Modulo7 : A full stack Music Information Retrieval and Querying Engine using Music Theoretic Principles

by

Arunav Sanyal

A thesis submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science.

Baltimore, Maryland

December, 2015

Abstract

Music Information Retrieval (MIR) is an interdisciplinary science of extracting non trivial information and statistics from music data sources. In today's digital age, music is stored in a variety of digitized formats - e.g midi, musicxml, mp3, digitized sheet music etc. Music Information Retrieval Software aim at extracting features from one or more of these source. MIR research helps in solving problems like automatic music classification, recommendation engine design etc. Users can then query the acquired statistics to acquire relevant information.

The author proposes and implements a new Music Information Retrieval and Query Engine called Modulo7. Unlike other MIR software which deal with low level audio features, Modulo7 operates on the principles of music theory and a symbolic representation of music. Modulo7 is a full stack deployment, with server components that parse various sources of music data into its own efficient internal representation and a client component that allows consumers to query the system with sql like queries which satisfies certain music theory criteria (and as a consequence Modulo7 has a custom relational algebra with its basic building blocks based on music theory).

Primary Reader: Dr David Yarowsky

Secondary Reader: Dr Yanif Ahmad

Acknowledgments

I would like to thank Dr David Yarowsky for giving me the opportunity to work on this project. His detailed insights have immensely helpful to me to power through my work and to also make the technical depth of the project accessible to laymen. I would like to thank Dr Yanif Ahmad for his crucial help in the systems aspects of my query engine and the implementation of the server side components. I would like to thanks Natalie Draper in the Peabody Conservatory, my instructor for music theory for teaching me the basics of music theory. I would like to thank Dr Cory Mckay from McGill University for his help with understanding concepts in symbolic music Information Retrieval and help with implementation specifics for symbolic retrieval. I would like to thank Dr Ichiro Fuginaga from McGill University for his guidance and help with Optical Music Recognition concepts.

I would like to thank my friends Aakash, Ankit, Satya and Japneeth for their support, help with technical and coding aspects of the project.

ACKNOWLEDGMENTS

Most importantly I would like to thanks my family for their unending support, and for instilling a love for music which has allowed me to take this in depth study of application of Computer Science to Music theory.

Dedication

This thesis is dedicated to my family and to all the music lovers in the world.

Contents

Chapter 1

Introduction

Why does a person like a particular song? What are the inherent aspects of a song that pleases a person's musical taste? Is it the complexity of a song, the beat the song or just a particular melodic pattern ? More so if a person likes a song, can we predict if he/she will like a similar song?

Music has been created since the dawn of civilization and these questions have plagued mankind just as long. In response to this, man has created elaborate systems of formal study for music and classification techniques in almost every ethnic community since antiquity. Two notable examples are the western system of solfege and classical music theory and the Indian system of raagas. These elaborate systems are based on very simple fundamental building blocks of melody and harmony and simple rules that govern the interplay of these building blocks. However very complex pieces of music can be created with these simple rules depending on the skill and virtuosity of artists. Composers use these rules and concepts to create novel music for mass consumption.

In the modern era industry and academia have attempted to address the problem of music recommendation and music classification. The industry has predominantly favored approaches that look at user preferences and history. For example Amazon Music recommendation works on users shopping history. Pandora on the other hand hires an army of musicologists to ascertain how a song is similar to another song and creates software that leverages this adhoc generated data. These approaches are either expensive in the human labor needed or in the amount of data processed that is input from a large number of users. More recently, companies like Echo Nest has extensively extracted features from music sources and mined cultural information on the web but leave it at consumers how best to leverage the data. Hence symbolic MIR is not traditionally used in industry and music theory is an after thought.

Academia on the other hand attempts to solve very particular problems in MIR. Typical examples would be cover song detection, processing information via signal processing, audio feature extraction, optical music recognition etc. In most cases the applications are of a very specific domain and does not fully scale with bulk music data. Generic frameworks like the jMIR[1] (which also happens to be a major inspiration for Modulo7) suite for automatic music classification exists, which is meant to facilitate research in MIR with a machine learning focus. However academia is disconnected with industry and no full scale MIR engines can satisfy the scale of industry applications.

This work is attempt to bridge both communities. Modulo7 is a full stack deployment of Music Information Retrieval Software, providing both a server architecture and a sql like client to query based on music theory criteria. Modulo7 is a big data and information retrieval framework to explore the possibilities of exploring music theoretical aspects of music sources. Modulo7 does not attempt to solve very complex music theoretic problems (e.g study orchestral music to identify counter point information). Rather Modulo7 acts a framework on which such analysis can be built upon. Most importantly, Modulo7 addresses the issue of scale and allows a fast comparison between songs on certain music theoretic criteria. Modulo7 also acts to address deficiencies in existing software, such as filling up incompleteness in music sources. Certain problem statement of this sort would be Key estimation, Tempo estimation etc.

Modulo7 only includes a unique yet novel indexing scheme. This indexing scheme involves creating an inverted index for global properties of songs (key signature, homophonic, time signature etc) . This indexing scheme allows for fast lookups for certain types of queries (e.g find all songs that in the key of C Major) and also allows for speedup in scenarios which require criteria based on indexed terms.

Chapter 2

Literature Review

Music Information Retrieval is an active and vibrant community. Both academia and industry diligently pursue it albeit with different goals in mind. While academia's primary aim is to explore particular problems (e.g cover song detection, estimating chords from chroma vectors[2]) etc. Whereas Industry is primarily interested in solving problems like song recommendation and similarity searches. The following sections outlines the software efforts and research problems tackled by MIR community in general.

2.1 Current MIR Software

Both Industry and Academia have created an extensive set of software for solving these problems. The author presents an overview of such software and the problems they attempt to address. The following software packages were investigated

2.1.1 jMIR

jMIR,[1] or Java Music Information Retrieval tool set is a collection of java code, GUI, API and CLI tools for the purpose of feature extraction from variety of music sources (in particular audio, midi) and mine cultural information from the web. jMIR extracts and exhaustive set of features that can be used in machine learning tasks. The primary use of jMIR is automatic music classification and feature extraction and not similarity computations per se (which is one of Modulo7's core goals). Moreover jMIR does not scale to myriad sources of music in existence. Unlike Modulo7 jMIR also relies on faithful recordings and does not attempt to fill up missing information (like key signature not being encoded etc). Nevertheless its one of the best Open Source MIR software in existence especially for MIR research.

2.1.2 Marsyas

marsyas[3] (Music Analysis, Retrieval and Synthesis for Audio Signals) is a software stack for audio processing with specific emphasis on Music Information Retrieval and music signal extraction. Marsyas is a heavily developed and a widely utilized state of the art framework for audio processing but also has a steep learning curve. Modulo7 has different goals (multiple format support, music similarity, structured querying etc) as compared to marsyas.

