can be used. However, once a representation exceeds a certain size, it may be necessary to consider alternative methods as the hash index size grows rapidly with alphabet size. For example, contour 4-grams can be represented with $3^4 = 81$ slots, but the directed modulo 12 standardisation method with

Standardisation Method (Alphabet Size)	Extraction Method	DB Size (uncompressed K)	DB Size (compressed K)
Contour (3)	all-channels	20,803.5	2,413.8
	all-mono	9,670.7	1,456.5
	entropy-channel	4,558.6	694.4
	entropy-part	2,744.1	514.4
	top-channel	2,905.0	452.2
Directed Modulo-12 (25)	all-channels	20,803.5	4025.6
	all-mono	9,670.7	3235.0
	entropy-channel	4,558.6	1323.5
	entropy-part	2,744.1	856.2
	top-channel	2,905.0	802.9
Exact Interval (255)	all-channels	21005.7	4354.1
	all-mono	9701.3	3910.1
	entropy-channel	4589.3	1466.5
	entropy-part	2774.8	912.2
	top-channel	2935.7	894.6

Table 8.2: *The relative sizes of extraction and standardisation methods for the collection of 10,466 MIDI files (and 69,032 channels). Database sizes are in kilobytes. The last column shows the compressibility of the database, by showing the size after compression with gzip.*

alphabet size of 25 requires $25^4 = 390625$ slots.

The n-gram approach used for our experiments consisted of using the first part of the n-gram to calculate an index to an array structure. Each array element then pointed to a binary tree of n-gram information. This allowed rapid look-up of n-grams that occurred in the query string. The pieces in the database were then scanned for n-grams and tallies kept of the number of common n-grams or different n-grams as appropriate for the type of similarity measure being applied for the query. Thus we preprocessed the query to create a rapid means of locating its n-grams and used an exhaustive search of the standardised collection in memory to find answers. While this approach was applicable for experiments testing multiple MIR methods, a practical MIR system would benefit from an indexed collection.

8.3.3 The Size of Music Representations

All the melody standardisation methods discussed in chapter 6 can be compressed to allow more efficient storage and matching. When deciding on an appropriate implementation for a music retrieval system, it is useful to look at the space and speed trade-offs in relation to the retrieval effectiveness of different methods. Table 8.2 shows the relative sizes of the extraction and standardisation methods explored in this thesis. The figure for the compressed size of the database was produced by compressing the file containing the standardised melodies with a standard compression utility (gzip). The uncompressed size relates directly to the number of symbols used to represent the melodies. These figures show that despite the raw size of the *all-channels* database, it compresses very well compared to the other databases, so that it is not much larger than the *all-mono* database. Databases represented by *exact interval* standardisation are not much larger than those using *directed modulo-12* standardisation. *Contour* standardisation results in a compressed database that is approximately half the size of the others. While the figures in Table 8.2 are a guide, the relationship between the size of indexes on this information may be slightly different, as it will depend on how the indexed information is organised and compressed.

8.4 Other Music Matching Techniques

While dynamic programming and n-gram techniques are our main focus and have been prototyped in different ways by other researchers [11, 93, 97], there have been a range of other techniques applied to the problem. Many researchers have applied variations on Baeza-Yates and Gonnet's [80], Baeza-Yates and Perleberg's [54] or Wu and Manber's [80, 92, 146] string matching techniques. These techniques are discussed in more detail in chapter 3. Some have applied polyphonic matching techniques [23, 34, 80], feature vectors [75, 76], rhythm, or chord matching [20, 21]. These are discussed briefly below.

Dovey's work [34] discusses polyphonic matching from a theoretical standpoint, and describes how a brute-force algorithm can be applied to matching with absolute pitches. Each instant in which notes commence is represented by a bit array containing a one for each pitch that starts at that instant. The approach consists of attempting a melody match at each of the possible pitches. To cut down on irrelevant matches, Dovey suggests having a limit to the number of events that can occur between matching notes. The approach was not evaluated for effectiveness. Dovey's approach ensures that there is a high recall of pieces that match a query exactly. However, there is likely to be low precision due to the comprehensive nature of the matching process producing many matches that do not make musical sense.

Clausen et al. [23] engineered an index solution to the music search problem. They stored absolute pitch and time for each note in an index. Matches were carried out by quantising the query and piece to the same level, for example semiquavers (16th notes). A sequence of notes was only matched if it occurs at the same position in a bar. Fuzzy matching involving sets of notes was also discussed. The technique was not evaluated for effectiveness. Clausen et al. sensibly combined pitch and time for their approach as the precision would otherwise be poor. The technique may not be robust for query-by-humming queries where there is pitch drift, however. There is also the limitation that queries must match the position in the bar. For example, if a melody was shifted by two beats compared to the matching piece in the database, the match would not be found. A further type of match that would not work well is one in which an arrangement changes the timing, as often happens in popular arrangements of traditional and classical pieces. For example, melodies by Bach have been converted from triple time to quadruple time for inclusion in popular songs.

Lemström et al. converted Baeza-Yates' and Gonnet's shift-or algorithm to the problem of matching polyphonic music [80]. They applied the technique to absolute pitch matching by using bitwise *and* to combine all the notes that occur in a single instant before applying the bitmask at each iteration through the piece. A further variation was developed that uses intervals instead of absolute pitches. In this case an algorithm collected appropriate intervals to match against before being applied to the mask. Due to the lack of filtering the approach is likely to have low precision.

Kosugi et al. [75, 76] applied the concept of *feature vectors* to music retrieval. Two main types of information were stored in the feature vectors. First, a "tone transition" feature vector contained pitch sequence information relative to divisions within a bar. Pitches were stored relative to a "base tone", which could be the most frequent pitch in the bar. The second feature vector stored "tone distribution", a vector representing the number of time divisions containing each pitch. Combining the two feature vectors in answering queries produced a slight improvement in retrieval effectiveness for a set of 186 hummed queries on a database of 10,069 MIDI files. The main limitation of the approach is that queries that are sung in a way that shifts their start position relative to the bar will not be successful. This will affect the retrieval of different arrangements of pieces, as variations in rhythm are quite common, such as anticipating a note, by playing it slightly before the beat.

Chou et al. [21] used a "PAT-tree" structure to store melodic information in the form of chords. This is a suffix tree structure that stores sub-strings for inexact string matching. All possible substrings are represented at the leaf nodes, and a tree structure leading to them is built based on each character sequence. Chou et al. implemented their PAT-tree by building a B+ tree of "chords" made up of the notes found in each bar of each melody. This system appeared to use absolute pitches in its representation. The indexing was quite coarse and could only be applied to pieces that remained in one key throughout, thus the technique is not scalable to large collections. In later work by Chen et al. [18] the suffix tree approach has been applied to other representations of melody fragments that include both pitch and duration information. The technique was not evaluated for effectiveness, however the precision should be quite good for exact matching.

8.5 Summary

Of the candidate methods for melody matching, we have explored several variations of dynamic programming and n-gram-based searching. The dynamic programming approach involves matching the query against each melody in the database. In contrast, the n-gram approach, is implemented as an index. When used with standardisation techniques based on pitch transitions these matching methods should allow successful matching regardless of the key in which pieces and queries are presented, or the beats per bar, and yet be sufficiently discriminating due to the melody extraction phase. Both the local alignment and n-gram techniques allow for errors in the matching process, and at the same time provide some flexibility for the location of different arrangements of the same piece. Other researchers have applied different techniques, some of which may be valuable. However, we have limited this work to the above-mentioned techniques. These techniques were implemented and used with the three main standardisation methods explored in this thesis, that is, contour, directed modulo-12, and exact interval standardisation. The success of each method is evaluated in our experiments in the next chapter.

		3	**0**	**2**	−5	3	**0**	**2**	**−5**	**3**	**0**	**2**	**−5**	**3**	**0**	−1	−1	**−3**	**3**	**0**	**2**	−5	**3**	**0**	**2**	−5	**3**	**0**	**2**	−5	3	−1	−4
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	**1**	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	0	0	1	**2**	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	1	2	2	2	**3**	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
−2	0	0	1	2	2	2	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	0	0	1	2	2	2	**3**	3	3	3	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	0	0	1	2	2	2	3	**4**	4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	6	6	6
−5	0	0	1	2	3	3	3	4	**5**	5	5	5	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7
3	0	1	1	2	3	4	4	4	5	**6**	6	6	6	7	7	7	7	7	7	7	7	7	8	8	8	8	8	8	8	8	8	8	8
0	0	1	2	2	3	4	5	5	5	6	**7**	7	7	7	8	8	8	8	8	8	8	8	8	9	9	9	9	9	9	9	9	9	9
2	0	1	2	3	3	4	5	6	6	6	7	**8**	8	8	8	8	8	8	8	8	9	9	9	9	10	10	10	10	10	10	10	10	10
−5	0	1	2	3	4	4	5	6	7	7	7	8	**9**	9	9	9	9	9	9	9	9	10	10	10	10	11	11	11	11	11	11	11	11
3	0	1	2	3	4	5	5	6	7	8	8	8	9	**10**	10	10	10	10	10	10	10	10	11	11	11	11	12	12	12	12	12	12	12
0	0	1	2	3	4	5	6	6	7	8	9	9	9	10	**11**	11	11	11	11	11	11	11	11	12	12	12	12	13	13	13	13	13	13
−2	0	1	2	3	4	5	6	6	7	8	9	9	9	10	11	11	11	11	11	11	11	11	11	12	12	12	12	13	13	13	13	13	13
−3	0	1	2	3	4	5	6	6	7	8	9	9	9	10	11	11	11	**12**	12	12	12	12	12	12	12	12	12	13	13	13	13	13	13
3	0	1	2	3	4	5	6	6	7	8	9	9	9	10	11	11	11	12	**13**	13	13	13	**13**	13	13	13	**13**	13	13	13	14	14	14
0	0	1	2	3	4	5	6	6	7	8	9	9	9	10	11	11	11	12	13	**14**	14	14	14	**14**	14	14	14	**14**	14	14	14	14	14
2	0	1	2	3	4	5	6	7	7	8	9	10	10	10	11	11	11	12	13	14	**15**	15	15	15	**15**	15	15	15	**15**	15	15	15	15

Figure 8.2: *"Année Passée" and "Rum and Coca-Cola" matched using LCS. The interval sequence for "Année Passée" is shown in the horizontal margin, and "Rum and Coca-Cola"'s interval sequence is shown in the vertical margin. Components of the longest common subsequences are shown in bold. Arrows indicate the different possible longest common subsequences. Vertical and horizontal arrows indicate gaps in the sequence; diagonals indicate a matching pair of symbols.*

		3	0	2	-5	3	0	2	-5	3	0	2	-5	3	0	-1	-1	-3	3	0	2	-5	3	0	2	-5	3	0	2	-5	3	-1	-4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
2	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0	0
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
2	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0	0
-5	0	0	0	0	3	1	0	0	3	1	0	0	3	1	0	0	0	0	0	0	0	3	1	0	0	3	1	0	0	3	1	0	0
3	0	1	0	0	1	4	2	0	1	4	2	0	1	4	2	0	0	0	1	0	0	1	4	2	0	1	4	2	0	1	4	2	0
0	0	0	2	0	0	2	5	3	1	2	5	3	1	2	5	3	1	0	0	2	0	0	2	5	3	1	2	5	3	1	2	3	1
2	0	0	0	3	1	0	3	6	4	2	3	6	4	2	3	4	2	0	0	0	3	1	0	3	6	4	2	3	6	4	2	1	2
-5	0	0	0	1	4	2	1	4	7	5	3	4	7	5	3	2	3	1	0	0	1	4	2	1	4	7	5	3	4	7	5	3	1
3	0	1	0	0	2	5	3	2	5	8	6	4	5	8	6	4	2	2	2	0	0	2	5	3	2	5	8	6	4	5	8	6	4
0	0	0	2	0	0	3	6	4	3	6	9	7	5	6	9	7	5	3	1	3	1	0	3	6	4	3	6	9	7	5	6	7	5
-2	0	0	0	1	0	1	4	5	3	4	7	8	6	4	7	8	6	4	2	1	2	0	1	4	5	3	4	7	8	6	4	5	6
-3	0	0	0	0	0	0	2	3	4	2	5	6	7	5	5	6	7	7	5	3	1	1	0	2	3	4	2	5	6	7	5	3	4
3	0	1	0	0	0	1	0	1	2	5	3	4	5	8	6	4	5	6	8	6	4	2	2	0	1	2	5	3	4	5	8	6	4
0	0	0	2	0	0	0	2	0	0	3	6	4	3	6	9	7	5	4	6	9	7	5	3	3	1	0	3	6	4	3	6	7	5
2	0	0	0	3	1	0	0	3	1	1	4	7	5	4	7	8	6	4	4	7	10	8	6	4	4	2	1	4	7	5	4	5	6

Figure 8.3: *Local alignment of melody fragments of the songs "Année Passée" (horizontal margin) and "Rum and Coca-Cola" (vertical margin). Once the array has been filled, the arrows, representing how the values pointed to were calculated, are followed to determine the parts that have been aligned.*

Chapter 9

Melody Matching Experiments

Having developed a range of methods for melody matching, we now need to test them. Our aim in these experiments was threefold: first, to confirm that the three-stage framework can be used to find matching melodies in a large corpus of musical works; second, to identify appropriate techniques for each stage of the retrieval process; and third, to verify that our IR-based methodology could be successfully applied to music.

As outlined in chapter 4, for the measurement of a retrieval system we need [113]: a collection of melodies, a collection of queries, and, for each query, relevance judgements as to which of the melodies are similar to each query. When a system is used to determine a ranking of melodies for a given query, the relevance judgements can be used to assign a score to the retrieval system; a system should get a high score if it is good at highly ranking similar melodies. We used precision-at-k and eleven-point precision average measures (described in Chapter 4) in the experiments described below. We have chosen precision-at-ten as the number of relevant answers for each query in our collection is usually less than 10. To our knowledge this was the first application of this standard system measurement technique to music retrieval.

9.1 The Query Set

The experiments described in this chapter used query melodies automatically extracted from pieces of music, using the algorithms described in Chapter 5. The pieces of music on which the queries were based were selected as described in Section 4.3. All arrangements were designated as the relevant pieces for each of the queries, including duplicate arrangements. Some alternative arrangements were not identified, usually where non-obvious names had been chosen for the files. These were classed as irrelevant in the evaluation process.

Typically there were two to six relevant pieces of music for each query, on the assumption that all other pieces are not similar—a reasonable assumption if the retrieval task is to find variant forms of the same piece of music. While the assumption may not be valid for other retrieval tasks, it should not discriminate against particular retrieval methods.

Two query sets were created, one containing 28 queries and the other 51. For each piece, the query was randomly selected from the set of versions. The experiments reported here use the smaller set of queries, but the results with the set of 51 queries were entirely consistent, and therefore have not been included for all experiments. The query sets contain melodies from a range of music genres, including pop and rock music from every decade since the fifties, some jazz works, classical compositions, country music, a Christmas carol, and several TV and movie themes. After the queries were chosen, the various extraction and standardisation techniques were applied. To simulate queries of varying lengths, each string was truncated to 10, 20, 40 and 80 notes, thus giving four versions of the collection of queries. This range approximates the range of manual queries gathered later for our manual experiments.

9.2 Dynamic Programming Experiments

In this first experiment, our aim was to determine whether techniques based on dynamic programming would work well at matching melodies in a large music database. We tested three dynamic programming techniques for their suitability to the task: local alignment, longest common subsequence, and longest common substring. Each technique was tested with variation in several parameters, including melody extraction method, melody standardisation type, and song length normalisation. We tested the techniques against databases of automatically extracted melodies and also the "all-channels" database that consists of melodies extracted from each channel of each piece in the collection.