2.1.3 SIMILIE

SIMILIE[4] is a set of tools for music similarity measures used for single melodies and features multiple ways to construct vector space models for melodies. The techniques used for melodic similarity analyss in SIMILIE are novel. Modulo7 uses a subset of these similarity measures as basis for an extended and improved model of similarities based on polyphonic music and harmonic elements. Moreover SIMILIE needs its own file format (called .mcsv) for analysis. Although the software package gives a converter for different sources, its not as variegated as Modulo7's format support is (which directly parses different music source files).

2.1.4 Echo Nest APIs

Echo Nest is a company that specializes in big data music intelligence. Echo Nest APIs and backend powers many music platforms like last.fm, Spotify etc. In particular Echo Nest provides APIs for extraction of audio features, acquiring artists similar to a particular artist etc. Echo Nest API is used for some sub tasks in Modulo7 (which is discussed in the Software Architecture Chapter).

Echo Nest also maintains the worlds biggest music database as well as data mined from them along with extracted audio features, web mined information, user preference etc).

2.1.5 Humdrum

Humdrum[5] is a set of tools for computer based automation and assistance in music research. Humdrum has the capability for solving very complex questions using music theoretic concepts. Humdrum supports its own file format for analysis. Humdrum is specifically designed for musicologists for automating tasks that they otherwise would have required manual analysis and not gathering statistic, music classification or music similarity analysis as an end goal.The fundamental difference of Modulo7 over humdrum is modulo7 acts as a bulk analysis tool while humdrum is designed for specific analysis of songs.

2.1.6 Gamera

Gamera[6] is Optical Symbol Recognition Open Source software based on supervised and hybrid learning approaches for training. Gamera is designed with the particular aim of symbol recognition of old documents. Gamera also supports creating of new plugins for custom tasks. For the purpose of Music Information Retrieval, gamera can be used to solve the problem of Optical Music Recogntion (OMR) since sheet music images are also a format for music source.

2.1.7 Audiveris

Audiveris is an Open source software for Optical Music Recognition. Unlike Gamera, Audiveris can be directly consumed as a service for the purpose of OMR. Audiveris is used as service in many leading Notation Platforms like Musescore etc. As such, Audiveris is used as a subcomponent of Modulo7's architecture for OMR.

2.2 Music Representation Formats

Modulo7 parses multiple formats for music. But there are many other sources that are worth mentioning.

GUIDO : GUIDO musical notation format is a computer notation format that is made to logically represent symbolic musical information that is easily readable by both humans and computers and can be stored as a text file.

KERN : The kern format is used in humdrum to symbolically denote events in columns while voices are represented in rows.[5] This facilitates a columnar representation of music on which humdrum can perform some sort of music theoretic analysis.

2.3 Typical problems of MIR

On top of the generic software created by researchers and industry experts, researchers have tackled specific problems in Music Cognition, Classification, Cover song identification, Query by Humming Systems etc. Only certain approaches have been incorporated in Modulo7 which help completing metadata information (e.g. if the key signature of a song is not present, Modulo7 estimates it using). Broadly speaking though, the problem statement falls in the following broad categories :-

2.3.1 Music Classification / Genre Identification

The problem of music classification is to assign a tag (also called a genre to a song) which broadly categorizes it according to some criteria. While the genre definitions for songs are often vague, it helps in giving information about which songs are relevant. Companies like Pandora and Microsoft assign genres to songs via musicologists[7] which means highly trained people manually classify music. Such approaches are expensive in terms of human labor and prone to error. Automatic Music Classification takes a different approach using different algorithms and machine learning approaches like jMIR[1] does to classify music.

2.3.2 Music Similarity Analysis

The problem of music similarity analysis lies at the heart of a large number other applications like Song Identification, Query by humming systems etc. Most literature have addressed the problem of melodic similarity[8] and not on generic polyphonic similarity. There are many systems and music databases in existence for the purpose of music similarity analysis.

2.3.3 Automated Musicological Research

In many cases musicological research is pursued manually by applying rules and music theoretic criteria. An example would be applying counterpoint analysis techniques given in a treatise[9] to music sheet manually. This is labor intensive and the research community tries to address automated analysis of music. A significant effort is done by the Humdrum community.[5] in automated musicological research.

2.3.4 Audio processing and feature extraction

Most music is represented in Audio format rather than symbolic format, as consumption of music is primarily for the layman or the musically uninitiated. One such task would be music transcription(also known as melody extraction[10]). This is needed for an accurate transcription of audio music to symbolic format. However researchers have only found success in melody extraction where one voice is clearly dominant in a recording.[10] Researchers have also worked on quantitatively defining the concept of timbre (a peculiar tone of a voice independent of pitch and loudness) with varying degrees of success both qualitatively[11] and computationally.[12]

2.3.5 Intelligent Music Archiving and Retrieval

Key to music information retrieval are efficient and novel techniques to archive musical sources so that they meaningful queries can be made against these archives. Many libraries and library sciences programs work actively in this regard. Our very own Johns Hopkins University Eisenhower Library has a vast collection of Sheet music on American Popular music called the "Lester Levy Sheet Music Collection"[13] . There are many such collections worldwide. There are many labs and institutions which work towards archiving digitized sheet music, notable among them are the DDMAL lab in McGill University[14] which works in archiving medieval sheet music in a digitized form as well as perform statistical analysis on it.

2.3.6 Music Recommendation

Perhaps the most commercialized application of Music Information Retrieval is the task of music recommendation i.e. intelligent suggestion of songs to a user given his or her preferences and past listening history. Music recommendation is an end goal in itself and not a distinct problem compared to the previous problems discussed in this section. In order to facilitate this various music databases and query systems are built and comparisons are based on lyrics genre tags and other properties of music data.[15] Most approaches have been based on collaborative filtering contextual metadata (information based on community of users) and sparingly from low level features extracted from a song.

2.3.7 Audio Fingerprinting and Song ID

A very industry relevant problem statement involves fingerprinting audio files and matching the finger print to an input(melody or fragment of a song) fingerprint. This is a subclass of the query by humming problem, with an exact match emphasized as the end goal. Many commercial systems are in existence including companies like Shazam[16] which have developed sophistated algorithms and systems dedicated to solve this problem.

Chapter 3

Basics of Music Theory

Music theory is defined as the systematic study of the structure, complexity and possibilties of what can be expressed musically. More formally its the academic discipline of studying the basic building blocks of music and the interplay of these blocks to produce complex scores (pieces of music). Modulo7 is built on top of western theoretic principles and hence only western music theory is explored. Also music theory is an extremely complicated subject and hence only the basics and relevant portions to the modulo7 implementation are discussed here.

Traditionally music theory is used for providing directives to a performer to play a particular song/score.

This section is primarily meant for people with a weak or lack of understanding of western music theory. The following section talks about the basic building blocks of music theory:-

3.1 Building Blocks

Music is built on fundamental quantities (much like matter is built on fundamental quantities like atoms and molecules). The following are the core concepts in order of atomicity (i.e successive blocks build on the preceding ones)

Pitch/Note: A pitch is a deterministic frequency of sound played by a musical voice (instrument or human). In western music theory, certain deterministic pitches are encoded as Notes. For example the note A4 is equal to 440 Hz. In other words Notes are symbolic representations of certain pitches. With certain notable exceptions, most music is played on these set frequencies.

Each note is characterized by two entities. First is the note type and the second is the octave. An octave can be considered as a range of 12 notes. There are 8 octaves numbered 0 to 7 which are played by traditional instruments or vocal ranges. Then is the note type. Notes are categorized into 7 major notes (called A, B, C, D, E, F, G) and 5 minor notes (also called as accidentals). They can be characterized by increasing or decreasing the frequency of the notes by a certain amount (called sharps(#) and flats(b) respectively). For example the accidental lying in between (A and B is called A# or Bb). Similarly accidentals lie in between C, D; D, E; F, G and G, A. (Note that there are no accidentals in between B and C and E and F).

Semitone and Tone: A semitone is an increment or a decrement between two notes. For instance there is one semitone in between A and A#. Similarly there are 3 semitones in between A and C. A tone is an increment in between two major notes. Another characterization of a tone is two semitones.

Beat/Tick: A beat or tick is a rhythmic pulse in a song. Beats in sequence is used to maintain a steady pulse on which the rhythmic foundations of a song is based.

Pitch duration: A pitch duration is a relative time interval the pitch persists on a musical instrument. For example a whole note will persist twice as longer as a half note.

Attack/Velocity: The intensity or force with which a pitch is played. This parameter influences the loudness of the note and in general the dynamics of the song (covered in a subsequent section)

Chord: A chord is a set of notes being stacked together (being played together at or almost at the same time). Chords are the basic building blocks of a concept called harmony (which will be discussed further on.). Traditionally a chord is constructed by stacking together notes played on a single instrument, but a chord can be constructed by different instruments simultaneously playing different notes.

Rests: Rests are pauses in between notes (with no sound being played at that point of time) for a fixed duration.

Melody: A melody is a succession of notes and rests which sound pleasing. There are many rules about what makes a melody sound good which we will get to in the subsequent reading.

Harmony: A harmony is a succession of chords (also known as a chord progression) along with the principles that govern the relationships between different chords.

Voice: A voice is an interplay of notes, chords and stops by a single instrument/vocalist. The reader can think of a voice as a hybrid or generalization of the melody and harmony concepts.

Register: For a given voice, the register of a voice is the range of notes that the singer of that voice can comfortably sing or a musical instrument sounds nice in.

Range: For a given voice, the range of a voice is the range between the maximum and minimum notes that a singer can sing or a musical instrument can play.

Score/Song: A score or a song is an interplay of voices. It is the final product of music that is delivered to the audience. Songs are of different types based on cultural context and complexity (for example an orchestra is a large number of voices being coordinated by a conductor. In contrast a folk song might be played by a single person on a guitar or a duet between a vocalist and an instrumentalist).

Interval: An interval is the relative semitone distance between any two notes. Intervals are categorized as melodic(semi tone distance between successive notes in a melody) and harmonic intervals (semi tone distance between notes within a chord).

3.2 General Concepts in Music Theory

On top of the building blocks of music, there are certain generic ideas or concepts on which music is based. The following sections describe them :-

Polyphony/Homophony: A homophonic song involves exactly one voice in the song. An example would be a single person singing a tune. A polyphonic song is one which involves two or more voices transposed with one another. An example of polyphonic music would be a Western classical orchestra of a rock band performing a chorus.

Phrase: A musical phrase is a subset of the song that has a complete musical sense of its own. One could think of phrases as musical sentences, whereas a voice could be considered a paragraph. As a side effect a musical phrase can be played independently and still be considered as a song albeit an incomplete one.

Meter: The meter of a song is an expression of the rhythmic structure of a song. In context of western classical music, its a representation of the patterns of accents heard in the recurrence of measures of stressed and unstressed beats. Meters dictate the rhythm or tempo in which a song is played.

Key/Tonality: Tonality or key of a song is a musical system in which pitches or chords are arranged so as to include a hierarchy of relation between musical pitches, stabilities and attractions between various pitches. For example if the song is in the key of C, C is the most stable pitch in that song and other pitches like B have a tendency to go towards C (also called resolution of a phrase) to inculcate a sense of completeness. Moreover other pitches in relation to this pitches have various degrees of stability.

Scale: A scale of a song is an ordered set of notes starting from a fundamental frequency or pitch. If viewed ascendingly or descendingly (increasing/decreasing frequency of the pitches respectively) on this ordering, a scale describes a relationship between successive notes and their semitone distances from each other. A scale restricts the set of notes being played once the fundamental pitch is determined.

Scale Degree: Given a scale and a root note, the scale degree for a note is defined as the distance from the root note to that note on the scale, if the notes on that scale are sequentially played from root note progressively towards the other note.

Key Signature: A key signature is a key along with a scale defined for a song (or in other words the fundamental pitch of the scale of the song is the same as the key of the song). A key signature is an expression of coherence for a song as well as a well defined set of notes that can be played for this piece, and as a result a song does not have notes that are outside of this key signature.

Chromatic Music: Chromatic music is any music that does not have a well defined key signature. Alternatively chromatic music can be categorized as music which is in the chromatic scale (chromatic scale is a scale in which all semitones in western music is present). Chromatic music is more difficult to analyze due to its lack of structure.

Melodic Contour: Melodic contour is the "shape" of melody. A melody with pitches going monotonically upward in frequency is called an ascending contour. Similarly a melody going monotonically downwards in frequency is called a descending contour. There are many other kinds of contour in music theory.

Dynamics: Dynamics is a coarse idea which indicate the variety of relative loudness between notes, speed of notes being played across phrases and other such ideas.

Counterpoint: Counterpoint is a musical phenomenon of two or more independent voices being interleaved to produce a rich and more interesting piece of music. Counterpoint pieces sound more interesting than the sum of their parts. Counterpoint is the basic fundamental on top of which orchestral pieces are built.

Chapter 4

Mathematics of Modulo7

The following sections describe the mathematical concepts used and implemented in Modulo7.

4.1 Preprocessing Steps

It might be that the input sources require certain preprocessing for any mathematical model to work. The following sub sections describe certain preprocessing operations that can be done in order to prepare input data to be transcribed into a vector space model.