A common experimental methodology for a retrieval system is to ask users to specify queries, but it is difficult to eliminate the possibility that the experimenter has influenced the query development process. For example, for a query-by-contour system an experimenter could subconsciously encourage less accurate queries than those that might be specified in other contexts. The query set we have gathered for these experiments does not have this bias, and therefore

			10			20			40			80		
			0	1	L	0	1	L	0	1	L	0	1	L
a	amdb	c	01.77	00.06	00.63	18.46	00.08	07.99	35.86	00.13	25.76	40.05	00.33	35.02
		d	**40.11**	00.25	36.69	**51.86**	00.41	**50.64**	**51.89**	00.89	48.96	**53.32**	02.04	**49.79**
		i	**42.32**	00.07	35.34	**49.06**	00.15	**46.87**	**48.08**	00.75	44.83	**48.76**	01.96	**45.87**
	ecdb	c	00.99	00.15	01.06	15.57	00.29	08.03	31.93	00.78	23.87	32.10	02.71	29.61
		d	**33.59**	00.57	26.77	**38.61**	02.22	36.32	38.05	04.28	37.33	39.49	04.06	36.38
		i	**34.22**	00.49	26.69	37.61	02.08	35.98	37.51	03.98	36.35	38.40	04.82	36.04
	epdb	c	00.23	00.19	01.30	10.19	00.77	10.18	21.70	02.57	23.00	22.18	03.55	22.85
		d	13.93	01.21	12.76	23.41	02.29	23.64	23.57	03.69	23.35	25.83	05.59	26.87
		i	15.85	00.96	12.98	23.42	02.27	23.12	24.88	03.40	24.60	26.00	04.97	27.28
	tcdb	c	02.52	00.09	01.26	18.68	00.14	11.12	32.02	00.17	24.62	31.73	00.39	28.82
		d	30.08	00.20	19.42	35.48	00.40	31.96	37.63	00.93	36.98	39.52	00.77	38.83
		i	30.73	00.14	19.65	36.41	00.42	32.80	36.76	00.70	36.49	39.64	00.75	39.08
l	amdb	c	00.01	00.04	00.04	00.01	00.04	00.06	00.01	00.04	00.15	00.08	00.04	01.72
		d	00.00	00.04	00.34	00.15	00.05	02.26	00.15	00.06	10.56	00.92	00.10	27.77
		i	00.31	00.06	00.74	00.40	00.09	06.71	01.50	00.07	20.31	07.63	00.13	34.62
	ecdb	c	00.00	00.19	00.36	00.00	00.25	00.49	00.07	00.20	01.44	00.25	01.49	04.34
		d	00.07	00.29	03.92	00.48	00.55	06.54	00.97	00.97	16.87	02.78	02.11	27.85
		i	00.33	00.27	03.85	00.84	00.41	09.79	02.79	01.05	19.12	10.47	02.68	27.97
	epdb	c	00.01	00.17	00.26	00.00	00.21	00.91	00.01	01.90	03.26	00.29	02.32	06.92
		d	00.16	00.50	03.58	01.72	01.24	07.51	02.14	02.57	13.22	06.45	02.93	20.94
		i	00.11	00.48	03.44	00.77	01.22	07.41	04.34	02.81	14.62	08.77	03.17	21.49
	tcdb	c	00.00	00.03	00.42	00.25	00.05	02.03	00.12	00.05	03.03	00.16	00.47	15.20
		d	00.41	00.11	04.48	00.50	00.14	14.02	01.85	00.19	23.93	05.08	00.68	26.50
		i	00.23	00.07	04.45	01.38	00.29	15.27	02.63	00.34	23.41	11.90	00.54	27.93
s	amdb	c	01.84	00.06	00.91	18.76	00.11	16.94	35.08	00.21	32.15	37.56	00.50	39.22
		d	**41.16**	00.25	37.51	**48.79**	00.42	**46.31**	**47.37**	00.87	**45.51**	**47.86**	01.59	**47.19**
		i	**42.32**	00.07	35.34	**49.06**	00.15	**46.87**	**48.08**	00.75	44.83	**48.76**	01.96	**45.87**
	ecdb	c	01.92	00.16	01.26	18.19	00.40	12.56	30.04	01.57	26.75	33.63	02.72	31.15
		d	**31.70**	00.57	26.04	**38.28**	02.23	35.63	37.30	04.30	36.50	36.72	04.67	37.82
		i	**34.22**	00.49	26.69	37.61	02.08	35.98	37.51	03.98	36.35	38.40	04.82	36.04
	epdb	c	00.24	00.19	01.34	10.81	00.79	14.54	22.45	02.47	22.39	22.71	03.33	23.69
		d	14.33	01.21	12.81	22.95	02.29	22.77	24.61	03.54	23.59	25.90	05.61	26.96
		i	15.85	00.96	12.98	23.42	02.27	23.12	24.88	03.40	24.60	26.00	04.97	27.28
	tcdb	c	02.95	00.03	01.52	20.07	00.09	14.17	30.52	00.17	28.01	32.16	00.25	32.67
		d	29.40	00.20	20.52	33.20	00.43	33.16	35.19	00.88	35.95	39.38	00.71	40.27
		i	30.73	00.14	19.65	36.41	00.42	32.80	36.76	00.70	36.49	39.64	00.75	39.08

Table 9.1: *Eleven-point recall-precision averages (as percentages) for dynamic programming-based matching without rests. Query lengths are 10, 20, 40, and 80. Melody length normalisations shown are: no normalisation (0), divide by the song length (1), and log normalisation (L). Similarity measures are: local alignment (a), longest common subsequence (l), and longest common substring (s). Melody standardisations are contour (c), directed modulo-12 (d), and exact interval (i). Melody extraction methods used for both the query and the database are: all-mono (am), entropy-channel (ec), entropy part (ep), and top-channel (tc). The best precision average — and values that were not statistically significantly different from it — for each query length is highlighted.*

			10			20			40			80		
			0	1	L	0	1	L	0	1	L	0	1	L
a	amdb	c	1.43	0.00	0.00	13.57	0.00	5.00	23.21	0.00	16.79	25.71	0.71	21.79
		d	**27.14**	0.00	25.36	**31.79**	0.00	**31.43**	**31.43**	1.07	**29.64**	**31.79**	1.43	**29.29**
		i	**27.86**	0.00	25.36	**31.79**	0.00	**29.64**	**31.43**	0.71	**28.93**	**30.71**	1.79	28.93
	ecdb	c	1.79	0.00	1.43	13.21	0.36	7.50	21.79	0.71	19.29	23.21	3.93	21.07
		d	**23.93**	0.00	20.36	**27.14**	2.14	25.71	**27.14**	5.00	25.00	**27.50**	4.29	26.07
		i	**25.36**	0.00	21.07	**27.14**	2.14	24.29	25.71	5.00	25.00	**27.50**	5.36	26.07
	epdb	c	0.36	0.00	0.71	10.00	0.36	8.21	19.29	3.57	18.57	19.29	4.29	19.29
		d	13.21	0.00	12.50	19.29	2.86	18.93	19.64	5.36	19.64	21.79	7.86	21.79
		i	14.29	0.00	12.50	19.29	2.86	18.93	19.64	4.64	19.64	21.79	7.86	21.79
	tcdb	c	2.50	0.36	0.71	14.29	0.36	7.50	22.14	0.36	17.86	22.50	0.36	21.79
		d	**23.21**	0.00	15.36	**26.79**	0.00	**25.36**	**27.50**	0.71	26.79	**27.50**	0.71	**27.50**
		i	**24.29**	0.00	16.79	**26.79**	0.00	**25.36**	**26.79**	0.00	**26.43**	**27.14**	0.36	26.79
l	amdb	c	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.43
		d	0.00	0.00	0.00	0.00	0.00	1.43	0.00	0.00	7.86	0.71	0.00	16.07
		i	0.00	0.00	0.36	0.00	0.00	5.71	1.79	0.00	15.00	9.29	0.00	24.29
	ecdb	c	0.00	0.00	0.71	0.00	0.00	0.36	0.00	0.00	0.71	0.00	1.07	4.64
		d	0.00	0.00	5.00	0.00	0.36	7.50	0.36	0.36	12.86	3.93	2.14	19.64
		i	0.36	0.00	5.00	0.36	0.36	9.64	4.29	0.36	15.00	10.36	2.50	20.71
	epdb	c	0.00	0.00	0.00	0.00	0.00	0.36	0.00	1.79	3.57	0.00	2.50	6.79
		d	0.00	0.00	4.29	1.07	0.00	7.86	3.21	3.21	11.79	10.00	3.57	18.57
		i	0.00	0.00	4.29	0.71	0.00	7.50	4.64	3.21	11.79	9.64	3.21	18.57
	tcdb	c	0.00	0.00	0.71	0.71	0.00	1.79	0.00	0.00	2.50	0.00	0.00	11.79
		d	0.71	0.00	4.64	0.00	0.00	10.71	1.79	0.00	17.86	5.36	0.71	19.29
		i	0.00	0.00	4.64	0.71	0.71	11.79	2.14	0.71	18.21	13.21	1.07	20.00
s	amdb	c	1.43	0.00	0.00	13.21	0.00	9.64	22.86	0.00	20.71	25.36	0.71	25.36
		d	**27.50**	0.00	25.36	**31.43**	0.00	**30.71**	**30.71**	1.07	**28.93**	30.36	1.43	28.21
		i	**27.86**	0.00	25.36	**31.79**	0.00	**29.64**	**31.43**	0.71	**28.93**	**30.71**	1.79	28.93
	ecdb	c	2.50	0.00	1.43	11.79	0.36	12.50	21.07	0.71	20.00	23.21	3.93	23.57
		d	22.14	0.00	19.64	**26.79**	2.50	23.93	26.07	5.00	25.00	26.07	5.36	26.07
		i	**25.36**	0.00	21.07	**27.14**	2.14	24.29	25.71	5.00	25.00	**27.50**	5.36	26.07
	epdb	c	0.00	0.00	1.43	8.93	0.36	12.50	19.64	3.57	19.64	19.29	4.29	19.64
		d	14.29	0.00	12.50	18.93	2.86	18.93	19.64	5.36	19.64	21.79	8.57	21.79
		i	14.29	0.00	12.50	19.29	2.86	18.93	19.64	4.64	19.64	21.79	7.86	21.79
	tcdb	c	2.86	0.00	0.71	16.07	0.00	9.64	22.86	0.00	21.43	22.50	0.00	23.57
		d	**23.21**	0.00	16.43	**25.00**	0.00	**25.36**	25.00	0.00	**25.71**	25.71	0.00	26.07
		i	**24.29**	0.00	16.79	**26.79**	0.00	**25.36**	**26.79**	0.00	**26.43**	**27.14**	0.36	26.79

Table 9.2: *Precision at 10 values (as percentages) for dynamic programming-based matching without rests. The same melody extraction methods used for both the query and the database.*

			010		020		040		080	
			0	L	0	L	0	L	0	L
acdb	amdb	c	05.98	01.36	11.31	06.32	13.79	11.22	14.86	10.41
		d	26.68	15.71	29.66	22.91	**34.90**	**31.40**	**32.58**	27.87
		i	25.00	16.66	29.71	22.12	**35.82**	29.81	**31.39**	27.83
	ecdb	c	08.28	00.66	22.33	11.65	**28.60**	24.60	**31.45**	29.88
		d	**27.82**	16.53	**33.45**	27.49	**36.94**	**34.53**	**35.68**	30.97
		i	**29.27**	17.12	**33.58**	27.47	**37.47**	**34.42**	**35.23**	30.78
	tcdb	c	04.72	02.87	20.21	16.50	**26.31**	24.14	**26.54**	25.43
		d	25.56	21.12	**29.82**	27.59	**31.60**	**31.21**	**29.87**	**29.97**
		i	**26.05**	20.28	**30.56**	29.04	**31.71**	**31.77**	**30.17**	**29.96**
amdb	amdb	c	05.83	02.09	22.26	15.85	**26.01**	25.33	26.24	26.68
		d	**30.33**	28.83	**34.83**	**35.44**	**37.04**	**35.84**	**33.14**	**32.36**
		i	**31.01**	29.76	**35.57**	**35.50**	**36.81**	**35.71**	**32.51**	**32.18**
ecdb	ecdb	c	07.24	06.70	17.35	15.45	21.20	20.16	23.31	24.01
		d	25.24	20.01	27.63	26.74	29.69	29.24	27.86	27.97
		i	**26.93**	20.72	**28.11**	27.10	**29.72**	29.43	28.29	28.04

Table 9.3: *Eleven-point recall-precision averages (as percentages) for local alignment matching with rests. The first column shows which database was used, and the second the melody extraction method used for the queries.*

should provide an excellent test of a music retrieval system. In addition, it is useful in its own right for answering queries that ask to find other versions of a given piece.

9.2.1 Method

For our experiment, we used each of our four melody extraction techniques in turn for both the queries and the collection. In addition, we used an *all-channels* technique that generated a separate melody from each channel, so that each piece could be represented several times. This reduces the chance of missing melodies but increases the chance of false matches—that is, recall should improve at some cost to precision. Use of all-channels increases the size of the collection to about six times the original. In this case, the collection contained 69,032 separate parts.

The melody standardisation methods tested were contour, directed modulo, and exact interval. In each case we tested inclusion and omission of rests. When rests were included, they were represented as a single symbol. Intervals following a rest were calculated using the note before the rest. Rests were inserted if there was a break of at least five time units between notes. This

			10			20			40			80		
			0	1	L	0	1	L	0	1	L	0	1	L
a	amdb	c	00.96	00.12	00.62	09.83	00.43	02.23	16.71	00.27	09.06	14.45	00.09	09.97
		d	**32.37**	00.06	14.78	**41.92**	00.10	33.39	**44.34**	00.21	38.48	**48.54**	00.20	42.56
		i	30.65	00.06	14.03	**43.49**	00.10	31.49	**45.18**	00.14	36.86	**47.96**	00.24	38.43
	ecdb	c	00.73	00.14	00.13	18.42	00.47	02.51	38.51	00.14	23.93	42.76	01.37	33.90
		d	**36.58**	00.02	11.01	**48.93**	00.10	38.12	**50.41**	00.21	44.97	**50.63**	00.09	46.51
		i	**39.03**	00.03	11.28	**49.26**	00.08	38.32	**47.73**	00.13	44.67	**50.15**	00.43	45.40
	epdb	c	00.16	00.11	00.09	07.18	00.21	01.46	16.79	00.15	08.40	18.03	00.10	17.83
		d	13.25	00.10	05.11	22.30	00.05	18.56	23.21	00.03	20.43	29.17	00.04	23.39
		i	13.16	00.21	04.95	23.45	00.01	18.45	21.65	00.04	20.53	29.65	00.04	23.24
	tcdb	c	00.61	00.17	00.48	19.31	00.14	09.06	37.55	00.15	26.97	40.35	01.38	33.28
		d	31.69	00.03	19.34	**42.57**	00.09	37.09	**47.17**	00.23	**42.43**	**48.06**	00.19	**45.15**
		i	**32.87**	00.06	18.63	**42.14**	00.11	38.08	**47.55**	00.25	**41.82**	**48.42**	00.55	**44.89**
l	amdb	c	00.01	00.08	00.06	00.01	00.47	00.10	00.00	00.09	00.28	00.01	00.03	01.76
		d	00.11	00.06	02.83	00.18	00.06	05.54	00.51	00.06	10.30	00.75	00.07	16.94
		i	00.41	00.05	03.73	00.66	00.07	07.36	02.34	00.17	14.21	03.47	00.07	16.78
	ecdb	c	00.01	00.12	00.01	00.00	00.04	00.13	00.00	00.01	01.06	00.07	00.09	06.33
		d	00.08	00.02	03.19	00.17	00.03	07.75	00.96	00.03	19.99	03.46	00.12	34.14
		i	00.63	00.02	03.38	00.59	00.03	08.13	03.68	00.02	20.82	10.92	00.14	36.06
	epdb	c	00.01	00.15	00.02	00.00	00.06	00.11	00.00	00.07	00.58	00.07	00.09	02.21
		d	00.11	00.05	01.15	00.09	00.06	02.92	00.21	00.04	08.47	00.97	00.02	15.44
		i	00.17	00.05	01.18	00.23	00.05	03.02	01.64	00.04	09.74	04.27	00.04	14.70
	tcdb	c	00.01	00.29	00.13	00.00	00.07	01.25	00.00	00.07	02.24	00.18	00.11	15.59
		d	00.07	00.03	03.87	00.26	00.04	15.09	01.06	00.05	25.46	03.60	00.14	32.05
		i	00.45	00.03	03.77	00.60	00.03	14.60	02.90	00.06	25.44	10.00	00.11	31.87
s	amdb	c	01.12	00.21	00.64	11.16	00.71	05.38	19.70	00.13	14.75	27.06	00.27	18.51
		d	**31.71**	00.06	14.61	**42.36**	00.10	29.86	44.32	00.20	34.04	**47.19**	00.20	38.11
		i	30.65	00.06	14.03	**43.49**	00.10	31.49	**45.18**	00.14	36.86	**47.96**	00.24	38.43
	ecdb	c	01.08	00.14	00.15	21.47	00.30	07.01	36.37	00.08	28.23	42.85	00.40	38.97
		d	**38.27**	00.03	12.03	**47.36**	00.09	39.38	47.40	00.19	44.04	**54.16**	00.13	48.46
		i	**39.03**	00.03	11.28	**49.26**	00.08	38.32	**47.73**	00.13	44.67	**50.15**	00.43	45.40
	epdb	c	00.42	00.19	00.10	07.09	00.09	02.61	14.94	00.10	13.05	18.10	00.09	16.71
		d	13.02	00.10	04.89	23.00	00.05	17.66	21.92	00.03	17.81	27.55	00.05	19.34
		i	13.16	00.21	04.95	23.45	00.01	18.45	21.65	00.04	20.53	29.65	00.04	23.24
	tcdb	c	00.96	00.15	00.57	19.52	00.31	11.03	34.57	00.21	31.63	40.96	00.39	37.73
		d	31.21	00.03	19.92	**41.32**	00.11	38.72	**43.71**	00.24	**42.71**	**46.42**	00.24	**46.89**
		i	**32.87**	00.06	18.63	**42.14**	00.11	38.08	**47.55**	00.25	**41.82**	**48.42**	00.55	**44.89**

Table 9.4: *Eleven-point recall-precision averages (as percentages) for dynamic programming-based matching against all channels without rests.*