4.1.1 Tonality Alignment

In order to compare two songs in different keys, the songs must be transposed to one key. This transposition shifts every note by a certain interval (same as the intervalic distance between the keys of the input songs.) This is analogous to correcting a global offset such that similarity measures based on string representations of music can be applied. Mathematically consider Songs S_1 and S_2 with their respective keys being K_1 and K_2. Define intervalic distance between keys as $K_{intervalic} =$

4.1.2 Voice to Melodic Representation Coversion

Given a voice, various instants inside the voice can be either single notes (melodic notes) or chords (stacks of notes). Often in order to apply pure melodic techniques a voice, a conversion is required from a generic voice to a melody. In order to do that, every chord in the voice is replaced by the root note of the chord.

4.1.3 Contourization

A contour is a quantitative representation of the direction of motion of a given voice / melody. Contours are clearly defined for melodies as a concept and hence the preprocessing steps of converting generic voices to pure melodies.4.1.2. There are many different representations of contour in literature and Modulo7 implements the following representations of contour.

Gross Contour : Gross Contour only contains the information of whether the successive notes of a melody goes up or down irrespective of the intervalic distance by which notes go down or up. Notes going up are designated with value 1, notes going down by -1 and notes staying on the same pitch with 0. So in essence the gross contour is a vector of 0's, 1's and -1s with length = number of melodic intervals present in the voice.

Natural Contour : The natural contour of a song is similar to the gross contour with a difference that the intervalic distance between subsequent notes are calculated instead of ignored as in gross contour. Define the gradient between succesive notes N1 and N2 as

4.2 Vector Space Models of Music

In traditional text based information retrieval retrieval systems, documents are indexed and a vector space representation of documents are created. Typical approaches for counting term frequencies or some weighting scheme like Term Frequency-Inverse Document Frequency Approach (TF-IDF). Analogous to text based IR, Music data can also be expressed as a vector space based on the approach taken. Some of these approaches are taken from the SIMILIE[17] but generalized for polyphonic music. Many approaches are novel based on the author's music theoretic studies.

4.2.1 Vector Space Models for Monophonic Music

Certain vector space models are with respect to a single voice. This allows vector space models to be represented as arrays with simple methods of computation that can be applied on top of it. First define pitch as a real number/string depending on context such that at given instant time t_i either frequency p_i is being played or its note representation p_i is being played. With this simple definition of pitch and onset time we can define our vector space models as follows

Pitch Vector: A voice can be expressed as a sequence of pitches $n_i = (p_i, t_i)$ where p_i is the pitch and/or the set of pitches at instant of time t_i. The symbolic representation of music essentially a discretized version of these values from music sources and hence a vector representation can be made. A voice V can be represented as a vector

$$P =< n_1, n_2, ...n_n > \tag{4.1}$$

A similar vector representation could be when the time information is eschewed in favor on only the pitches. This vector is called the raw pitch vector and is denoted as the follows :-

$$R =< p_1, p_2, ..., p_n > \tag{4.2}$$

Pitch Interval Vector: Another way to look at elements is the interval spacing between elements. This is same as the interval concept in the music theory chapter. Mathematically an interval is defined as $\Delta p_i = p_i - p_{i-1}$. And thus an pitch interval vector is defined as

$$PI =< \Delta p_1, \Delta p_2, ..., \Delta p_n > \tag{4.3}$$

Rhythmically weighted Pitch Interval Vector: In order to include the rhythmic information in the pitch interval Vector, define rhythmically weighted pitch as $rp_i = \Delta p_i \times t_i$. Now the rhythmically weighted pitch vector can be represented as

$$RPI =< rp_1, rp_2, ...rp_n > \tag{4.4}$$

$$gr(N_1, N2_2) = Interval_{dist}(N_1, N_2) \tag{4.5}$$

Thus then the natural contour can be defined as:-

$$NC(Voice) = (gr(N_0, N_1), gr(N_1, N_2), ...gr(N_{len_{Voice}-1}, N_{len_{Voice}})) \tag{4.6}$$

where N_i is the i^{th} note in the voice after converting it into a melody 4.1.2

4.2.2 Vector Space Models for Polyphonic Music

Normalized Tonal Histogram Vector: The tonal histogram is a vector or map of 12 distinct intervals present in western music theory and the number of time. Each position in the vector corresponds to the total number of times that interval has occurred in a song. This is the total summation of the intervals over each individual voice. Mathematically define $\Delta P^{voice_j} = \sum_{i=1}^{len(voice)} p_i^{voice_j}$ and for a song $\Delta P^{song} = \sum_{voice_j} \Delta P^{voice_j}$. Define interval fraction as : $\Delta p_i^f = \frac{sum_i \Delta p_i}{\Delta P^{song}}$ where p_i stands for the interval quantity = i. Thus we can define the normalized tonal histogram vector as

$$NTH =< \Delta p_1^f, \Delta p_2^f, ..., \Delta p_{12}^f > \tag{4.7}$$

Normalized Tonal Duration Histogram Vector: The tonal duration histogram is a vector or map of 12 distinct intervals present in western music theory. Each position in the vector corresponds to the cumulative duration for which that interval has occurred in a song.This is the total summation of the duration of intervals over each individual voice for the entire song. Mathematically define $\Delta T^{voice_j} = \sum_{i=1}^{len(voice)} t_i^{voice_j}$ and for a song $\Delta T^{song} = \sum_{voice_j} \Delta T^{voice_j}$. Define durational interval fraction as : $\Delta t_i^f = \frac{\sum_i \Delta t_i}{\Delta T^{song}}$ where the sum in the numerator stands for the cumulative duration for which an interval quantity = i is played. Thus we can define the normalized tonal duration histogram vector as

$$NTDH =< \Delta t_1^f, \Delta t_2^f, ..., \Delta t_{12}^f > \tag{4.8}$$

Normalized Pitch Duration Histogram Vector: The pitch duration histogram is a vector or map of 12 distinct pitches present in western music theory. Each position in the vector corresponds to the cumulative duration for which that pitch has occurred in a voice and for a song it is the summation of cumulative durations over all the voices. Mathematically define $\Delta T^{voice_j} = \sum_{i=1}^{len(voice)} t_i^{voice_j}$ and $\Delta T^{song} = \sum_{voice_j} \Delta T^{voice_j}$. Define durational interval fraction as : $\Delta t_i^p = \frac{\sum_i \Delta p_i}{\Delta T^{voice}}$. Here the summation in the numerator is the cumulative time for which the pitch in the i^{th} position of the western music chromatic scale is played. Thus we can define the normalized tonal duration histogram vector as

$$NPDH =< \Delta t_1^p, \Delta t_2^p, ..., \Delta t_{12}^p > \tag{4.9}$$

4.3 Similarity Measures

Similarity is defined in Modulo7 as a function which takes as input two voices or songs and outputs a value between 0 to 1 where 0 stands for least similar and 1 stands for most similar. Similarity measures are a cornerstone of recommendations and many recommender engines are based on rank similarity measures for different criteria. Mathematically :-

$$Sim_{song}(S_1, S_2) \in (0, 1) \tag{4.10}$$

$$Sim_{voice}(V_1, V_2) \in (0, 1) \tag{4.11}$$

4.3.1 Similarity Measures for Monophonic Music

Similarity measures are different concepts for monophonic and polyphonic music as it stems from comparing different vector representations. For the following sections assume vectors of equal length. In a further section 4.3.3 we extend standard similarity measures to vectors of unequal length.

Edit Distance on Raw Pitch Vector Representation: Consider the raw pitch vector in equation 4.2. This vector is essentially a vector of tokens or equivalently a string. Hence standard edit distance algorithms in normal text IR can be applied to it (e.g Leveinstein Distance, WagnerFischer algorithm etc[18]).

4.3.2 Similarity Measures for Polyphonic Music

In order to incorporate vector space models to polyphonic similarity, monophonic measures can be extended in order to accomodate for polyphony. Another approach would be to apply measures.

Generic maximal voice similarity An approach would be to take pairwise voice similarities between two voices of a song, and then representing the max of these pairwise computed similarities. This model is especially useful in cases where comparing a melody against a song which contains a similar melody. Mathematically

$$GMVS(S_1, S_2, VSim) = arg_{max}(VSim(V_i, V_j))\ s.t\ V_i \in S_1\ and\ V_j \in S_2 \tag{4.12}$$

4.3.3 Similarity of vectors of unequal length

Its almost certain that two voices will never have the same length. Hence its important at this point to ascertain how to map similarity measures to unequal length voices. Moreover, its also important to judge which regions of one melody are maximally similar to which other regions of the other melody (also called as alignment) . Modulo7 takes inspiration from bio informatics domain and uses the smith waterman algorithm modified for voice similarity.[19] The algorithm is as follows:-

4.3.4 Meta Data based similarity

All the similarity measures considered so far is based on similarity on voices and sets of voices in a a song. However there are other global properties of a song such as the key signature or the time signature of a song. This global information can give us context about a song's particular characteristics (for example songs in Minor

```
procedure SMITH WATERMAN VOICE SIMILARITY(V1, V2, INSIM)
    Define WM = Array[len(V1)][len(V2)]
    for i in 1 to len(V1) do
        WM[i][0] = 0
    end for
    for j in 1 to len(V2) do
        WM[0][j] = 0
    end for
    for i in 1 to len(V1) do
        for j in 1 to len(V2) do
            WM[i][j] = max(0, WM[i - 1][j - 1] + InSim(V1(i), V2(j)), WM[i - 1, j] + InSim(V1(i), $\phi$), WM[i, j - 1] + InSim($\phi$,V2(j))
        end for
    end for
    return WM[len(V1), WM(len(V2))] / max(len(V1), len(V2)
end procedure
```

scale are generally sadder than songs in the Major scale). Hence estimates can be more quickly derived by comparing meta data features rather than voices (whose computation). These similarity measures can be used for a additional purposes(for example completing incomplete meta data).

4.3.5 Tonal Similarity

Often pieces of one key are similar to pieces on a different key, simply based on the fact that the keys themselves are similar.

4.4 Criteria Analysis

While Modulo7's primary goal is on comparing similarities between pieces, often its better to ascertain whether a certain piece satisfies a certain music theoretic predicate. Some example problems of such sorts are if the piece has a species 1 counterpoint (i.e. the voices move with the exact same speed) or if the piece has voices in the STAB criteria (with exactly 4 voices and their ranges being in particular range of high and low notes). This allows a consumer to build complex queries based on pieces satisfying selectivity requirements on top of similarity measures or alternatively if certain pieces just satisfies a one or more criteria. Following are the criteria implemented in Modulo7

4.4.1 Simple criteria

Simple criteria are based on simple global level properties of the song.

Polyphonic Criteria: Its a simple criteria which decides whether a piece of music is polyphonic or not. This is decided on the basis of the number of voices in the song.

Key Signature Equality Criteria: Its a simple criteria that checks if a song is in a particular key or not.

4.5 Statistics Analysis

A statistic when applied to a given song outputs a real number. Alternatively statistics could be thought of a non trivial extracted single value features. Mathematically a feature can be defined as:-

$$Criteria(Song) = x \ s.t \ x \in \mathbb{R} \tag{4.13}$$

The following are the features implemented in Modulo7.

Melodic Repeatability Fraction: Given a voice, compute a sub voice that repeats the maximum number of times within the voice and then take the fraction between the length sub voice which satisfies this criteria against the length of the voice. This measure also uses the pre-processing step

Interval Index: An interval index is the fraction of intervals being played in a song divided by the total number of intervals present in the song. These statistics are coarse measures of a song. There are three classes of interval indices :-

1. Happiness Index : The happiness index of a song is the number of major intervals in a song divided by the total number of intervals. A major interval sounds "happy" to a layman hence a higher concentration of them makes a song happier.

2. Sadness Index : The sadness index of a song is the number of minor intervals in a song divided by the total number of intervals. A minor interval sounds ”sad” to a layman hence a higher concentration of them makes a song sadder.

3. Power Index : The power index of a song is the number of perfect interval in a song divided by the total number of intervals. Perfect melodic intervals are very prevalent in a rock and metal songs and are an expression of a neutral/powerful tone.

Chapter 5

Software architecture and Methodology

The following sections present the software architecture and the methodology of Modulo7 and also the limitations of Modulo7

5.1 Server Side architecture

Modulo7 is designed with the purpose of scalability. A block diagram of the components of the server side architecture is presented below :-

1. Source Converter : Converts music sources (e.g. music XML, midi etc) into modulo7's binary representation.

2. Music Theory Models : The model is a description of music theoretic criteria

that can be applied on top of a song. Examples would be melodic contour, tonal histogram etc.

3. Distributed Storage Mechanism : The modulo7 internal representation is a conversion to create a song representation with all the meta data of the song (Key, Scale, etc) along with the sequences of note events stored as lists. This representation is then serialized and stored in and Hadoop Distributed File System. This allows for fault tolerance and a distributed deployment of the input data.

4. Lyrics Indexer : A distributed index of songs lyrics. This acts as a base on which standard techniques for similarity analysis might be applied. Alternatively it can provide a framework on which custom models (e.g. semantic intent of the song, correlation between music theory models and lyrics might also be applied).

5. Lyrics similarity models : A set of similarity models that can be applied to an index.

6. Query Engine : An SQL like interface to a client that allows you to gather and ascertain useful information (based on music theoretic criteria).