			10			20			40			80		
			0	1	L	0	1	L	0	1	L	0	1	L
a	amdb	c	0.36	0.00	0.00	4.64	0.00	1.79	10.36	0.00	7.86	10.71	0.00	9.29
		d	**20.36**	0.00	12.50	**26.43**	0.00	22.86	**27.86**	0.00	28.57	**30.00**	0.00	**24.64**
		i	20.00	0.00	11.79	**27.50**	0.00	20.71	**27.86**	0.00	26.43	**27.86**	0.00	24.29
	ecdb	c	0.36	0.00	0.00	11.43	0.36	3.21	26.07	0.00	16.07	24.64	1.43	22.50
		d	**25.00**	0.00	9.64	**31.07**	0.00	25.71	**30.71**	0.00	27.86	**32.50**	0.00	28.57
		i	**27.86**	0.00	8.21	**31.07**	0.00	24.64	**29.29**	0.00	28.21	**32.14**	0.00	28.93
	epdb	c	0.00	0.00	0.00	5.36	0.00	1.79	11.43	0.00	7.14	11.43	0.00	10.71
		d	8.93	0.36	4.64	14.64	0.00	11.07	13.21	0.00	11.07	16.43	0.00	12.50
		i	9.64	0.36	4.64	15.00	0.00	11.07	13.21	0.00	11.07	16.43	0.00	12.50
	tcdb	c	0.36	0.36	0.71	13.21	0.00	6.43	22.86	0.00	18.21	22.50	1.43	21.07
		d	22.86	0.00	11.79	**27.86**	0.00	25.36	**28.21**	0.00	**27.14**	**28.21**	0.00	**28.21**
		i	**23.57**	0.00	12.14	**27.50**	0.00	25.36	**28.57**	0.36	**27.14**	**29.29**	0.00	**28.21**
l	amdb	c	0.00	0.00	0.00	0.00	0.36	0.00	0.00	0.00	0.71	0.00	0.00	1.79
		d	0.00	0.00	2.86	0.00	0.00	5.36	0.36	0.00	8.21	0.36	0.00	13.21
		i	0.71	0.00	4.29	0.71	0.00	8.57	2.50	0.36	12.86	2.50	0.00	12.86
	ecdb	c	0.00	0.00	0.00	0.00	0.00	0.36	0.00	0.00	1.43	0.00	0.00	6.43
		d	0.00	0.00	3.57	0.00	0.00	7.50	0.36	0.00	15.00	2.86	0.00	22.14
		i	0.36	0.00	3.57	0.71	0.00	8.93	4.29	0.00	16.43	11.79	0.00	22.50
	epdb	c	0.00	0.00	0.00	0.00	0.00	0.36	0.00	0.00	0.71	0.00	0.00	2.14
		d	0.00	0.00	1.43	0.00	0.00	2.86	0.00	0.00	7.14	1.07	0.00	8.93
		i	0.00	0.00	1.43	0.00	0.00	2.86	1.43	0.00	7.14	2.86	0.00	8.93
	tcdb	c	0.00	0.00	0.00	0.00	0.00	1.79	0.00	0.00	3.21	0.00	0.00	12.14
		d	0.00	0.00	4.64	0.00	0.00	10.71	0.36	0.00	17.50	3.57	0.00	21.07
		i	0.36	0.00	4.64	0.71	0.00	11.79	2.50	0.00	17.50	9.29	0.00	21.07
s	amdb	c	0.71	0.00	0.00	6.43	0.36	3.93	10.71	0.00	9.29	16.43	0.36	11.79
		d	**21.07**	0.00	12.50	**27.50**	0.00	20.71	28.21	0.00	22.86	**28.21**	0.00	22.14
		i	20.00	0.00	11.79	**27.50**	0.00	20.71	**27.86**	0.00	26.43	**27.86**	0.00	24.29
	ecdb	c	0.71	0.00	0.00	13.57	0.71	5.00	23.93	0.00	18.93	26.43	0.36	25.71
		d	**24.64**	0.00	11.07	**30.36**	0.00	26.43	29.29	0.00	28.57	**31.79**	0.00	**31.07**
		i	**27.86**	0.00	8.21	**31.07**	0.00	24.64	**29.29**	0.00	28.21	**32.14**	0.00	28.93
	epdb	c	0.36	0.00	0.00	6.07	0.00	2.50	11.07	0.00	7.50	11.43	0.00	10.71
		d	8.93	0.36	4.64	14.29	0.00	11.07	13.21	0.00	11.79	16.07	0.00	13.21
		i	9.64	0.36	4.64	15.00	0.00	11.07	13.21	0.00	11.07	16.43	0.00	12.50
	tcdb	c	1.07	0.00	0.71	13.21	0.71	8.21	22.86	0.00	21.79	24.29	0.36	24.64
		d	22.14	0.00	13.21	**26.07**	0.00	27.50	**27.50**	0.00	**27.86**	**27.50**	0.00	**28.93**
		i	**23.57**	0.00	12.14	**27.50**	0.00	25.36	**28.57**	0.36	**27.14**	**29.29**	0.00	**28.21**

Table 9.5: *Precision at 10 (as percentages) for dynamic programming matching against all channels without rests.*

is a fairly short value whose actual duration varies between pieces, depending on the speed of the music.

The similarity measures were local alignment, longest common subsequence, and longest common substring. The scores calculated for each method were normalised in three different ways: dividing by the log of the melody length plus one, dividing by the melody length, and leaving the score unchanged. This was done to determine if it was necessary to compensate for varying piece length. The melody length used for the normalisation process was defined as the number of characters in the melody string.

The experiment itself was factorially exhaustive. We combined every melody extraction technique with every standardisation method and every edit distance, over the database as well as over each of the four query lengths. We ran all possible experiments in which queries processed with one extraction technique were run against a version of the database processed with another extraction technique. Statistical significance of the results was measured using the Wilcoxon test. This design is intended to allow complete analysis of which factors are important for music similarity measurement.

9.2.2 Results

Results are shown in Tables 9.1, 9.2, 9.4, and 9.5. Table 9.1 shows the effect of varying query length, melody extraction technique, standardisation technique, song length normalisation, and similarity measure, where queries and the database are identically processed, measured by a recall-precision average. Perhaps the most important result in this table is that it demonstrates that the framework is effective: matching melodies can be found using the appropriate combination of extraction, standardisation, normalisation, and similarity measure, with, in the best case, even for short queries one in three retrieved melodies being correct (and, in the top 10 as shown in Table 9.2, thirty percent being correct). By the performance standards of document retrieval systems, these are good results.

Table 9.1 shows that contour is always the worst standardisation technique, and is almost certainly not usable in practice. Modulo and exact intervals have similar performance to each other, with modulo better for long queries and exact interval better for short queries. Only for queries of 40 notes or more does contour have any success at finding matches. Note that in many cases 40 notes is more or less the whole melody; extending to 100 notes often just introduces repetition of the theme.

Another clear result is that local alignment is the best similarity measure, followed by longest common substring. Predictably, longest common substring was about as effective as local alignment for short queries. Longest common subsequence performed poorly. Local alignment has allowed good matching with all melody extraction techniques, even for short queries. It is quite clear from these experimental results that no normalisation should be used when applying local alignment to melody matching.

Of the melody extraction techniques, the entropy-part method (the most complex technique tried) was the weakest; the others have all worked well, with all-mono and entropy-channel both giving good results. All-mono was usually the best in conjunction with local alignment. We also found that using the same melody extraction method as that used for the database melodies is always better than matching queries processed one way against a database processed in another.

Table 9.2 concerns the same experiments and variables as in Table 9.1, but using precision at 10—the number of correct answers in the first 10 pieces retrieved—instead of recall-precision. This table shows that the results are independent of the performance measure.

Table 9.3 shows the results of experiments where rests are included in the melody standardisation; in all other respects the experiments are identical to those reported in Table 9.1. Comparing these two tables, it can be seen that rests are only helpful with short queries, and the best performance with rests is not as good as the best performance without. Rests have helped contour, but not enough to make it useful.

Table 9.4 shows the results of experiments where each piece in the data set was represented several times, by the highest-note sequences from each channel, while the queries were processed as before. The performance of the channel-based extraction methods is much improved, particularly for longer queries, while the all-mono method has not worked as well. Overall results are not quite as good as the best reported in Table 9.1. The fall in performance is probably because, while use of all-channels eliminates the need to guess which channel contains the melody, the addition of "noise" channels increases the likelihood of false matching.

The difference between the best local alignment and the best longest common substring results is only statistically significant for long queries according to the Wilcoxon test. The difference between the best and worst normalisation methods was found to be statistically significant, as was the difference between the two better dynamic programming methods and longest common subsequence. A similar pattern of results was found when the experiment was run with a query set containing 46 queries, suggesting some real differences between methods. The difference between the directed modulo-12 and exact interval standardisation methods was not found to be statistically significant.

A more detailed analysis of the experimental results for 30-note queries using directed modulo-12 intervals revealed that certain queries were always unsuccessful in retrieving answers other than the piece from which they were extracted. This usually occurred when the melody was not cleanly extracted, that is, the *all-mono* method produced a melody with many accompanying notes in it. These same queries typically caused the other melody extraction methods to select a non-melody part.

9.2.3 Discussion

We argued that a minimal matching process must involve at least melody extraction, to reduce each piece to a linear sequence of notes; standardisation, to represent each piece in a form that is independent of variables such as key that do not affect similarity perception; and a similarity measure, for scoring the similarity of two pieces of music. Using a large number of polyphonic pieces in the MIDI format extracted from a public-domain collection of music, we have shown that this methodology effectively finds pieces that are similar to each other.

In particular, we have found that local alignment is the best dynamic programming technique for melody matching, and that it is best not to normalise for melody length when using this technique. The choice of melody extraction technique is important for the success of a melody matching implementation. We have found that the method that was shown by listeners to be the most effective, the all-mono technique, was also the most effective when applied to melody matching. In fact, the ranking of extraction methods is exactly the same for both experiments.

9.3 N-Gram Experiments

In the previous experiment we compared various similarity measurement, melody extraction, and melody standardisation methods. The results clearly indicated that local alignment was an effective method for similarity measurement, while *all-mono* was the best of the melody extraction methods.

			10			20			40			80		
			0	1	L	0	1	L	0	1	L	0	1	L
amdb	c	3	23.90	04.65	34.72	36.87	07.87	42.55	38.38	06.40	42.63	32.52	09.46	40.86
		4	37.30	19.73	**39.73**	44.75	29.44	**46.51**	**46.15**	24.89	**47.84**	46.66	24.51	**49.47**
		5	**37.93**	34.49	**40.09**	48.10	42.59	**48.93**	**47.47**	37.89	**49.53**	**51.50**	35.42	**50.30**
		6	36.09	36.86	**36.87**	**47.43**	46.38	**47.93**	**49.68**	41.19	**49.08**	**50.03**	38.21	**50.18**
		7	32.03	33.41	33.52	43.45	**45.10**	**44.61**	**47.44**	44.57	**47.95**	46.35	39.50	**47.45**
		8	24.09	24.74	24.81	42.61	**44.29**	**44.31**	46.21	43.90	46.09	44.60	40.90	45.53
	n	3	04.97	08.71	04.91	06.63	13.62	07.04	07.74	13.91	08.69	07.87	15.69	09.18
		4	16.40	21.69	19.98	23.41	28.59	24.96	20.52	25.73	20.98	21.27	27.29	23.20
		5	32.76	34.17	35.27	36.73	42.28	39.27	34.59	38.49	35.99	33.16	35.37	36.24
		6	**37.69**	**37.81**	**37.53**	45.20	45.81	45.81	42.93	45.74	44.09	38.09	37.28	39.22
		7	32.55	32.87	32.63	42.35	43.28	42.70	46.08	46.57	46.18	40.00	38.49	41.54
		8	24.44	24.54	24.28	42.64	**44.44**	43.00	44.13	45.03	44.91	40.31	39.54	40.58
	u	3	00.04	25.28	00.04	00.04	41.34	00.04	00.05	39.13	00.08	01.03	41.46	01.68
		4	00.04	31.23	00.04	00.04	42.95	00.04	00.05	38.23	00.06	01.03	39.06	01.67
		5	00.04	22.82	00.04	00.04	43.22	00.04	00.05	38.53	00.06	01.03	36.49	01.67
		6	00.04	12.47	00.04	00.04	39.93	00.04	00.05	37.75	00.06	01.03	37.04	01.67
		7	00.04	00.14	00.04	00.04	36.63	00.04	00.05	37.40	00.05	00.73	35.08	01.67
		8	00.04	00.06	00.04	00.04	31.23	00.04	00.05	36.81	00.05	00.71	35.04	01.67
ecdb	c	3	21.81	06.25	26.70	34.56	08.80	**36.63**	35.12	09.99	37.96	33.04	10.68	34.22
		4	**30.58**	14.94	**34.37**	**38.29**	20.36	**38.24**	38.87	20.24	**38.72**	38.45	22.51	39.32
		5	**32.32**	26.89	**34.38**	**38.59**	29.92	**38.36**	38.00	25.55	38.06	36.83	29.64	38.22
		6	28.72	26.96	28.79	**39.61**	32.48	**38.87**	38.11	30.26	38.12	38.09	31.48	37.72
		7	23.03	23.28	23.28	**37.30**	33.63	**37.46**	39.29	31.02	36.93	39.59	33.83	39.14
		8	14.77	14.70	14.70	**37.19**	35.35	**37.42**	36.86	32.20	36.72	38.12	35.15	37.67
	n	3	04.43	10.49	04.79	05.56	14.25	07.10	05.12	16.12	07.27	04.54	16.56	05.20
		4	14.61	19.19	16.54	17.11	20.67	18.85	15.24	18.95	15.94	14.91	18.90	16.78
		5	26.75	28.46	**28.83**	28.45	27.53	27.64	20.68	22.71	21.91	20.62	22.67	23.59
		6	28.34	27.44	28.05	32.66	31.99	32.84	27.73	28.46	30.12	27.98	28.72	28.63
		7	23.40	23.40	23.40	34.32	33.38	34.49	32.88	30.11	31.68	30.07	31.87	30.32
		8	15.03	14.70	15.03	35.23	35.22	35.20	32.98	32.53	32.94	32.72	32.91	32.75
	u	3	00.13	16.79	00.14	00.47	29.05	00.83	03.49	32.04	04.62	04.63	36.37	05.53
		4	00.15	21.49	00.15	00.48	33.75	00.82	03.67	34.10	04.63	04.58	36.90	05.52
		5	00.13	14.57	00.15	00.49	35.12	00.67	03.52	35.70	04.49	04.55	35.01	05.74
		6	00.12	03.71	00.12	00.48	35.98	00.50	01.86	34.39	04.66	04.52	30.01	05.58
		7	00.12	00.18	00.13	00.19	29.11	00.49	01.50	34.68	04.64	04.51	29.64	05.72
		8	00.12	00.13	00.12	00.18	21.29	00.20	01.27	32.54	04.48	04.66	29.32	05.48
tcdb	c	3	19.85	05.96	24.27	31.43	06.95	32.71	30.09	06.95	31.39	27.86	10.02	30.07
		4	27.41	13.46	30.19	32.87	14.42	34.21	33.85	16.28	33.58	34.92	20.03	36.27
		5	29.12	22.85	31.24	34.07	23.52	33.59	35.34	23.64	34.61	37.91	24.51	39.24
		6	25.71	26.38	27.93	33.85	28.91	33.84	36.61	26.30	34.83	38.59	27.40	38.89
		7	22.95	22.65	23.71	33.66	31.32	34.07	35.33	28.49	34.68	37.91	31.72	39.41
		8	21.55	21.99	21.99	33.40	31.90	33.81	36.04	30.10	34.61	37.42	33.60	38.74
	n	3	09.98	10.94	10.88	09.83	10.67	10.98	09.66	15.22	10.89	08.82	09.95	08.94
		4	18.91	15.80	19.01	18.93	19.33	19.64	17.57	21.91	19.13	16.98	19.61	19.07
		5	27.87	26.51	28.09	30.94	26.16	31.50	27.74	26.04	28.03	27.48	27.58	28.78
		6	27.76	27.03	28.05	32.89	31.30	33.78	33.04	29.84	32.93	33.08	32.50	33.92
		7	25.88	23.97	26.80	35.54	34.36	36.19	34.01	30.50	34.68	35.48	35.92	36.76
		8	24.90	23.46	24.90	34.64	33.88	34.83	34.35	31.79	34.45	36.04	36.76	36.37
	u	3	00.51	17.24	01.34	02.33	27.67	02.39	05.86	30.36	06.07	15.46	33.51	15.76
		4	00.11	19.83	01.34	02.18	31.75	02.39	05.75	31.81	06.09	15.10	33.99	16.19
		5	00.03	19.69	00.13	02.28	32.75	02.39	05.86	32.77	06.02	15.28	33.47	17.36
		6	00.02	11.95	00.05	02.07	32.78	02.49	02.83	33.73	06.12	15.07	33.24	17.13
		7	00.02	00.04	00.02	01.75	28.76	02.49	02.79	33.84	06.07	14.99	33.84	17.29
		8	00.03	00.03	00.03	01.86	24.43	02.28	02.89	33.67	06.18	15.11	33.29	17.40

Table 9.6: *Eleven-point precision averages for n-grams. The automatically generated queries were processed with the same extraction technique as the melody database. The second heading row shows the normalisation amount used, with L indicating division by the log of the length and the others being the k-th root of the length. Query lengths used are 10, 20, 40 and 80. This set used directed modulo 12 melody standardisation.*

In this experiment our aim is to determine whether n-grams will work well for the same problem, as they are an attractive matching technique. In genomics, for example, they can be used for an initial, rapid coarse search, allowing a subsequent more careful search of a smaller set of potential answers. In this experiment, using the same set of melody string extraction techniques as previously, we show that simple n-gram formulations can work very well indeed, particularly for short queries. Further, Downie [42] has reported successful experiments with n-gram matching based on classic information retrieval similarity techniques, in which a "TF-IDF" formulation was used to rank melody strings using both overall n-gram rareness in the corpus and the frequency of occurrence of the n-grams in each melody string. Our experiments reported here show that TF-IDF ranking is poor in comparison to both local alignment and simple coordinate matching with n-grams.

One of the variables that we explore in this experiment is the n-gram length. Varying n trades recall against precision—high n provides closer matching but is more likely to miss variations with omitted or additional notes. We show that, for music, the n-gram length of choice depends partly on the standardisation method. For methods that retain interval information, matching 5-grams from the query against each channel of stored pieces provides good effectiveness for all query types. Considerably longer n-grams are required for optimal matching with contour standardisation.

9.3.1 Method

For this experiment we applied the same methodology as that of our earlier experiment. In addition to the variables examined previously, we tested the effect of different n-gram lengths.

The three n-gram techniques described in Chapter 8 are tested, namely, *sum common* (**n**), in which the frequency of each n-gram that is common to both query and piece is summed, *count distinct* (**c**, also known as "coordinate matching"), in which the number of distinct n-grams that are common is counted, and the *Ukkonen measure* (**u**), which sums the difference in n-gram frequencies between the two melodies.

We also tested the TF-IDF technique that is widely used in text retrieval [152] and used by Downie [42] for music retrieval. These are written as:

***i*:** TF-IDF ranking using Equation 8.6.

***L*:** TF-IDF ranking using Equation 8.7.

			010		020		040		080	
			0	L	0	L	0	L	0	L
amdb	c	3	21.95	24.00	28.62	34.45	24.38	32.44	21.00	30.09
		4	**32.31**	**31.36**	37.67	42.04	**43.49**	**44.44**	40.80	40.75
		5	**32.05**	**32.50**	**42.67**	41.87	**48.21**	**46.66**	43.63	42.80
		6	**30.84**	**29.46**	**43.94**	41.88	**47.84**	**46.34**	42.92	43.05
		7	27.06	26.53	38.24	38.95	**43.39**	**43.22**	42.35	41.62
		8	16.77	16.82	36.39	35.76	**43.68**	**43.18**	43.54	42.04
	n	3	03.10	03.36	03.46	03.91	02.85	03.27	03.48	03.69
		4	09.66	11.30	14.00	15.50	11.62	14.08	11.51	13.33
		5	24.14	24.49	30.78	30.90	24.15	24.06	23.19	24.24
		6	**30.65**	**30.88**	37.69	38.76	35.41	37.62	30.94	31.61
		7	25.94	25.65	35.27	34.93	39.11	39.49	34.12	34.90
		8	16.79	16.41	32.63	32.41	**41.84**	**41.69**	37.53	37.87
ecdb	c	3	22.54	25.17	39.10	40.38	43.84	**42.75**	42.29	45.01
		4	**37.34**	36.31	**45.55**	43.71	**48.43**	**45.59**	**50.04**	**54.24**
		5	**37.07**	**38.01**	**49.23**	**44.98**	**48.39**	**46.81**	**52.98**	**56.30**
		6	**34.31**	**32.44**	**49.25**	**47.98**	**50.69**	**47.27**	**52.13**	**52.52**
		7	28.89	28.30	**46.04**	**47.04**	**46.67**	**46.31**	**53.30**	**54.22**
		8	20.76	20.76	**47.70**	**46.14**	**48.13**	**47.32**	**53.18**	**50.26**
	n	3	03.11	03.13	03.25	03.92	03.19	04.36	04.52	05.77
		4	10.09	10.57	12.60	12.69	13.74	14.62	13.66	13.46
		5	29.21	29.85	29.76	29.43	21.85	21.93	22.16	23.77
		6	28.75	29.76	36.36	35.42	29.56	30.04	31.12	32.42
		7	28.05	27.87	39.90	39.91	36.62	37.87	38.36	37.97
		8	22.00	21.19	44.05	43.28	40.93	41.50	40.85	40.76
tcdb	c	3	20.57	26.32	36.56	39.43	35.87	35.93	34.08	37.44
		4	**31.48**	**33.35**	**43.06**	**42.24**	**42.65**	**41.34**	40.73	42.55
		5	**32.83**	**35.69**	**42.96**	**41.76**	**42.51**	**42.55**	42.97	**45.61**
		6	**29.64**	**32.70**	**40.58**	**41.98**	**42.64**	**43.51**	43.33	**45.25**
		7	**29.71**	**29.61**	**40.64**	**42.23**	**42.14**	**43.37**	42.57	**44.79**
		8	25.86	27.24	**40.95**	**41.88**	**42.05**	**42.00**	**45.00**	**45.45**
	n	3	05.18	05.30	05.72	06.37	05.51	05.41	05.31	04.93
		4	09.92	13.96	13.13	12.57	11.76	13.39	10.10	10.06
		5	26.21	28.71	31.00	31.28	27.46	30.39	25.73	27.39
		6	**28.83**	**29.19**	37.75	36.89	37.35	36.77	35.63	35.77
		7	**28.75**	**28.25**	**39.97**	39.81	39.30	38.45	40.51	40.94
		8	25.67	26.23	**40.60**	**39.93**	38.82	38.53	40.60	40.90

Table 9.7: *Eleven-point precision averages for n-grams. The automatically generated queries were applied to the* all-channels *database. The second heading row shows the normalisation amount used, with L indicating division by the log of the length and the others being the k-th root of the length. Query lengths used are 10, 20, 40 and 80. This set used directed modulo melody standardisation.*

		010		020		040		080	
		0	L	0	L	0	L	0	L
amdb	3	00.00	00.07	00.03	00.15	01.87	00.31	00.10	00.33
	4	00.08	00.25	00.21	01.55	00.36	04.09	00.19	04.86
	5	00.56	01.25	02.74	04.08	02.80	08.55	05.60	14.02
	6	00.46	02.83	02.58	08.67	11.32	18.18	13.46	19.40
	7	00.69	**03.79**	06.90	14.77	17.28	**21.48**	18.27	22.45
	8	**04.21**	**04.63**	11.09	**17.39**	**21.38**	**23.31**	21.28	**25.90**
ecdb	3	00.17	00.19	00.03	00.57	00.41	00.84	00.07	01.59
	4	00.13	**00.52**	00.21	02.53	00.54	05.15	00.59	10.84
	5	**00.93**	**01.46**	00.93	05.79	03.47	12.38	06.42	17.64
	6	**00.71**	**01.72**	03.51	10.04	11.16	**17.56**	12.07	**20.02**
	7	**00.78**	**02.04**	07.62	**14.13**	**14.28**	**19.70**	18.42	**23.82**
	8	**01.23**	**02.22**	10.42	**17.19**	**18.99**	**23.19**	**22.30**	**24.61**
tcdb	3	00.09	00.73	00.44	01.80	00.08	02.23	00.10	02.94
	4	00.17	**01.00**	00.25	03.62	01.08	08.50	01.21	15.97
	5	**00.57**	**01.48**	01.09	07.39	04.13	14.05	08.68	18.53
	6	**00.69**	**02.28**	03.69	**11.93**	09.59	20.24	15.21	21.18
	7	**01.17**	**02.48**	**06.94**	**14.67**	14.34	22.71	18.35	**22.88**
	8	**01.41**	**03.53**	**09.84**	**16.45**	18.72	**23.76**	**21.58**	**24.49**

Table 9.8: *Eleven-point precision averages for n-grams. The automatically generated queries were processed with the same extraction technique as the melody database. This set used contour melody standardisation and coordinate matching.*

They are described fully in Chapter 8.

We used the same database of 10,466 MIDI files, and the set of 28 queries used in the previous experiment. The n-gram counting methods were varied by use of different length normalisation techniques, including no normalisation, division by the length of the melody, division by the square root, cube root, and ninth root of the length, and the more typical division by the log of the length. These length normalisation techniques have very different effects: the ninth-root method, for example, distinguishes amongst very short pieces but otherwise has little effect, whereas some of the other methods distinguish amongst pieces of all lengths.

Query lengths of 10, 20, 40 and 80 were tried on all n-gram lengths from 1 to 8. We tested all standardisation methods, however, we mainly report on experiments with the directed modulo-12 method of melody standardisation, as the pattern of results was similar to that found in the dynamic programming experiment. Query melodies used the same extraction method as those of the database against which they were being compared. We also tried matching query melodies

		010		020		040		080	
		0	L	0	L	0	L	0	L
amdb	3	00.00	00.32	00.01	00.56	00.05	00.09	00.07	00.07
	4	00.01	00.67	00.84	01.43	00.08	01.93	00.10	00.23
	5	00.53	**00.93**	00.65	02.20	00.96	03.26	00.33	01.84
	6	**00.46**	**01.21**	02.65	02.83	04.79	05.41	01.16	03.59
	7	00.53	**01.39**	03.84	05.03	05.73	08.04	02.83	04.38
	8	**00.80**	**00.97**	05.96	06.87	07.97	10.46	03.60	06.34
ecdb	3	00.04	00.04	00.04	00.25	00.05	00.55	00.05	00.81
	4	00.13	00.20	00.06	01.01	00.19	04.01	00.23	09.03
	5	00.24	00.87	00.79	06.53	02.44	10.86	05.88	19.48
	6	**00.51**	01.13	03.52	07.57	12.51	16.69	14.33	28.89
	7	**00.62**	01.48	05.87	08.47	**18.38**	**21.88**	23.66	**32.43**
	8	**00.83**	**01.88**	10.11	**11.31**	**21.96**	**27.32**	28.81	**32.79**
tcdb	3	00.00	00.33	00.01	00.74	00.06	00.84	00.04	01.00
	4	00.02	**00.42**	00.06	02.13	00.13	06.50	00.25	15.50
	5	00.54	**00.56**	00.84	06.63	01.55	16.17	03.90	21.19
	6	**00.59**	**00.89**	02.79	11.17	10.09	22.79	14.25	26.63
	7	**00.35**	**00.96**	05.61	13.57	20.01	26.01	20.74	**28.30**
	8	**00.53**	**01.42**	11.60	**15.34**	**23.25**	**27.72**	27.06	**30.60**

Table 9.9: *Eleven-point precision averages for n-grams using the* all-channels *database. This set used contour melody standardisation.*

against the "all-channel" database, which contains the extracted melody of each channel in each melody, using the "all-mono" extraction method.

Methods were evaluated by calculating eleven-point precision averages and precision at 10 pieces retrieved, but the results for the measurement techniques are largely consistent and we only report the former.

9.3.2 Results

Results comparing counting methods, normalisation, and melody extraction technique are shown in Tables 9.6 to 9.11.

Table 9.6 shows results for directed modulo-12 queries when query and database have the same melody extraction technique. Table 9.7 shows results for different melody extraction techniques against the *all-channels* database. Table 9.14 shows the results using the set of 51 queries against the *all-channels* database. Results for contour are shown in Tables 9.8 and 9.9. Exact interval standardisation experiments are shown in Tables 9.10 and 9.11 respectively. Entropy-part results were so poor in all experiments that we have chosen not to report them. N-gram lengths of one and two were uniformly worse than longer n-gram lengths, so these have also been omitted from the results tables.

These results illustrate the difficulty of choosing a "best" matching technique. The Ukkonen (u) method is generally poor, but for long queries gives acceptable performance for one kind of normalisation. (Due to these results, however, particularly on short queries, we do not consider it further.) The *sum common* (n) method may have been penalised because some n-grams, such as when the same note is repeated several times, are very common, thus favouring long pieces of music when queries contained these common n-grams. Shorter queries are less likely to contain repetitions and common n-grams, and perform about as well as long queries. Overall, coordinate matching (c) has been the most effective method, working well on the *all-mono* database and with *all-mono* and *entropy-channel* melody extraction against the *all-channels* database. It rivals the local alignment method tested in our dynamic programming experiment.

Also of note is that the best normalisation methods seem to be those that only affect the scores by a small amount, namely log normalisation, dividing by the ninth root of the melody length (not shown), and no normalisation. This is not true for the Ukkonen measure, however, which appears to perform best when divided by the melody length, making it an "error rate" measure. That the Ukkonen measure is less effective than other formulations is a similar result to that found for genomic databases [140].

The best results were achieved by the entropy-channel method using the count-distinct n-gram measure and the *all-channels* database. Other observations are that longer query lengths again lead to more relevant answers being retrieved when using count-distinct or Ukkonen measures but not when using the sum-of-frequency measure.

The tables of results also show the effect of varying the n-gram length. Lengths from $n = 5$ to $n = 7$ give the best results for coordinate matching with directed modulo-12 standardisation. N-gram length seems to vary with standardisation method, with contour requiring much longer n-grams. We report only $n = 5$ for the remainder of our experiments.

Results for TF-IDF are shown in Tables 9.12 and 9.13. TF-IDF is consistently poor compared to coordinate matching. TF-IDF formulation 1 (i), works better than *sum-common* and TF-IDF formulation 2 (L), statistically significantly for all but short queries. Interestingly, the best

		010		020		040		080	
		0	L	0	L	0	L	0	L
amdb	3	32.64	36.22	40.68	**44.02**	42.67	**48.25**	42.96	**47.27**
	4	**41.11**	**39.80**	**46.25**	**46.76**	**49.08**	48.44	46.65	**48.94**
	5	**39.99**	**40.83**	**47.34**	**47.45**	49.43	**51.00**	**47.36**	**48.65**
	6	**37.65**	**38.14**	**46.97**	**47.29**	49.07	**50.63**	46.60	**46.97**
	7	32.62	32.94	43.36	**44.31**	**48.45**	47.37	46.30	46.57
	8	24.68	24.68	40.95	**41.79**	44.91	44.80	45.73	45.96
ecdb	3	25.04	30.25	**35.06**	**35.71**	34.65	37.13	33.47	34.38
	4	**33.20**	**35.32**	**38.89**	**38.12**	39.49	39.05	38.28	38.21
	5	**33.38**	**37.09**	**39.56**	**37.21**	38.61	**39.62**	37.20	38.60
	6	29.22	**29.70**	**37.69**	**38.49**	36.61	37.83	37.68	38.51
	7	23.02	23.27	**37.30**	**37.56**	37.73	36.99	39.52	38.96
	8	14.77	14.70	**37.19**	**37.45**	36.43	36.38	37.89	38.78
tcdb	3	23.03	25.21	31.24	33.29	29.75	31.30	28.98	31.04
	4	28.83	30.74	32.41	33.62	34.41	32.91	34.10	35.78
	5	30.28	31.75	33.93	34.73	34.71	34.51	37.97	39.71
	6	26.76	28.48	32.67	33.88	34.88	33.83	39.63	39.19
	7	23.29	23.10	33.17	34.79	34.61	34.91	37.53	39.47
	8	21.55	22.11	33.03	34.08	35.67	34.89	36.88	38.87

Table 9.10: *Eleven-point precision averages for n-grams. The automatically generated queries were processed with the same extraction technique as the melody database. This set used exact interval melody standardisation.*

results for these tend to be when divided by the song-length. Regardless, coordinate matching is clearly the best approach.

The difference between the best n-gram and the best local alignment methods are not statistically significant according to the Wilcoxon test.

9.3.3 Discussion

That counting distinct n-grams has performed well in these experiments suggests that it is possible to successfully produce a useful n-gram-based index for melody retrieval. It also makes clear that term frequency is unimportant in melody retrieval. Previously published work [42] suggests that the TF-IDF technique developed in information retrieval works well for melody retrieval. Our results indicate, however, that it would not perform as well as does ignoring term frequency. Another technique often discussed [144] uses the log of the term frequency instead of the term frequency itself, to reduce the weighting of terms that are repeated. We predict that

		010		020		040		080	
		0	L	0	L	0	L	0	L
amdb	3	25.41	25.75	34.13	37.13	31.93	**37.96**	27.06	37.10
	4	30.43	30.70	38.65	**43.12**	**45.42**	**48.47**	43.79	45.31
	5	**33.19**	**32.18**	**41.45**	**41.11**	**47.06**	**47.27**	**44.56**	45.23
	6	30.53	30.98	**41.61**	**42.18**	**45.76**	**46.76**	43.03	41.76
	7	26.59	26.87	38.03	**38.25**	**43.93**	**43.84**	41.25	42.64
	8	16.12	16.66	33.81	34.14	**41.59**	**43.38**	41.76	41.80
ecdb	3	25.94	27.54	42.29	40.70	42.59	**44.29**	42.39	44.32
	4	**38.89**	**37.98**	**45.17**	**47.38**	**49.00**	**47.88**	**49.07**	**54.50**
	5	**40.44**	**40.27**	**48.88**	**49.16**	**45.77**	**47.84**	**51.90**	**56.84**
	6	**33.33**	34.06	**46.39**	**47.12**	**47.91**	**47.89**	**53.91**	**54.37**
	7	28.62	29.46	**46.43**	**47.33**	**48.02**	**47.88**	**52.91**	**54.42**
	8	20.16	20.55	**46.63**	**46.96**	**48.79**	**49.67**	**52.84**	**51.29**
tcdb	3	20.33	26.50	38.97	39.31	38.39	**36.42**	36.05	38.55
	4	29.55	33.50	**43.23**	**45.07**	**42.28**	**44.71**	41.70	43.05
	5	**35.02**	33.22	**43.23**	**45.17**	**43.01**	**44.02**	**44.24**	45.23
	6	30.16	32.04	**41.57**	**43.07**	**42.72**	**43.64**	42.87	**44.79**
	7	28.89	27.84	**41.55**	**43.13**	**44.29**	**43.13**	**42.64**	**44.53**
	8	26.74	26.91	**40.59**	**43.73**	**40.73**	**43.26**	**43.41**	**44.90**

Table 9.11: *Eleven-point precision averages for n-grams using the* all-channels *database. This set used exact interval melody standardisation.*

this also would perform less well than ignoring the term frequency.

9.4 Summary

The success of the matching requires effective extraction, standardisation, and similarity techniques. We selected a variety of such techniques for experimental evaluation, many of which had been previously proposed for this task. Combining simple melody extraction (taking the highest note starting at any time), relative pitch intervals, and local alignment gave excellent effectiveness: for even short queries, the top 10 answers contained most of the relevant answers for most queries.