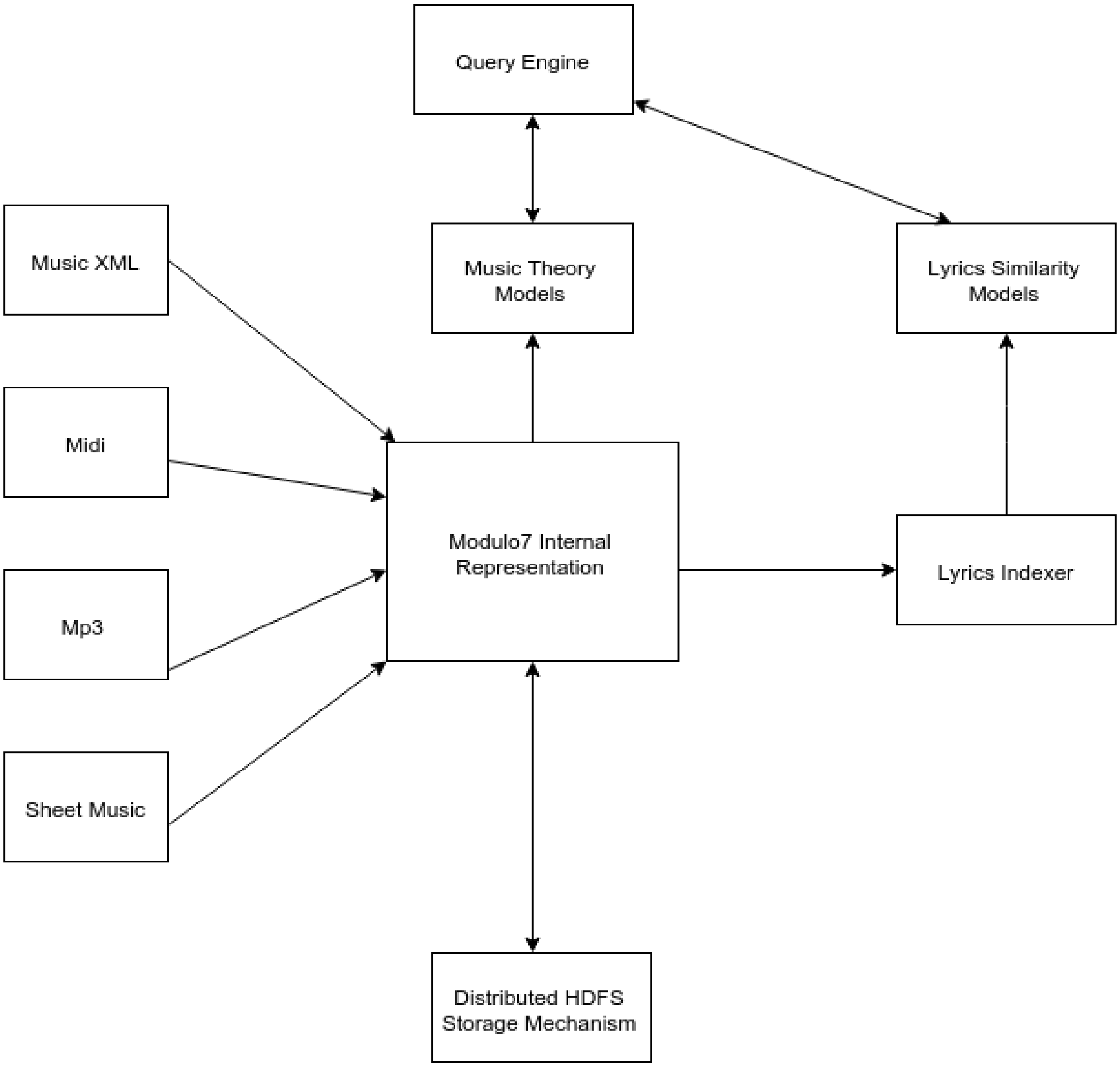

Figure 5.1: Modulo7 architectural design

5.2 Client architecture

The server exposes a sql like interface as well as a consumable API. Some sample queries would be :-

1. select midi files from database where $melodic\_complexity > somethreshold$
2. select * from database where $artist = led\_zepplin\ and\ harmonic\_movement > harmonic\_movement(stairway\_to\_heaven)$
3. select $num\_voices\ from\ Database\ where\ songName = someSong.midi$

An API will also be exposed to the client along a remote invocation procedure. The API would primarily target single sources for specifics. Some example API would be :-

1. int getNumVoices(String midiFilePath)
2. double melodicContourMovement(String pngSheetFilePath)
3. double compareAverageAttack(String musicXMLFile)

This API can be consumed for specific song analysis. As design this API will not work on a bulk of files like its sql counterpart.

Moreover the client also exposes a highly customized search engine based on the custom vector space representation of features extracted by Modulo7.

5.3 Song sources

At the heart of Modulo7's design is its song sources adaptors (or converters) into its own internal binary format. Each music source is a different representation and while certain sources ascribe what how music should be played (e.g musicxml, sheet music), other formats ascribe what is actually being played (e.g midi, mp3). There are many other music sources in existence (e.g guitar tablature, GUIDO format, humdrum format), but for the purposes of breadth and ubiquity, these four sources have been targeted as input for Modulo7. Note that acquiring features from each format is a domain specific challenge and inaccuracies are inherent because of that. The following subsections describe the individual formats in detail and the challenges encountered in parsing them.

5.3.1 Midi format

MIDI (short for Musical Instrument Digital Interface), is a technical specification for encoding of events on a midi enabled instrument and a protocol for interfacing and communicating between various midi enabled instruments. Typically any midi enabled electronic instrument when played, relays to its internal circuitry a message. Examples of such messages could be a particular note is being hit on a keyboard, a note is being hit off after being hit on, tempo based messages on the number of ticks per second etc. While MIDI is a technical specification for encoding music the score is being played, Modulo7 treats it as a symbolic representation of music. Midi was also a simple and popular encoding format for music and gaming industry in the nineteen ninties.

A symbolic representation is a codification of music which acts a higher level of abstraction (individual notes or chords being played) as compared to lower level representations like audio files (which codify information like waveforms). Modulo7's internal representation is also a symbolic representation. Symbolic representations are easier to manipulate when applying a music theoretic criteria.

Midi is one of the easier formats to parse for musical specifications. Moreover there is a big volunteer community of midi encoders. As such acquiring and parsing non trivial amounts of midi data is not a very challenging task.

5.3.2 Western Sheet Music

Sheet music is one of the oldest forms of music in existence. Its a hand written or printed form of music that uses a specific script (a set of musical symbols on a manuscript paper) to ascribe music. Music Composers from Medieval and Modern periods of the western world use western sheet scripting to codify their work while performers play from these sources. A vast body of older work and particularly orchestral work is codified in sheet music.