Some options, however, were highly unsuccessful, finding virtually no answers at all. For example, melody contour standardisation—used in some "query by humming" systems—does not work with queries of reasonable length, and longest common subsequence is a poor similarity function. First attempts at manual queries to a music retrieval system are likely to consist of

		010		020		040		080	
		L	i	L	i	L	i	L	i
amdb	0	34.01	39.10	40.21	**43.94**	38.39	**47.26**	37.96	**47.01**
	1	35.80	**39.75**	42.94	**45.92**	44.27	**48.37**	40.20	**47.91**
	9	35.40	**40.19**	41.74	**43.66**	40.61	**46.69**	40.01	**47.58**
	L	35.40	**40.06**	41.68	**43.67**	40.86	**46.52**	40.14	**47.53**
ecdb	0	**27.96**	**28.25**	30.43	**38.61**	25.17	**39.85**	26.28	39.22
	1	**29.17**	**30.47**	29.87	**38.26**	28.32	38.20	30.92	38.03
	9	**29.63**	**29.80**	29.95	**37.55**	28.73	39.09	27.40	**40.09**
	L	**29.69**	**30.16**	30.18	**37.87**	28.85	**40.49**	27.91	**41.34**
tcdb	0	28.42	**30.86**	32.11	**35.45**	30.49	34.66	29.19	34.60
	1	27.20	29.20	27.85	31.28	27.89	32.97	30.06	33.61
	9	28.46	**30.92**	32.72	**34.80**	30.71	34.89	30.22	34.82
	L	28.41	**30.45**	32.86	**34.57**	30.93	35.02	30.48	34.66

Table 9.12: *Eleven point precision averages for two variants (*i *and* L*) of TF-IDF for n-gram melody matching with* $n = 5$*, using the set of 28 automatically extracted melody queries. Same extraction method as database.*

a phrase, and thus will usually be less than 10 notes, and therefore good performance for short queries is vital. Rests appeared to be unhelpful in matching.

Our experiments have revealed that the best approach to using n-grams for ranking melodies is one that ignores term frequency completely and uses n-grams of length five with a melody standardisation method that retains interval information. There should be either no length normalisation or very little, such as log normalisation. When used in this manner, the technique rivals local alignment in its ability to produce relevant answers to melodic queries. Local alignment itself works well with the frequently applied weights of 1 for matches, -1 for mismatches, and -2 for indels with no normalisation for song-length.

We expect that manual queries (in contrast to our queries derived from variant transcriptions) would most resemble those produced by entropy-channel and top-channel methods, which contain fewer extra notes from other parts of the composition. It follows that for manual queries the use of all channels from each composition may be valuable. The experiments reported in Chapter 10 support this hypothesis.

Some questions remain unanswered and we investigate some of them in experiments in Chapter 10. However, our results clearly answer the key question for music databases: use of the three-stage framework allows effective retrieval of music by theme. The combination of interval-

		010		020		040		080	
		L	i	L	i	L	i	L	i
amdb	0	**24.83**	**27.48**	**31.75**	**33.18**	27.24	**36.29**	25.45	36.79
	1	**23.45**	**30.05**	25.72	29.77	23.05	33.58	21.43	33.76
	9	**25.74**	**27.70**	**32.25**	**32.57**	28.28	**35.48**	26.92	36.46
	L	**25.38**	**27.97**	**31.84**	**32.80**	28.43	**35.33**	27.08	36.89
ecdb	0	29.25	**32.56**	30.14	**43.07**	24.93	**42.10**	25.75	**44.83**
	1	27.61	**29.50**	23.62	35.82	22.49	**38.53**	27.27	**46.97**
	9	30.68	**33.18**	31.15	**41.39**	25.39	**42.28**	26.58	**45.48**
	L	30.22	**33.10**	31.76	**41.76**	25.44	**42.00**	27.33	**45.69**
tcdb	0	**27.63**	**29.30**	32.04	**41.44**	33.30	**40.76**	28.25	**41.27**
	1	25.46	**28.04**	26.25	33.76	26.24	35.92	28.96	**37.76**
	9	**29.61**	**30.30**	34.10	**39.28**	32.13	**39.97**	29.99	**41.01**
	L	**28.74**	**30.46**	34.78	**39.40**	32.54	**40.37**	30.49	**41.07**

Table 9.13: *Eleven point precision averages for two variants of TF-IDF for n-gram melody matching with n=5. This uses the set of 28 automatically extracted melody queries and the* all-channels *database.*

based standardisation and either n-gram counting or local alignment matching algorithms works well. In addition, the experimental methodology we have used allows fruitful comparison between music retrieval methods.

			010		020		040		080	
			0	L	0	L	0	L	0	L
amdb	c	3	08.86	12.32	12.24	16.95	11.87	17.71	11.97	19.93
		4	13.48	14.69	19.07	21.37	21.18	25.11	22.68	24.02
		5	13.10	14.02	21.66	22.71	23.75	23.53	26.63	28.05
		6	12.60	13.81	21.53	20.88	23.83	23.03	29.03	28.36
		7	12.47	12.80	19.98	20.23	22.36	21.81	28.12	27.65
		8	10.17	10.27	18.16	17.66	21.39	21.18	27.55	26.92
	n	3	03.94	03.97	04.01	04.06	03.91	04.04	03.44	03.73
		4	06.57	06.61	06.48	06.60	06.36	06.79	06.07	06.51
		5	12.37	12.43	13.14	13.54	13.39	13.72	12.91	13.59
		6	14.28	14.13	17.55	17.96	17.19	18.13	18.63	18.59
		7	13.21	13.52	19.96	19.29	19.83	20.47	22.31	22.72
		8	10.11	10.11	17.99	18.09	21.71	22.21	23.40	24.56
ecdb	c	3	11.44	16.48	23.51	29.49	32.28	**39.50**	33.52	39.64
		4	18.02	19.45	**36.50**	35.25	**42.84**	**43.82**	44.80	45.80
		5	**19.38**	**19.90**	**37.77**	**37.63**	**44.12**	**41.87**	47.27	**49.27**
		6	**18.92**	19.87	**36.76**	**36.79**	**44.43**	**43.49**	**50.68**	**49.33**
		7	16.41	16.71	**37.09**	36.02	**44.03**	42.65	**49.49**	**48.81**
		8	14.75	14.98	**35.84**	**35.68**	43.53	41.25	48.96	48.64
	n	3	01.20	01.40	01.21	01.39	01.03	01.20	00.64	00.92
		4	08.42	09.36	04.90	05.33	05.16	06.11	04.26	05.68
		5	14.00	15.53	15.57	17.29	20.57	23.41	17.00	19.24
		6	15.32	15.67	24.29	24.83	29.08	30.25	24.18	26.05
		7	14.81	15.13	30.01	30.12	34.17	34.17	33.13	33.50
		8	15.00	15.11	33.67	**33.59**	38.01	36.59	38.48	39.27
tcdb	c	3	09.27	14.99	21.80	26.89	26.14	32.98	29.76	35.73
		4	**16.07**	**17.35**	31.50	30.93	**36.74**	35.72	40.01	40.98
		5	**17.27**	**18.48**	**34.14**	**32.48**	**37.60**	36.53	41.20	41.82
		6	**17.17**	**17.94**	**32.88**	**33.37**	**39.70**	36.62	42.59	42.18
		7	15.11	15.68	**32.90**	**32.79**	**38.90**	36.86	42.27	41.91
		8	13.68	13.89	32.73	**32.84**	**37.04**	35.90	41.85	41.42
	n	3	01.36	01.64	01.17	01.47	01.31	01.62	00.83	01.01
		4	05.74	07.18	05.01	06.11	06.40	08.12	04.64	06.26
		5	13.80	14.10	14.60	14.90	16.98	16.33	12.34	13.06
		6	**13.99**	13.99	22.94	22.93	23.44	24.03	17.76	19.77
		7	15.48	16.59	25.84	26.35	27.96	27.64	25.28	26.58
		8	14.57	14.83	30.01	29.82	28.20	28.31	29.25	30.83

Table 9.14: *Eleven-point precision averages for n-grams. The set of 51 automatically generated queries were applied to the* all-channels *database. The second heading row shows the normalisation amount used, with L indicating division by the log of the length and the others being the k-th root of the length. Query lengths used are 10, 20, 40 and 80. This set used directed modulo melody standardisation.*

Chapter 10

Experiments with Manual Queries and Judgements

In Chapter 4 we described the process of collecting manual queries and relevance judgements. In this chapter we focus on the application of these queries and judgements to the evaluation of our similarity measurement techniques. Our purpose is two-fold: to determine whether the same similarity measurement techniques are successful with manual queries; and to determine whether the method of evaluation makes a significant difference in performance of MIR techniques.

We report two experiments in this chapter. The first makes use of a manual query set to test the best of our similarity measurement techniques as determined with automatic queries and judgements. The second experiment makes use of a set of manual relevance judgements, collected in the manner described in Chapter 4. These manual relevance judgements are compared to the ones created by locating versions of pieces of music. Both sets are used to evaluate similarity measures and we discuss the differences found in the evaluations. The experiments reported in this chapter are extensions of those that have appeared elsewhere [134, **?**].

10.1 Manual Query Experiment

Our main aim in these experiments was to explore whether melody matching techniques for automatic queries were valid for manual queries. We gathered a set of manual queries by asking a musician to listen to the MIDI files of pieces that had been selected previously as the basis for automatic queries, and to play a melodic query representing the piece. This allowed us to use

		10			20			30		
		0	9	L	0	9	L	0	9	L
acdb	3	05.00	10.45	08.82	10.24	20.81	20.46	12.86	24.23	**26.78**
	4	08.78	13.33	13.52	24.46	28.94	27.18	25.55	**32.14**	**32.11**
	5	09.59	**15.04**	**14.97**	**27.76**	**30.23**	**27.84**	**33.36**	**34.63**	**35.77**
	6	**11.41**	10.77	10.77	25.07	**28.98**	**26.78**	**34.75**	**34.17**	**33.77**
	7	06.47	06.29	06.29	**28.42**	**28.04**	**26.92**	31.70	**32.51**	**31.35**
	8	04.08	04.36	04.36	26.53	**27.79**	**26.74**	**31.55**	**33.76**	**32.63**
amdb	3	02.37	05.43	05.19	02.65	06.09	06.15	02.87	05.28	06.10
	4	04.49	07.03	06.99	07.04	10.27	10.34	05.97	08.04	08.60
	5	05.16	08.63	08.49	12.39	14.11	14.05	10.83	13.11	12.70
	6	04.44	06.36	06.06	10.75	11.76	11.66	11.17	12.56	12.57
	7	03.43	04.98	04.98	10.79	10.90	10.90	10.46	10.52	10.52
	8	04.21	04.86	04.86	09.96	10.24	10.24	10.58	10.15	10.15
ecdb	3	02.70	05.42	05.32	05.30	09.35	09.34	05.25	08.54	08.70
	4	05.08	07.57	07.55	12.10	14.37	13.80	11.71	13.63	13.50
	5	06.00	**08.62**	**08.58**	13.46	15.07	15.18	16.73	17.34	17.27
	6	04.97	06.17	06.17	12.74	12.75	12.80	15.03	15.61	15.39
	7	03.99	04.84	04.84	10.98	11.35	11.35	13.61	13.83	13.90
	8	02.60	02.84	02.84	09.97	10.50	10.50	13.26	13.96	13.51
acdb	a	10.57	12.92	04.76	**35.46**	**28.92**	20.78	**36.81**	**35.02**	28.90
amdb	a	05.87	08.45	06.45	13.16	13.46	12.65	11.59	14.28	12.84
ecdb	a	06.09	08.06	05.70	15.56	14.76	12.36	14.33	14.65	13.42
tcdb	a	08.85	09.36	02.47	15.64	14.03	10.31	17.55	17.17	14.43

Table 10.1: *Eleven-point precision averages for different n-gram lengths (shown in column 2) using a set of 30 manually-produced melody queries and the "count distinct" measure. Also shown is local alignment (where column 2 contains "a"). The best two results for each query length are highlighted.*

the same relevance judgements as before. Using these queries we re-evaluated the extraction, standardisation, and matching techniques.

Our experiment was to produce comparable results for a set of 30 manual queries, created by a volunteer who listened to each selected piece in turn and created a melody query by playing on a synthesiser keyboard connected via MIDI to a sequencing program. There was overlap in the pieces represented in the manual set of queries and the two automatic sets used in the experiments reported in Chapter 9. Where this occurred, the same MIDI file was used as the basis of the query.

The databases that were used for searching were the same as those tested in Chapter 9, that is, the collection of 10,466 MIDI files, and each set of extracted melodies using the four extraction algorithms discussed earlier, plus the *all-channels* version of the database. We restricted our experiment to modulo-12 melody standardisation, which was shown to work well in the experiments reported earlier. We also tested three levels of normalisation.

The manual queries ranged from 15 to 88 notes. The mean length was 31.9 and the median 28. We ran the queries in two ways: truncating the queries to 10, 20, or 30 notes, or using the full manual query. In the case of truncated queries, some are shorter than the nominated length, in which case the entire query was used. Thus, all queries used in the length 10 query set are indeed of length 10; all but 5 queries were truncated to length 20, the others being in the range 15-19; and approximately half of the queries in the length 30 set were truncated. We chose to include queries of length 30 as this length falls midway between the mean and median for the untruncated manual queries in the set.

10.1.1 Results

Tables 10.1, 10.2, and 10.3 show the results for manual queries. The results when using full manual queries, as opposed to queries truncated to 10, 20, or 30 notes, are shown in Table 10.3. Coordinate matching appears to be superior to all other methods when applied to full manual queries and those that have been truncated to 10 notes. Local alignment worked best for queries truncated to 20 or 30 notes, however, this result is not statistically significant. It is also clear from these results that all-channels is the best choice for the database, and n-gram lengths from five to eight are significantly better than shorter ones for most query lengths and normalisations.

For full queries, all-channels is, again, far better than the other database versions. While the count-distinct method appears to have outperformed local alignment, the difference is not statistically significant. N-gram lengths from five to seven gave the best results, with n-grams of length five being statistically significantly better than length four and shorter.

Unlike the automatic queries used in Chapter 9, applying longest common substring to manual queries gave poor results. Table 10.2 shows the results for longest common substring and the *thresholded-substring* variation, which only gives a score for substrings of length k or greater. As with the other methods, using the all-channels database worked the best.

The *top-channel* and *entropy-part* melody extraction methods were more effective for manual queries than for automatic ones, with top-channel occasionally producing better results than entropy-channel when used with the substring measurements.

10.1.2 Analysis

On close analysis of the best n-gram and dynamic programming techniques for modulo-12 intervals, we observed that all relevant answers retrieved within the top twenty by the local alignment technique were also retrieved by n-gram coordinate matching. The relevant answers retrieved by coordinate matching but not by the dynamic programming approach tended to have significant sections of the query matching widely spaced sections of the answer. Upon examination of these answers, the reason seems to be that the query melody partially matches different variations of the same theme within the answer melody. The coordinate matching n-gram method allows these partial matches to be accumulated, without including repeated fragments. We have already seen that, when repetition is included, the precision of answers is reduced.

Another interesting observation was that the number of distinct n-grams of length five for the all-mono approach was 50% greater than that for all-channels. Entropy-channel had approximately 60% of the n-grams of all-channels. This suggests that there is little penalty and much to gain from using the all-channels approach for melody indexing.

A clear result of this experiment is that some techniques do not work as well for manual queries as they do for automatic ones. This is especially true of the longest common substring technique. This result makes sense when we consider how the automatic queries were created: by automatically extracting the melody from a MIDI file in the collection. There are likely to be several exact matches in the relevance set as a result. For manual queries, however, minor differences between the query and the data in the collection are found. As discovered in our analysis, this typically involves the query containing melodic variations that occur at different parts of the target piece. Local alignment results were similarly affected, but not to the same extent. The all-mono database also has more success with automatic queries than with manual ones.

It can be seen that the best n-gram length is five. We hypothesised that part of the success of the 5-gram coordinate matching technique is that it screens out melodies with short matches. If this were so, then a technique that only allowed matches of a minimum length of five should

		10			20			30		
		0	9	L	0	9	L	0	9	L
acdb	1	**11.48**	13.13	04.74	25.76	24.66	18.35	**30.48**	29.65	21.74
	2	**11.48**	13.75	07.69	25.76	25.00	19.74	**30.48**	**31.82**	24.53
	3	**11.39**	**14.87**	11.66	25.76	26.41	21.64	**30.48**	**31.74**	25.54
	4	11.19	**14.33**	**13.72**	25.71	**27.03**	25.62	**30.61**	**30.84**	29.31
	5	09.61	**14.87**	**14.80**	**26.51**	26.91	25.43	**33.16**	**31.00**	30.18
	6	**11.49**	**10.77**	**10.77**	26.92	**28.53**	26.46	**30.73**	**32.66**	**31.14**
	7	06.47	06.29	06.29	**28.31**	**28.00**	**26.95**	**30.88**	**32.42**	**30.74**
	8	04.08	04.36	04.36	**27.77**	**27.89**	**26.84**	**30.11**	**33.72**	**32.67**
amdb	1	06.16	08.16	06.05	11.54	12.26	11.99	11.72	12.70	11.90
	2	06.16	08.52	07.52	11.54	12.57	12.26	11.72	12.04	12.22
	3	06.11	09.15	08.41	11.51	12.83	12.37	11.72	12.55	12.00
	4	06.80	**09.44**	**09.35**	12.08	12.97	12.51	12.06	12.45	12.46
	5	05.95	09.12	08.97	13.00	**12.06**	12.89	11.41	12.09	12.15
	6	06.20	06.86	06.36	10.79	11.26	11.26	11.92	12.01	12.01
	7	03.43	04.98	04.98	11.10	11.01	11.01	10.88	10.52	10.62
	8	04.21	04.86	04.86	10.59	10.14	10.14	10.13	10.26	10.26
ecdb	1	06.16	08.08	05.72	11.99	13.04	11.72	13.92	13.88	12.94
	2	06.16	**08.77**	07.25	11.99	13.07	12.55	13.92	14.03	13.27
	3	05.86	**08.84**	**08.36**	**12.02**	13.02	13.30	13.92	14.01	13.91
	4	06.00	**08.87**	**08.78**	11.74	12.76	13.04	14.49	14.31	13.96
	5	06.30	**08.75**	**08.75**	12.59	12.73	12.99	13.91	14.91	14.06
	6	04.85	06.18	06.18	11.99	12.88	12.87	15.00	14.55	14.40
	7	03.99	04.84	04.84	11.17	11.66	11.66	13.73	13.86	13.79
	8	02.60	02.84	02.84	10.17	**10.30**	10.30	12.93	13.26	13.26
tcdb	1	07.92	09.07	02.74	13.15	13.59	10.09	16.65	15.38	13.41
	2	07.92	09.26	06.23	13.15	13.42	10.43	16.65	14.92	14.27
	3	**09.44**	**10.49**	07.14	13.23	13.08	12.05	16.64	15.83	14.54
	4	07.65	09.56	08.91	13.42	14.03	12.91	16.10	14.64	15.46
	5	**08.40**	**09.60**	**09.45**	14.30	12.58	12.76	15.03	14.80	14.70
	6	06.84	07.79	07.79	14.98	14.71	13.70	16.15	16.85	15.74
	7	03.02	03.63	03.63	11.50	13.07	11.95	14.26	15.71	14.60
	8	01.40	01.74	01.74	13.02	13.55	12.44	14.43	15.56	14.45

Table 10.2: *Eleven-point precision averages for longest common substring and thresholded substring measurements using a set of 30 manually-produced melody queries truncated to lengths 10, 20 and 30, and standardised using directed modulo 12 standardisation. Column 2 shows the minimum substring length for which a score is given. A minimum of one is equivalent to the conventional longest common substring technique.*

		acdb			amdb			ecdb			tcdb		
		0	9	L	0	9	L	0	9	L	0	9	L
c	3	13.17	27.79	28.15	02.91	04.93	05.83	05.37	09.33	10.34	07.77	13.51	15.53
	4	31.05	34.03	34.40	07.80	13.57	13.28	11.86	14.69	15.07	16.01	17.88	16.66
	5	**39.31**	**41.02**	37.99	11.96	15.41	15.42	17.13	18.55	17.90	19.02	20.73	19.44
	6	**37.44**	**36.35**	**36.04**	10.29	11.71	11.61	15.58	16.08	15.66	18.76	18.92	18.88
	7	**35.65**	**35.86**	35.97	08.90	09.15	09.24	13.98	13.79	14.00	16.43	17.61	17.72
	8	**33.23**	34.50	34.21	09.06	09.31	09.28	13.94	13.93	13.96	16.53	18.71	18.59
a	0	**34.98**	**32.56**	29.61	09.30	09.84	08.77	15.86	14.92	13.61	19.89	19.85	17.25

Table 10.3: *Eleven-point precision averages for local alignment using a set of 30 complete manually-produced melody queries. The headings l, 9 and 0 refer to the type of normalisation applied to the similarity scores. Coordinate matching is shown for n-gram lengths from three to eight.*

perform well. This does not appear to be true for substring matches, but may well work for local alignment. Combining the minimum length for contributing matches with a low penalty for larger gaps may be beneficial. We are currently exploring these possibilities.

10.2 Manual Relevance Experiment

From the start of our investigation of MIR in 1997, we have focused on identification of which techniques would be most effective for melody extraction, melody similarity matching, and so on. The approach we have used has been based on techniques found in the field of IR research. In this experiment we compare the manual and automatic queries with our two sets of relevance judgements.

The automatic and manual queries concern the same pieces of music, so we can measure our matching techniques using each possible combination of automatic and manual queries and judgements, calculating the eleven-point precision averages for each run using a pool of 1000 answers.

We examined the local alignment technique against each possible melody extraction technique. Queries were truncated to lengths 10, 20, and 30. As with the previous experiment, in some cases the manual query was shorter than the designated query length, in which case the entire query was used.

We also did a length-matched version of the experiment, in which the automatic queries were truncated to the length of the full manual query.

10.2.1 Results

Table 10.4 shows the eleven-point precision averages as a percentage for the various combinations of manual and automatic queries and judgements, for two different styles of local alignment. For each combination of query and judgement sets and query length, the best result is shown in bold; for example, it can be seen for both manual queries and automatic queries that the agreement between the two sets of judgements is excellent — both sets of judgements identify the same best method in each case.

It can also be seen that, when manual queries are used, the all-channels (ac) database outperforms the others by a large measure. The difference between all-channels and the all-mono (am) database are even more pronounced when manual judgements are used instead of automatic judgements. When the manual judgements are used, entropy-channel (ec) automatic query results are favoured, and entropy-part (ep) also shows significant improvement.

Table 10.5 shows the results for length-matched queries. The results are consistent with Table 10.4. Interestingly, the manual queries that were truncated to 30 notes were measured to be more effective on average than full manual queries. Automatic queries, however, benefited slightly from being truncated to the same length as the corresponding manual query.

We tested the results for statistical significance by applying the Wilcoxon test. The best method for each query length was significantly better to a 99% confidence level when compared to most of the other methods with the same query length in the same table. In the case of manual queries of length 20 and 30, the all-channel (ac) method was significantly better than all the other methods listed. For short automatic queries with manual relevance judgements the results were less clear. In general, the manual judgements allowed greater discrimination for manual queries and the automatic judgements discriminated better between automatic query tests.

We observed similar effects for coordinate matching, when the different combinations of automatic and manual queries and judgements were tested. The results were completely consistent with those for local alignment, so we have chosen to not include them.

Overall, the two sets of judgements showed good agreement in their ranking of matching and melody extraction techniques. We observe decidedly different results from the two query sets. For example, with the automatic queries, the all-channels (ac) database was no better than the others, in which a specific part is used to represent each piece in the database. However, for the manual queries, there is a marked difference in the success of the two approaches, with the all-channels approach being far better.

10.2.2 Discussion

The results of this experiment show that the two sets of relevance judgements give fairly good agreement when applied to the task of evaluating melody ranking techniques. There are slight differences, for example the automatic relevance favours all-mono and using the manual set of relevance judgements improves the measured effectiveness of entropy-part. The most marked difference occurs between manual and automatic queries, with the all-channels database shown to be far more effective for manual queries.

Automatic and manual queries differ in their nature. The automatic melodies that were used as a basis for automatic queries were created by applying a melody extraction technique to the entire piece. In contrast, the manual queries consisted of a sequence of notes considered to be a sufficiently large portion of the melody for it to be uniquely identified, as judged by a volunteer musician. Truncating the automatic queries to match the length of each manual query was the best that could be achieved in regard to making an unbiased comparison of the effects the two types of query have on a music matching technique. The results of the experiments that truncated both the automatic and manual queries to specific lengths agree with those where each automatic query is truncated to the same length as the corresponding full manual query. However, the results show that more is not always better for melody queries, as some gave worse results when longer queries were used.

10.3 Summary

We have proposed a methodology for matching pieces of music according to whether they are likely to be perceived as similar by a listener. We applied the sets of manual and automatic queries and judgements we have collected to the music matching techniques that we have implemented. When manual queries are used, we have found that retaining the melody from each channel, what we call the *all-channels* technique, is far better than any other approach. The fact that n-grams work so well means that we should be able to build indexes that are effective at rapidly producing useful answers to queries. If indexes are built based on the directed-modulo

			10			20			30		
			0	9	L	0	9	L	0	9	L
auto	mndb	acdb	10.57	12.92	04.76	**35.46**	**28.92**	20.78	**36.81**	**35.02**	**28.90**
		amdb	05.87	08.45	06.45	13.16	13.46	12.65	11.59	14.28	12.84
		ecdb	06.09	08.06	05.70	15.56	14.76	12.36	14.33	14.65	13.42
		epdb	04.86	04.99	04.08	08.08	08.47	07.66	07.96	07.98	07.52
		tcdb	08.85	09.36	02.47	15.64	14.03	10.31	17.55	17.17	14.43
	amdb	acdb	14.49	11.81	06.71	22.62	22.47	20.85	28.04	27.32	20.21
		amdb	**28.43**	**28.91**	25.21	**36.22**	**37.58**	**36.93**	**37.96**	**37.32**	**36.42**
	ecdb	acdb	**22.36**	18.62	12.08	**36.78**	**36.33**	28.93	**43.22**	**40.77**	35.91
		ecdb	21.05	20.39	16.02	**33.15**	**33.07**	**30.04**	**35.59**	33.62	32.53
	epdb	acdb	05.46	05.88	04.28	18.95	17.93	10.37	18.72	18.51	12.29
		epdb	13.06	13.01	11.21	20.47	19.98	18.98	19.83	21.50	19.18
	tcdb	acdb	20.89	19.74	14.40	**34.55**	**32.41**	25.51	**35.88**	34.29	28.03
		tcdb	**21.41**	20.79	16.40	**32.22**	**31.10**	25.73	**32.79**	**32.94**	29.70
man	mndb	acdb	14.81	15.49	06.31	**45.63**	**35.83**	27.70	**50.26**	**50.13**	39.30
		amdb	05.63	06.61	05.04	10.13	11.65	10.26	09.13	11.45	10.25
		ecdb	13.21	15.44	08.48	27.31	27.46	20.50	28.60	29.37	26.80
		epdb	07.64	07.75	05.49	14.79	14.01	10.90	15.14	15.51	14.05
		tcdb	11.58	12.30	02.87	15.05	19.09	13.33	22.30	23.23	18.99
	amdb	acdb	07.18	07.26	04.65	**12.33**	**11.74**	10.36	**16.30**	**15.34**	**09.55**
		amdb	**25.65**	**25.75**	23.04	**34.16**	**32.49**	**29.22**	**37.35**	**36.76**	**35.45**
	ecdb	acdb	**22.42**	18.04	15.53	**41.42**	**38.92**	**28.47**	**46.89**	**45.00**	**39.77**
		ecdb	25.16	24.57	20.71	**37.28**	**39.22**	**33.38**	**41.40**	**38.53**	**39.63**
	epdb	acdb	09.36	09.31	08.74	28.11	25.54	16.25	30.10	31.69	23.04
		epdb	17.80	17.29	18.03	**31.97**	**28.86**	28.11	31.38	**33.89**	30.72
	tcdb	acdb	19.27	19.34	13.97	**33.87**	**33.35**	**24.92**	**34.29**	**35.07**	**28.47**
		tcdb	**17.80**	18.13	18.26	**32.53**	**31.60**	**26.69**	**35.77**	**33.72**	**29.65**

Table 10.4: *Eleven-point precision averages for queries of lengths 10 to 30, comparing automatic and manual relevance judgements and queries. The first column indicates which relevance judgements were used (automatic or manual). The second column shows which type of query was used: manual (mn),* all-mono *(am),* entropy-channel *(ec),* top-channel *(tc), and* entropy-part *(ep). Column three shows the database used, including the* all-channels *(ac) database, and those based on each of the melody extraction techniques.*

		auto			man		
		0	9	L	0	9	L
mndb	acdb	**34.98**	**32.56**	29.61	**47.07**	**43.96**	39.03
	amdb	09.30	09.84	08.77	07.67	07.54	06.09
	ecdb	15.86	14.92	13.61	**29.84**	29.86	26.38
	epdb	09.35	09.35	07.00	16.95	16.61	13.97
	tcdb	19.89	19.85	17.25	23.58	22.84	23.85
amdb	acdb	30.47	31.06	24.95	21.74	20.51	19.89
	amdb	**39.88**	**40.09**	**38.30**	**38.20**	**37.96**	**33.68**
ecdb	acdb	**44.62**	**44.26**	36.28	**47.85**	**46.55**	40.99
	ecdb	33.97	33.86	33.21	**38.33**	**39.50**	39.92
epdb	acdb	18.81	18.81	13.24	29.57	29.20	21.15
	epdb	21.57	20.52	20.91	32.12	29.69	30.04
tcdb	acdb	**36.31**	34.91	30.35	**37.52**	**38.14**	29.46
	tcdb	32.76	**33.88**	29.29	**33.78**	**35.06**	29.51

Table 10.5: *Eleven-point precision averages for automatic and manual queries of matched lengths, comparing automatic and manual relevance judgements and queries. The first column shows which type of query was used: manual (mn),* all-mono *(am),* entropy-channel *(ec),* top-channel *(tc), and* entropy-part *(ep). Column two shows the database used, including the* all-channels *(ac) database, and those based on each of the melody extraction techniques.*

standardisation, we recommend indexing n-grams length of five.

That the choice of query set for evaluating melody matching methods affects the outcome highlights the fact that it is important to have appropriate tools for evaluation. Purely using statistical reasoning and known-item searches (described in Section 4.2) may lead to misleading results. We have developed software for acquiring human relevance judgements for music similarity. The manual relevance judgements acquired with our software are a valuable addition to the music collection, manual queries, automatic queries, and automatic relevance judgements that we have already collected. Together these resources provide a trustworthy tool for measuring the effectiveness of MIR systems. Our illustrative experiments show how these tools can be used in practice, discriminating between different similarity measures, melody extraction techniques and definitions of relevance.

Now that MIR technology is beginning to mature and to be available outside the research benchtop, it is essential that MIR systems to be evaluated in a consistent and rigorous manner.

We believe that the tools we have developed can provide an essential resource for the MIR community, and we plan to make our data sets and software available to the music retrieval community for standardised evaluation of systems.

Chapter 11

Future Work and Conclusions

Before our research commenced in 1997 there was little published work on MIR. Some query-by-humming systems had been built [54, 71, 93] but there had been little evaluation of their effectiveness. Kageyama et al. [71] used known-item searches to evaluate several techniques, but most work either used statistical measures of likely effectiveness [11, 54, 93] or only addressed efficiency issues [21]. Similarly, the approaches to melody extraction, where applicable, were partially documented and largely unsubstantiated [11, 54]. Our main contribution has been to apply a more rigorous methodology to the evaluation of the components of MIR systems, to develop and test some new MIR techniques, and to produce tools to assist in the evaluation of other similar systems. In this chapter we examine potential extensions to our work and review the contributions of this thesis.

11.1 Future Work

There were several aspects to our research work—melody extraction, melodic similarity measurement, and methodology for evaluating effectiveness of MIR techniques—but there is much that remains to be done in this area of research. In this section, we discuss each of the main areas of our research in turn in terms of questions that still need to be answered, and suggest research questions that could be explored by experts in the discipline of music psychology.

Music Psychology

In Chapter 2 we surveyed music psychology research to identify features that affect music similarity perception and melody perception. The facts gleaned from the survey were developed into two hierarchies of melodic similarity—one for long-term memory and the other for short-term memory. Information on perceived melody was somewhat more scant. Francès [51] stated that the musical part with the highest pitch is usually perceived as the melody except when it is repetitive, but to our knowledge there has been no previous experimental work to confirm this. Our own experiment on melody extraction techniques did so in part, but did not isolate from the experiment other factors such as rhythm and loudness that would be necessary in the context of music perception research.

In terms of perceived melodic similarity, it was clear from our survey that the results for long-term memory are quite different to those for short-term memory. Other factors that influence perception include the duration of the melodies played to listeners [47] and the musical ability of the listener [51]. When constructing the hierarchies we found that there were some types of melodic transformation that had not been tested empirically, preventing one of the hierarchies from being a simple linear list. For example, we don't know how similar melodies with the same contour and relative interval sizes compare to those with the same contour and note names but not necessarily the same relative interval sizes. Also, most work in music psychology on melodic similarity deals only with pitch. More work on how rhythm affects perception of similarity would be useful.

Research into music memory has shown that pitch sequences are more important than rhythm sequences for allowing listeners to identify melodies. Another interesting question that remains to be answered is how difficult is it for listeners to identify a melody with the rhythm changed. In our experience this seems to be quite difficult. Whether such a melody consisting of the same intervals as a given piece of music would be judged as less similar than another that has the same contour and rhythm would be of interest, as this would influence the choice of ranking techniques for a MIR system.

Improving Melody Extraction

In Chapter 5, we presented four algorithms for extracting melodies. It was found that the simplest of these was the most reliable, as judged by a group of volunteers listening to a small collection of pieces of music. The algorithm that was judged the most successful (all-mono) selects the highest pitch note commencing at every instant. In our later experiments on similarity, it was found that keeping a melodic part from each separate channel for matching was a successful approach to melody matching. This alleviates the need for selecting which instrument has the melody and thus should improve the precision of melodies that are extracted. However, with better melody extraction algorithms the situation could be reversed.

It was found in our experiments that entropy and pitch are valuable for identifying the melody of a piece. This confirms the perception concepts put forward by Francès [51]. In the selection process it may be useful to combine the two aspects. For example, if a channel melody has an entropy that is out of the accepted range for a melody, implying that it is too monotonous or too random, then it will not be selected. The highest average pitch is sometimes a successful metric for selecting the melody from a set of channel melodies, but some channels contain a part that is very short and therefore could not be the melody of the piece. A comparison of channel melody duration with the duration of the piece of music may eliminate some of these distractors.