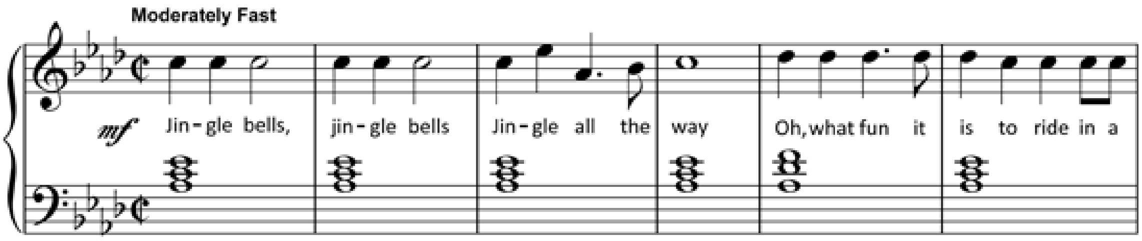

Figure 5.2: Jingle bells melody sheet music representation

Like midi, sheet music is also symbolic in nature. However unlike midi, its an expression of how a score should be played, rather than what is being played. Modulo7 converts digitized versions of these sheet music (e.g sheet music stored .tiff, .png. jpeg etc formats)

Parsing digitized sheet music is an extremely challenging task. It requires a solid understanding on Computer Vision and even the state of the art software in existence today cant handle all scores (especially a poorly digitized formats). Given the amount of domain knowledge required, Modulo7 uses a third party library called Audiveris .

5.3.3 Music XML format

Music XML format is a standard open format for exchanging digital sheet music. A music XML format is unusual as its a format that is easy to parse for computers and easy for humans to understand it. MusicXML formats are heavily used by music notation applications. Music XML format is a symbolic format and can be considered a modernization of the Sheet music format. Its disadvantage however is unlike sheet music, a performer cant read the piece and play it on the spot directly.

Just like Western Sheet music and midi, music XML is a symbolic format as well. Music XML is also a transcription format which specifies how a score should be played.

5.3.4 MP3 format

For the sake of completeness, Modulo7 also supports an audio format called mp3. Its an audio encoding format that uses lossy compression to encode audio data. Mp3 gives a reasonably good approximation to other digital audio formats of music storage with a significant savings in space for storage. Its one of the defacto standards of digital music compression and transfer and playback on most digital audio players.

5.4 Modulo7 Internal Representation

Modulo7 consists of converters that convert data into Modulo7's internal representation. This representation can be thought of a document representation on which similarity measures described in Chapter 4 can be applied to. Moreover the internal representation can be thought of as an indexed meta data structure for any source of song from which relevant information can be acquired. Hence Modulo7 indexing schematic is a symbolic representation of music much like music xml and sheet music. The converters are responsible for converting different music sources to this representation format. Its important to note that depending on there source one or more of the subcomponents of the internal representation may be missing or wrong. Modulo7 indexes songs based on each of these criteria and on top of these boolean queries can be formulated. The components are broadly categorized as the following:-

Song Metadata: The metadata aspects in a song e.g. The name of the song/ the composer/performer's name, Key Signature of the Song, Meter of the Song etc. These are global properties of the song.

Voices in a song: Similar to the Voices in Music theory, Voices in Modulo7 represent the same symbolic data as is present in the sources from which the information is parsed.

Lyrics of a song: The textual representation (along with delimiters for line breaks) for the lyrics of a song. Lyrics can live independently as separate entities (if the input to Modulo7 is a text file containing the lyrics and no other information). However midi/musicxml and sheet music have optional lyrics elements present in their transcriptions and Modulo7 transcribes from those.

In most cases though lyrics exists as a separate entity from songs. In such cases, Modulo7 separately indexes lyrics. In certain datasets, the lyrics representation is different (for example the million song dataset has a representation format as a bag of words with counts of the words occuring for each format[20]). Modulo7 accomodates such formats as well.

5.5 Methodology

This section contains the methodology followed in the information retrieval phase and then the indexing steps taken after the domain specific conversion is completed by Modulo7's adapters

1. Given a root directory, Modulo7 recursively parses all the sheet music image files, mp3, midi and music xml files. Depending on the file type individual parser modules are invoked and an internal representation is created in memory and serialized to disk (depending on user preference)

2. Modulo7 then indexes all the objects created on specific meta data (such as key signature, time signature and artist of a song). Moreover it also creates a lucene index on lyrics extracted. It stores all these indices in memory.

3. Modulo7 then exposes a prompt to the consumer which contains a set of standard querying options along with a SQL like querying interface. Consumer can then choose the option they like and query the constructed database.