In the case of part-splitting, our initial approach used two types of information: the overlap of parts and proximity of pitch. This approach performed poorly compared to the other techniques, but had some inbuilt limitations: the order of note events that occurred at the same instant affected the allocation to parts. We hope to develop a new technique that removes this problem and in addition considers all the main factors: average pitch, entropy and part duration. A duration check can be helpful in removing from the selection process parts that have few notes in them or widely spaced ones.

Sometimes a melody moves from instrument to instrument. This is a common feature in jazz, where each instrument takes the lead in turn. It also appears in classical and popular music, for example, in a duet. An approach to melody extraction to better handle this situation may be a sliding window technique that determines what is likely to be part of the melody on a local basis.

Using Other Music Features

Most of our experiments thus far have restricted melodic standardisation to pitch only. On a database of 10,466 MIDI files, pitch interval sequences were enough to retrieve answers well.

For larger databases more detail may be required to rank answers. It may also be important to ensure that the best answers to a query are in the top five to ten answers retrieved, due to the time it takes a user to listen to answers and determine if they are relevant. A way of reducing the number of irrelevant answers would be to include rhythm information. We experimented with the inclusion of rests, which helps to indicate breaks between phrases, and distinctive staccato rhythms, such as that found in the Addams Family theme-song. In our experiments the use of rests was unsuccessful, but varying the minimum duration of a rest may yield better results. Storing further rhythm information such as the duration of notes should help to remove answers that, upon listening, bear little resemblance to the query melody despite a similar pattern of pitches.

We have limited our research to the problem of answering melody queries. This is not the only type of query that may be posed of a MIR system. There are many unanswered questions regarding the handling of queries on tonality, harmonic progression, and other musical features.

Improvements to Similarity Measurement

We have concentrated on two specific techniques to measuring similarity: the use of n-grams and dynamic programming. These techniques are combined with melody extraction and standardisation methods. We restricted the scope of our experiments to measuring similarity using standardisation techniques based on note transitions. While some of the techniques were shown to be effective, a single note error results in two adjacent symbols being incorrect, whereas a single incorrect symbol in a string causes a change in pitch relative to the start of the match—unless it differs by an octave —which can result in a somewhat better rank than is warranted. An approach that would be worth exploring is the modification of the local alignment matching process so that mismatches up to k in length are checked to see if they retain the key of the earlier part of the melody fragment being matched. The weights applied in the matching process then allow discrimination between two types of mismatches.

One of the difficulties in the field of MIR is that there are no standard collections universally available at present. Another is that similarity measurement techniques that can be used on one collection cannot be easily adapted to another. For example, the dynamic programming approach used by Mongeau and Sankoff made use of sets of musical variations that were in a single key, and so allowed a melody standardisation method that was relative to the key of the piece. Their dynamic programming approach was specifically designed to work with this standardisation method, an approach that becomes impossible when a wide range of music, some of which has complex key changes, is used for evaluation. In this case it becomes impossible to compare the methods for effectiveness. Therefore a problem that needs to still be addressed is the evaluation of these existing techniques in a uniform framework.

Improving the Experimental Framework

We have developed an effective experimental framework for testing melody retrieval methods, however it is hoped that in future there will be more comprehensive sets of data. In particular, it would be valuable to gather better quality relevance judgements for a larger set of queries, to allow greater discrimination between ranking techniques than was possible with the sets used for our experiments. As better techniques are found, the size of these sets will be crucial to obtaining statistically meaningful results. Also important for future MIR work will be a means of comparing not only the retrieval effectiveness but other measures such as space and time efficiency so that implementers can evaluate the trade-offs between different approaches.

For more realistic evaluation of MIR techniques it would be useful to obtain real queries. There are collections of these in archives which, it is hoped, can be made available to the research community for analysis and application to evaluation. It would help answer such questions as how long and how precise melody queries are, and thus inform MIR researchers of how to best proceed with the development of new techniques.

Other Areas

In this work we have mainly concentrated on solving the problem of finding matches to melody queries. We have concentrated on the techniques required to select the parts of music that are most useful for matching, and to carry out the matching process. We have not explored efficiency issues to any great depth. We have only briefly touched on the problems of presenting answers to users. As discovered in the process of collecting music relevance judgements, there are real issues with the method of presenting answers in a way that allows users to efficiently and easily determine if the answers are relevant.

This thesis has used note-based musical information as its data set. The techniques that are successful for this type of data are useful for indexing MIDI files, and other note-based data such as the collections of music in abc, Humdrum, and DARMS formats. However, the majority of music is not available in a digital note-based format at all. Some music is available as sheet music, a form that can be converted to notes using Optical Music Recognition (OMR) techniques [5, 22]. However most music is only available as recordings, for which a different approach is required. In addition, some contemporary music is less note-oriented and would not be well-serviced by a note-based approach. Therefore the ideal MIR system would need to incorporate other non-note-based techniques.

11.2 Summary and Conclusions

Within the MIR research field we have chosen to concentrate on answering questions related to the problem of posing melody fragment queries to music databases. We hope that the influence of our contribution extends beyond this domain, as our work in evaluation of retrieval effectiveness is applicable across a broader spectrum of retrieval problems. In this section we discuss our conclusions and the contribution of this thesis.

Methodology

Our approach to the MIR problem began with a survey of music perception research presented in Chapter 2, from which we discovered that important factors for determining the melody of a polyphonic piece of music are the pitch and variety. Loudness is only one of several features that are important for determining the melody of a piece. For grouping of notes into musical parts, the most important principle is pitch proximity. In addition we learned that contour was the melodic feature that people remember best. Two hierarchies of similarity were synthesised from various pieces of music perception research. One is based on short-term memory and the other on long-term memory. It is likely that the long-term memory hierarchy will be more applicable for answering user queries to a MIR system.

In addition to a grounding in music perception, the methodologies of the field of information retrieval were an important basis for this research. This became the approach of our experimental work. As outlined in Chapter 4 our approach to the music matching problem has consisted of three stages: melody extraction, melody standardisation, and similarity measurement. The emphasis of our work has been on evaluation of techniques, rather than the building of systems.

We commenced with melody extraction algorithms based on music perception concepts. To evaluate the techniques we used human judgements. Our methodology for the music similarity measurement experiments followed the IR tradition of using a set of relevance judgements to evaluate the effectiveness of ranking techniques. There were no such relevance sets available prior to our work. We created our first query and relevance set by locating pieces of music for which there were more than one version in the collection of music. Versions were located by examining filenames and listening. Queries were created by applying the melody extraction techniques to query pieces, and truncating to specific lengths. This query and relevance set was used as a first approach to measuring the success of ranking techniques.

The next stage involved using this relevance set and the top answers of the most successful ranking techniques (as judged by measuring recall-precision averages for the query and relevance set) to obtain human judgements of similarity.

Experiments with the various sets revealed that the type of melody query used for evaluating ranking techniques makes a significant difference. Automatically generated melody queries provided more exact matching with pieces in the collection, favouring techniques that are probably less effective for real user queries. The two relevance sets had minor differences in their effects on evaluation of ranking techniques. The experiments lead us to conclude that the kind of query and relevance sets used for evaluation should resemble as closely as possible those that the algorithm is being designed for.

Extraction

Prior to our research there had been no published work on testing algorithms that extract melodies from pieces of music, but only research on splitting polyphonic music into its parts and some partially documented information on how a particular system extracted melodies [54]. In Chapter 5 we discussed four simple melody extraction algorithms that we had developed. They incorporated the concepts of relative pitch, variety — measured using first-order predictive entropy — and pitch proximity. Using listeners to evaluate the melodies that were generated by the four algorithms, we found that the most successful algorithm was also the simplest one, which we called *all-mono*. The all-mono technique places all notes into one channel and selects the highest-pitch note that starts at each instant. Thus it was found through experimental evidence that the best technique used pitch alone to determine the melody, providing some confirmation of Francès's theory [51].

Standardisation

There are many possible ways of representing a melody for matching. Several researchers have discussed different methods of representing melodies in the context of uniqueness within a collection [11, 40, 93] or in terms of the accuracy of sung queries [83]. In work concurrent with our own, Downie [41] has formally presented a set of classifications of melody representations based on pitch alone. Another detailed discussion is found in Selfridge-Field's 1998 article [120] that addresses many of the problems associated with melodic representation. In particular, the author discusses capturing information about key as well as pitch. In Chapter 6 we listed and discussed the many approaches that have been used for melody matching, plus several other possibilities that could be useful. We discussed these in terms of their advantages and disadvantages for melody matching.

Data

There have been analyses of collections of music published in the past, for example, Dowling's tally of interval frequencies in a collection of 80 Appalachian songs [36]. In Chapter 7 our focus was slightly different to that of the musicologist or psychologist. Our aim was to learn more about the distribution of notes and intervals in melodies to allow a more informed approach to music matching. We used the n-gram as the basic melodic term to analyse. We graphed the frequency distribution of n-gram occurrences in several collections of MIDI files. We used both contour representation and exact interval representation for this process.

Through our analysis, we discovered a distribution that bears broad similarity to that found in n-gram distributions of textual data, with one minor difference: there is one term that is always far more frequent than all others in the music collection, and that is the one that represents a sequence of repeated notes. This effect was exaggerated in this study as "melodies" from accompanying parts were included in the analysis. We determined the expected number of exact answers that would be retrieved given contour n-gram queries of different lengths. From this we concluded that contour queries would need to contain at least 12 notes to retrieve less than 10 answers from a set of 2697 tracks. If inexact matches are allowed—as is essential for MIR—then queries need to be much longer.

Similarity

In Chapters 8 and 9 we presented the techniques that we explored for measuring the similarity of melodies. We developed several variations on dynamic programming, and tried a few formulations that use n-grams as their basis.

Using eleven-point precision averages and our set of "automatic" queries and relevance judgements to evaluate experimental results, we discovered that local alignment and longest common substring were the most successful dynamic programming techniques for matching melodies. These were best left un-normalised for song length. The best choice for the database to match against in these cases was either the all-mono database, with the same method used for the query, or the all-channels database with the entropy-channel method used for queries.

For n-grams we discovered that the best technique, when measured in the same way as for the dynamic programming experiments, was to use what is commonly known as coordinate matching, that is, counting how many n-grams both the query and piece have in common, ignoring any duplicates. For this technique, either no normalisation or a small amount of normalisation for length were best. Again the two best databases and melody extraction techniques were entropy-channel melody extraction combined with the all-channels database, and the all-mono technique used for both the melody and the database. The best n-gram lengths were in the range five to seven.

We used a set of manual queries and human relevance judgements for further experiments. These were described in Chapter 10. We discovered that the type of query set significantly affects the success of different ranking techniques. In particular, the all-mono database was not as successful for manual queries compared to the all-channels database. Longest common substring did not appear to work as well either. Manual relevance judgements had more subtle effects on the evaluation of techniques, such as improving the measured effectiveness of melody extraction techniques that split pieces into parts before selecting the melody.

Implications

There have been many approaches suggested for MIR systems with few of them evaluated for both effectiveness and efficiency. Our work has tested the effectiveness of a small selection of techniques. Based on our experience we conclude that a practical approach to MIR for note-based information such as MIDI would extract melodies from each channel or defined musical part in the collection. The technique for melody extraction would select the highest pitch note occurring at each instant. A melody standardisation approach would be used to represent the melodies of the collection and the query that is presented to the system. Using a sequence of intervals or pitch distances is an effective choice for standardisation. The search mechanism that satisfies both retrieval effectiveness and efficiency concerns is the use of an n-gram index that uses n-grams of length five. The ideal index structure for this hasn't been determined. However, a look-up array for the first two to four characters in the n-gram and overflow binary trees for searching for the remainder of each n-gram is a simple practical structure that can be used. The size of the structure can be further reduced with compression. The n-gram index would be used to look up the pieces that contain the same n-grams as the query. To rank the pieces for presentation to the user a count of the n-grams that each piece has in common with the query disregarding duplicates would be the most effective similarity measurement.

The method of presenting answers to the user has not been addressed in detail as yet. However, our experience with collecting relevance judgements suggests that the positions within the piece that match the query should be clearly shown in the display, allowing the user to listen to the segment of interest in the least time-consuming way.

On a more theoretical note, our work has suggested further questions that need to be addressed in the parent discipline of music psychology, in addition to the future work required in MIR research. Also, we found that the type of queries used for evaluating MIR techniques significantly affect the outcome of the experiments. As such it is important that sets of real queries are made available so that evaluation of techniques can be done as appropriately as possible.

In summary, we have achieved the aim of this thesis which was to find methods that successfully produce answers to melody fragment queries. We have developed an approach to evaluating MIR systems, an evaluation of four melody extraction algorithms, the evaluation of many variables that affect two main approaches to melody matching, two techniques that are successful at matching melodies, and a set of tools that can be used to evaluate MIR systems. These advances are a major step towards practical music information retrieval.

Bibliography

[1] A. Apostolico and Z. Galil, editors. *Pattern Matching Algorithms.* Oxford University Press, New York, New York USA, 1997.

[2] R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Communications of the ACM*, 35(10):74–82, 1992.

[3] R. A. Baeza-Yates and C. H. Perleberg. Fast and practical approximate string matching. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proceedings of the 3rd Annual Symposium on Combinatorial Pattern Matching*, number 664, pages 185–192, Tucson, AZ, 1992. Springer-Verlag, Berlin.

[4] D. Bainbridge. MELDEX: A web-based melodic locator service. *Computing in Musicology*, 11:223–229, 1998.

[5] D. Bainbridge. Towards a digital library of popular music. In *Proc. ACM Digital Libraries.* ACM, 1999.

[6] I. V. Bakhmutova, V. D. Gusev, and T. N. Titkova. A search and classification of imperfect repetitions in song melodies. *Acta et Commentationes Universitatis Tartuensis: Quantitative Linguistics and Automatic Text Analysis*, 827:20–32, 1988. in Russian.

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

[8] Barlow and Morganstern. *A dictionary of Musical Themes.* 1948.

[9] J. C. Bartlett and W. J. Dowling. Recognition of transposed melodies: A key-distance effect in developmental perspective. *Journal of Experimental Psychology: Human Perception and Performance*, 6(3):501–515, 1980.

[10] D. Beeferman. QPD: Query by pitch dynamics indexing tonal music by content. 1998.

[11] S. Blackburn and D. D. Roure. A tool for content-based navigation of music. In *Proc. ACM International Multimedia Conference*. ACM, Sept. 1998.

[12] S. Blackburn and D. D. Roure. Musical part classification in content based systems. In *Open Hypermedia Systems Workshop*, volume 6, pages 66–76, 2000.

[13] D. C. Blair. STAIRS redux: Thoughts on the STAIRS evaluation, ten years after. *Journal of the American Society for Information Science*, 47:4–22, 1996.

[14] J. Borchers. Worldbeat: Designing a baton-based interface for an interactive music exhibit. In *Proc. CHI 97*, pages 131–138, New York, 1997. ACM, ACM Press.

[15] J. Borchers and M. Muhlhauser. Design patterns for interactive musical systems. *IEEE Multimedia*, 5(3):36–46, 1998.

[16] D. Byrd, J. S. Downie, T. Crawford, W. B. Croft, and C. Nevill-Manning, editors. *International Symposium on Music Information Retrieval*, volume 1, Plymouth, Massachusetts, Oct. 2000.

[17] H. Charnasse and B. Stepien. Automatic transcription of german lute tablatures: an artificial intelligence application. In Marsden and Pople [88], pages 143–170.

[18] A. L. P. Chen, M. Chang, J. Chen, J.-L. Hsu, C.-H. Hsu, and S. Y. S. Hua. Query by music segments: An efficient approach for song retrieval. In *Proc. IEEE International Conference on Multimedia and Expo*, pages 873–876. IEEE, 2000.

[19] B. Chen. Query by singing. Master's thesis, National Tsing Hua University, 1998.

[20] J. C. C. Chen and A. L. P. Chen. Query by rhythm an approach for song retrieval in music databases. In *Proceedings of IEEE International Workshop on Research issues in Data Engineering*, pages 139–146. IEEE, 1998.

[21] T.-C. Chou, A. L. P. Chen, and C.-C. Liu. Music databases: Indexing techniques and implementation. In *Proceedings IEEE International Workshop in Multimedia DBMS*, 1996.

[22] G. S. Choudhury, T. DiLauro, M. Droetboom, I. Fujinaga, B. Harrington, and K. MacMillan. Optical music recognition system within a large-scale digitization project. In Byrd et al. [16].

[23] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. PROMS: A web-based tool for searching in polyphonic music. In Byrd et al. [16].

[24] C. Cleverdon and J. Mills. The cranfield tests on index language devices. In *Aslib Proceedings*, volume 19, pages 173–192, 1967.

[25] D. Cope. Signatures and earmarks: Computer recognition of patterns in music. *Computing in Musicology*, 11:129–138, 1998.

[26] F. Crane and J. Fiehler. Numerical methods of comparing musical styles. In Lincoln [82], chapter 15, pages 209–222.

[27] T. Crawford, C. S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11, 1998.

[28] M. Crochemore. Off-line serial exact string searching. In Apostolico and Galil [1], chapter 1, pages 1–53.

[29] M. Crochemore, C. S. Iliopoulos, and H. Yu. Algorithms for computing evolutionary chains in molecular and musical sequences. In C. S. Iliopoulos, editor, *Proceedings of the ninth Australian Workshop on Combinatorial Algorithms AWOCA'98*, pages 172–184, School of Computing, Curtin University of Technology, Perth, Western Australia, 1998.

[30] C. Cronin. Concepts of melodic similarity in music-copyright infringement suits. *Computing in Musicology*, 11:187–209, 1998.

[31] D. Deutsch. Grouping mechanisms in music. In *The Psychology of Music* [32], chapter 4, pages 99–134.

[32] D. Deutsch, editor. *The Psychology of Music.* Academic Press, Inc., 1982.

[33] M. Dillon and M. Hunter. Automated identification of melodic variants in folk music. *Computers and the Humanities*, 16:107–117, 1982.

[34] M. J. Dovey. An algorithm for locating polyphonic phrases within a polyphonic musical piece. In *Proceedings of the AISB Symposium on Musical Creativity*, Edinburgh, Apr. 1999. AISB.

[35] M. J. Dovey. Heuristic models of relevance ranking in searching polyphonic music. In *Diderot Mathematical Forum: Conference on Computational and Mathematical Methods in Music*, Vienna, Dec. 1999. European Mathematical Society.

[36] W. J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.

[37] W. J. Dowling. Melodic information processing and its development. In Deutsch [32], chapter 13, pages 413–429.

[38] W. J. Dowling and D. S. Fujitani. Contour, interval and pitch recognition in memory for melodies. *The Journal for the acoustical society of America*, 49(2):524–531, 1971.

[39] J. S. Downie. The Musifind musical information retrieval project, phase II: User assessment survey. In *Proceedings of the annual conference of the Canadian Association for Information Science, Montreal*, pages 144–166. CAIS, 1994.

[40] J. S. Downie. The Musifind music information retrieval project, phase III: evaluation of indexing options. In *Canadian Association for Information Science proceedings of the 23rd Annual Conference, Connectedness: Information, Systems, People, Organisations*, pages 135–46. CAIS, 1995.

[41] J. S. Downie. Informetrics and music information retrieval: an informetric examination of a folksong database. In *Proceedings of the Canadian Association for Information Science, 1998 Annual Conference*, Ottawa, Ontario, 1998. CAIS.

[42] J. S. Downie. *Evaluating a Simple Approach to Musical Information Retrieval: Conceiving Melodic N-grams as Text.* PhD thesis, University of Western Ontario, 1999.

[43] J. S. Downie. Access to music information: The state of the art. *Bulletin of the American Society for Information Science*, 26(5), June/July 2000.

[44] J. S. Downie. Evaluation of a simple and effective music information retrieval method. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 73–80, Athens, Greece, July 2000. ACM, ACM.

[45] B. M. Eaglestone. Extending the relational database model for computer music research. In Marsden and Pople [88], pages 41–66.

[46] W. Ebeling and T. Poschel. Entropy, transinformation and word distribution of information-carrying sequences. *International Journal of Bifurcation and Chaos*, 5(1):51–61, 1995.

[47] J. Edworthy. Interval and contour in melody processing. *Music Perception*, 2(3):375–388, 1985.

[48] D. P. W. Ellis. A computer implementation of psychoacoustic grouping rules. Technical Report 224, MIT, 1994.

[49] J. Foote. Visualizing music and audio using self-similarity. In *Proc. ACM International Multimedia Conference*, pages 77–80, Orlando Florida, USA, Oct. 1999.

[50] J. Foote. ARTHUR: Retrieving orchestral music by long-term structure. In Byrd et al. [16].

[51] R. Francès. *La Perception de la Musique.* L. Erlbaum, Hillsdale, New Jersey, 1958. Translated by W. J. Dowling (1988).

[52] C. Francu and C. G. Nevill-Manning. Distance metrics and indexing strategies for a digital library of popular music. In *IEEE International Conference on Multimedia and Expo (II)*, pages 889–892, 2000.

[53] A. J. Gabura. Music style analysis by computer. In Lincoln [82], chapter 16, pages 223–276.

[54] A. Ghias, J. Logan, D. Chamberlin, and B. Smith. Query by humming — musical information retrieval in an audio database. In *Proc. ACM International Multimedia Conference*, 1995.

[55] R. Giancarlo and R. Grossi. *Suffix Tree Data Structures for Matrices*, chapter 10, pages 293–340. In Apostolico and Galil [1], 1997.

[56] M. Good. Representing music using XML. In Byrd et al. [16].

[57] M. Goossens, S. Rahtz, and F. Mittelbach. *The LaTeXGraphics Companion.* Addison-Wesley, 1997.

[58] C. Grande. The notation interchange file format: A windows-compliant approach. In Selfridge-Field [119], chapter 31, pages 491–512.

[59] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge, USA, 1997.

[60] D. Halperin. Musical chronology by seriation. *Computers and the Humanities*, 28:13–18, 1994.

[61] S. Handel. *Listening: An introduction to the perception of auditory events.* MIT Press, 1989.

[62] M. Hawley. The personal orchestra, or audio data compression by 10,000:1. *Computing Systems*, 3(2):289–329, 1990.

[63] S. Heinz, J. Zobel, and H. E. Williams. Burst tries: A fast, efficient data structure for string keys. 2001. in submission.

[64] D. S. Hirschberg. Serial computations of Levenshtein distances. In A. Apostolico and Z. Galil, editors, *Pattern Matching Algorithms*, chapter 4, pages 123–141. Oxford University Press, 1997.

[65] J.-L. Hsu, C.-C. Liu, and A. L. P. Chen. Efficient repeating pattern finding in music databases. In *Proceedings of the 1998 ACM 7th international conference on Information and knowledge management*, pages 281–288. ACM, ACM Press, 1998.

[66] B. Hudson. Toward a comprehensive French chanson catalog. In Lincoln [82], chapter 17, pages 277–287.

[67] D. Huron. *Humdrum reference manual.*

[68] D. Huron. Design principles in computer-based music representation. In Marsden and Pople [88], pages 5–39.

[69] D. Huron and M. Royal. What is melodic accent? converging evidence from musical practice. *Music Perception*, 13(4):489–516, 1996.

[70] K. S. Jones. Reflections on trec. *Information Processing and Management*, 31(3):291–314, 1995.

[71] T. Kageyama, K. Mochizuki, and Y. Takashima. Melody retrieval with humming. In *Proc. International Computer Music Conference*, 1993.

[72] T. Kageyama and Y. Takashima. A melody retrieval method with hummed melody. *Transactions of the Institute of Electronics, Information and Communication Engineers*, J77-D-II(8):1543–1551, 8 1994. in Japanese.

[73] L. Knopoff and W. Hutchinson. Entropy as a measure of style: the influence of sample length. *Journal of Music Theory*, 27:75–97, 1983.

[74] A. Kornstadt. A web based melodic search tool. *Computing in Musicology*, 11:231–236, 1998.

[75] N. Kosugi, Y. Nishihara, S. Kon'ya, M. Yamamuro, and K. Kushima. Music retrieval by humming - using similarity retrieval over high dimensional feature vector space. In *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, Canada, Aug. 1999.

[76] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima. A practical query-by-humming system for a large music database. In *Proc. ACM International Multimedia Conference*, Los Angeles, USA, Nov. 1999.

[77] C. L. Krumhansl and R. N. Shepard. Quantification of the hierarchy of tonal functions within a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance*, 5(4):579–594, 1979.

[78] K. Lemström and P. Laine. Musical information retrieval using musical parameters. In *Proc. International Computer Music Conference*, pages 341–348, Ann Arbour, 1998.

[79] K. Lemström, P. Laine, and S. Perttu. Using relative interval slope in music information retrieval. In *Proc. International Computer Music Conference*, pages 317–320, Beijing, China, Oct. 1999.

[80] K. Lemström and J. Tarhio. Detecting monophonic patterns within polyphonic sources. In J. Mariani and D. Harman, editors, *Proc. Conference on Content-Based Multimedia Information Access*, volume 6 of *RIAO*, Paris, France, Apr. 2000.

[81] D. J. Levitin and P. R. Cook. Memory for musical tempo: Additional evidence that auditory memory is absolute. *Perception and Psychophysics*, 58(6):927–935, 1996.

[82] H. Lincoln, editor. *The Computer and Music.* Cornell University Press, Ithaca, New York, 1970.

[83] A. T. Lindsay. Using contour as a mid-level representation of melody. Master's thesis, MIT, Massachusetts, 1996.

[84] C.-C. Liu, J.-L. Hsu, and A. L. P. Chen. Efficient theme and non-trivial repeating pattern discovering in music databases. In *Proc. IEEE International Conference on Data Engineering*, pages 14–21, Sydney, Australia, 1999. IEEE, IEEE Computer Society Press.

[85] L. Logrippo and B. Stepien. Cluster analysis for the computer-assisted statistical analysis of melodies. *Computers and the Humanities*, 20:19–33, 1986.

[86] C. D. Manning and H. Schutze. *Foundations of statistical natural language processing.* MIT Press, Cambridge, Massachusetts, 1999.

[87] A. Marsden. Modelling the perception of musical voices: a case study in rule-based systems. In Marsden and Pople [88], pages 239–263.

[88] A. Marsden and A. Pople, editors. *Computer Representations and Models in Music.* Academic Press, London, England, 1992.

[89] K. D. Martin. A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, MIT, 1996.

[90] K. McMillen. ZIPI: Origins and motivations. *Computer Music Journal*, 18(4):47–51, 1994.

[91] R. J. McNab. Interactive applications of music transcription. Master's thesis, Department of Computer Science, University of Waikato, New Zealand, 1996.

[92] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The New Zealand Digital Library MELody inDEX. *Digital Libraries Magazine*, May 1997.

[93] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proc. ACM Digital Libraries*, 1996.

[94] M. Melucci and N. Orio. Musical information retrieval using melodic surface. In *Proc. ACM Digital Libraries*. ACM, 1999.

[95] M. Melucci and N. Orio. The use of melodic segmentation for content-based retrieval of musical data. In *Proc. International Computer Music Conference*, Beijing, China, 1999. International Computer Music Association.

[96] C. B. Monahan, R. A. Kendall, and E. C. Carterette. The effect of melodic and temporal contour on recognition memory for pitch change. *Perception and Psychophysics*, 41(6):576–600, 1987.

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

[98] C. Nevill-Manning and I. H. Witten. Protein is incompressible. In *Proc. IEEE Data Compression Conference*, pages 257–266, Snowbird, Utah, Mar. 1999. IEEE.

[99] D. Ó'Maídin. A geometrical algorithm for melodic difference. *Computing in Musicology*, 11:65–72, 1998.

[100] Parsons. *The Directory of Tunes*. Spencer Brown and Co., Cambridge, England, 1975.

[101] J. Pickens. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In Byrd et al. [16].

[102] E. Pollastri. Melody-retrieval based on pitch-tracking and string-matching methods. In *Proc. Colloquium on Musical Informatics*, Gorizia, 1998.

[103] D.-J. Povel and P. Essens. Perception of temporal patterns. *Music Perception*, 2(4):411–440, 1985.

[104] J. Rameau. *Treatise on Harmony*. Dover Publications, New York, USA, 1971. Originally published in 1722. Translated by Philip Gossett.

[105] M. F. Reynolds. Selle v. gibb and the forensic analysis of plagiarism. In *College Music Symposium*, volume 32, pages 55–78. College Music Society, 1992.

[106] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR 94*, pages 232–241. ACM, 1994.

[107] P. Rolland, G. Raskinis, and J. Ganascia. Musical content-based retrieval: an overview of the melodiscov approach. In *Proc. ACM International Multimedia Conference*, pages 81–84, Orlando, Florida, USA, Oct. 1999.

[108] D. C. D. Roure and S. G. Blackburn. Content based navigation of music using melodic pitch contours. *Multimedia Systems Journal*, 8(3):190–200, 2000.

[109] R. Rousseau. A fractal approach to word occurrences in texts: the zipf-mandelbrot law for multi-word phrases. 1998.

[110] R. Rousseau. George Kingsley Zipf: life, ideas and recent developments of his theories. In *Beijing International Seminar of Quantitative Evaluation of Research and Development in Universities, and Fifth All-China Annual Meeting for Scientometrics and Informatics*, Beijing, Dec. 1998. preprint.

[111] W. B. Rubenstein. A database design for musical information. In U. Dayal and I. Traiger, editors, *Proc. ACM-SIGMOD International Conference on the Management of Data*, pages 479–490, San Francisco, CA, May 1987. ACM, ACM Press.

[112] P. Salosaari and K. Järvelin. MUSIR: A retrieval model for music. Technical Report 1, University of Tampere, 1998.

[113] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[114] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[115] D. Sankoff and J. B. Kruskal, editors. *Time Warps: String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983.

[116] H. Schaffrath. The retrieval of monophonic melodies and their variants: Concepts and strategies for computer-aided analysis. In Marsden and Pople [88], pages 95–110.

[117] E. D. Scheirer. *Music Listening Systems.* PhD thesis, MIT, Massachusetts, June 2000.

[118] R. Sedgewick. *Algorithms in C.* Addison-Wesley, Reading, Massachussetts, second edition, 1990.

[119] E. Selfridge-Field, editor. *Beyond MIDI: the Handbook of Musical Codes.* MIT Press, 1997.

[120] E. Selfridge-Field. Conceptual and representational issues in melodic comparison. *Computing in Musicology*, 11:3–64, 1998.

[121] Y. Shi. Correlations of pitches in music. *Fractals*, 4(4):547–553, 1996.

[122] T. Sonoda and Y. Muraoka. A www-based melody-retrieval system - an indexing method for a large melody database. In *Proc. International Computer Music Conference*, 2000.

[123] J. Spitzer. 'oh susanna': Oral transmission and tune transformation. *Journal of the American Musicological Society*, 47(1):90–136, 1994.

[124] D. A. Stech. A computer-assisted approach to micro-analysis of melodic lines. *Computers and the Humanities*, 15:211–221, 1981.

[125] J. Stinson. The scribe database. *Computing in Musicology*, 8:65, 1992.

[126] J. Sundberg. Perception of singing. In Deutsch [32], chapter 3.

[127] J. Tague-Sutcliffe, J. S. Downie, and S. Dunne. Name that tune! an introduction to musical information retrieval. In *Proceedings of the annual conference of the Canadian Association for Information Science, Montreal*, pages 203–211. CAIS, 1993.

[128] H. G. Tekman. Interactions of perceived intensity, duration and pitch in pure tone sequences. *Music Perception*, 14(3):281–294, 1997.

[129] H. G. Tekman. Effects of melodic accents on perception of intensity. *Music Perception*, 15(4):391–401, 1998.

[130] J. M. Thomassen. Melodic accent: experiments and a tentative model. *The Journal for the acoustical society of America*, 71(6):1596–1605, June 1982.

[131] Y.-H. Tseng. Content-based retrieval for music collections. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 176–182. ACM SIGIR, ACM Press, 1999.

[132] A. L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In B. Smith and W. Effelsberg, editors, *Proc. ACM International Multimedia Conference*, pages 235–240, Bristol, UK, Sept. 1998. ACM, ACM Press.

[133] A. L. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In D. Bulterman, K. Jeffay, and H. J. Zhang, editors, *Proc. ACM International Multimedia Conference*, pages 57–66, Orlando Florida, USA, Nov. 1999. ACM, ACM Press.

[134] A. L. Uitdenbogerd and J. Zobel. Music ranking techniques evaluated. In *International Symposium on Music Information Retrieval*, Plymouth, Massachussetts, USA, Oct. 2000. poster.

[135] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92:191–211, 1992.

[136] R. van Egmond and D.-J. Povel. Perceived similarity of exact and inexact transpositions. *Acta Psychologica*, 92:283–295, 1996.

[137] T. von Schroeter, S. Doraisamy, and S. M. Rüger. From raw polyphonic audio to locating recurring themes. In Byrd et al. [16]. poster submission.

[138] M. Welsh, N. Borisov, J. Hill, R. von Behren, and A. Woo. Querying large collections of music for similarity, Nov. 1999.

[139] W. J. Wilbur and D. J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proc. National Academy of Sciences USA*, 80:726–730, Feb. 1983.

[140] H. Williams. *Indexing and retrieval for genomic databases.* PhD thesis, RMIT, Melbourne, Victoria, Australia, 1998.

[141] H. Williams and J. Zobel. Indexing and retrieval for genomic databases. *IEEE Transactions on Knowledge and Data Engineering.* (to appear).

[142] H. Williams and J. Zobel. Indexing nucleotide databases for fast query evaluation. In *Proc. Int. Conf. on Advances in Database Technology (EDBT)*, pages 275–288, Avignon, France, Mar. 1996.

[143] I. H. Witten and T. C. Bell. Source models for natural language text. *International Journal on Man Machine Studies*, 32:545–579, 1990.

[144] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kaufmann, San Francisco, California, second edition, 1999.

[145] D. Wolfram. Applications of informetrics to information retrieval research. *Informing Science*, 3(2):77–82, 2000.

[146] S. Wu and U. Manber. Fast text searching allowing errors. *Communications of the ACM*, 35(10):83–91, Oct. 1992.

[147] C. Zaniolo, S. Ceri, C. Faloutsos, R. Snodgrass, V. S. Subrahmanian, and R. Zicari. *Advanced Database Systems.* Data Management Systems. Morgan Kaufmann, San Francisco, 1997.

[148] R. H. Zaripov. The construction of frequency vocabularies of musical intonations for analysis and simulation of melodies. *Problems of Cybernetics*, 41:207–252, 1984. in Russian.

[149] G. K. Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, 1949.

[150] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In R. Wilkinson, B. Croft, K. van Rijsbergen, A. Moffat, and J. Zobel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, July 1998.

[151] J. Zobel and P. Dart. Finding approximate matches in large lexicons. *Software—Practice and Experience*, 25(3):331–345, 1995.

[152] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.

[153] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23(4):453–490, 1998.