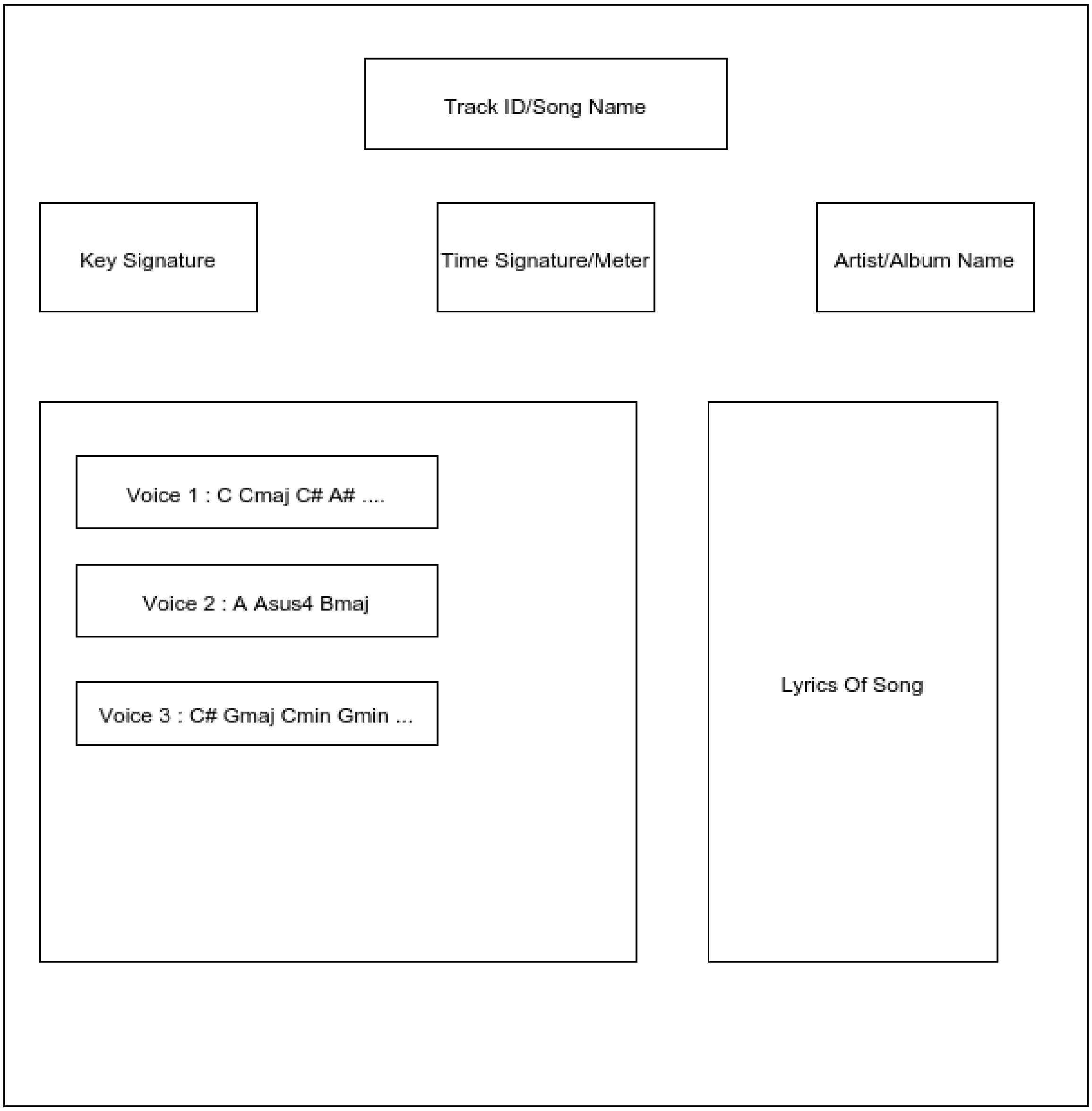

Figure 5.3: Modulo7 internal representation

5.5.1 Modulo7 standard query set

Modulo7 exposes a standard set of querying features to the consumer. These queries are useful to extract simple information from the parsed dataset from Modulo7. The following are the sample queries that can be relevant for a user :-

1. Return all songs that are in the key of CMajor
2. Return all songs that are in in the Minor scale
3. Return a ranked order of lyrics given an input string (for example a verse in the song) while treating lyrics like a normal document
4. Return all songs that are performed by Led Zepplin
5. Return all polyphonic songs in the Database
6. Return all music xml files that have been transcribed with a treble clef
7. Return song with a melodic repetition factor above a given thresh hold
8. Return all songs that are of species two counterpoint.

The simple query framework has limited expressiveness in querying options but is an example set to the user on what can be queried. Modulo7 also exposes a slightly harder to use SQL like query syntax to concatenate boolean expressions of these example queries and more (boolean combinations of all criteria and statistics defined in criteria 4.4 and statistic 4.5 sections)

5.5.2 Modulo7 SQL Language Specifications

On top of the standard set of query set defined as an example set, Modulo7 also supports a custom query language for extracting relevant information from a parsed and indexed data set. This language is similar to SQL but its internal processing is radically different. A generic expression can be expressed as follows

$$\textbf{select input\_src\_list from DATABASENAME where expr\_list} \tag{5.1}$$

Here input list is a set of music sources to be parsed. An expression list is a boolean conjunctive or distinctive list that outputs a subset of the original set of songs in the database. The definitions for the terms are as follows:-

1. **input_src_list** : An argument list of all the acceptable formats of is any combination of songs : midi, musicxml, sheet and mp3. This clears out all the formats that are irrelevant to the consumer.

2. **DATABASENAME** : The name of the Modulo7 Database. Its an internal consistency check to determine if the consumer is querying against the right Modulo7 database.

3. **expr_list** : A conjunctive and/or disjunctive list of boolean queries on statistics and criteria defined in sections 4.4 and 4.5. This allows for a greater degree of customization as compared to the other frameworks in literature as well as

expose a structured query language for querying (which is sorely lacking in other frameworks). The elements of the expr_list are defined as follows:-

(a) **criteria is or is not true** : Returns a subset of songs from a candidate state which either satisfy or dont satisfy a given criteria. The argument criteria is replaced by an implemented criteria in 4.4

(b) **statistic relational_op doubleValue** : Returns a subset of a songs from candidate set which satisfy this criteria : When a statistic is applied on a song in a candidate set, the returned value of the statistic satisfies a relational operation to the given value. The arguments to this expression is a statistic implemented in 4.5, a relational operator and a double value.

(c) **statistic between value1 and value2** : This form is a range query. This query returns the subset of songs from a candidate set which satisfy this criteria : When a statistic is applied on a song in a candidate set, the returned values lies in between value1 and value2.

Each of these basic query component returns a subset of songs that satisfy the query component. These query components can be concatenated conjunctively or disjunctively to form a boolean query. So a query is effectively $Q = \cup_i | \cap_i (qc)$, where qc is a query component described above and Q is the resultant query.

5.5.3 Modulo7 Similarity Engine

On top of Modulo 7 supporting custom queries, it also acts in a ranked search engine mode. However the ranking model of the search engine is based on similarity measures based on the structural analysis of the music sources and are described in 4.3

5.5.4 Modulo7 Lyrics Analyzer Architecture

The modulo indexer also indexes lyrics, but treats lyrics objects as text components. So the standard model of text Information Retreival techniques can be used type analyze lyrics. Modulo7 implements lyrics indexing and standard NLP operations on lyrics.

1. Modulo7 parses lyrics components from some of its sources (for example musicxml and midi have embedded lyrics structures inside it). This is stored along with the song object

2. Modulo7 also parses independent lyrics structures provided to it. This allows for increased flexibility for Modulo7 to just parse lyrics objects

3. Modulo7 creates a lucene index of the lyrics objects once parsed from its sources. This allows for users to make standard text queries via Lucene.

   Modulo7 also provides support for rudimentary Natural Language Processing operations on top of the lyrics obtained. Two supported operations for lyrics are :-

1. **Language ID** : Modulo7 can detect what language the song's lyrics is written in. It does this via an language ID call to alchemy A.7.

2. **Sentiment Analysis** : Modulo7 can detect the positivity or negativity sentiment of a song's lyrics and assigns a score to it (with -1 standing for highest degree of negativity and similarly 1 standing for highest degree of positivity).

On top of these features, the lyrics analyzer provides support for meta data (also called as tag) prediction. Given a data set with tags annotated to songs along with lyrics, Modulo7 can predict tags for new input lyrics. The following tag estimation schemes are implemented in Modulo7:-

5.5.5 Naive Tag Membership

Consider $T(S_i)$ be defined as the set of tags for the S_i which is the i^{th} song in the music match dataset. Let S_{new} be a new song for which tags need to be predicted and get L(S) represent the lyrics of a song. Hence L_{new} should be similar to some $L(S_k)$ for their tags to be similar. Let $S_{sim} = S_i | isSim(S_i, S_new) = true$ be the set of all the songs similar to S_new. We define $T_{new} = \{\cup\ T(S_i) \mid S_i \in S_{sim}\}$. In other words the tags of the new song is the union of the tags in the songs similar to the new song. This estimation scheme is called the **naive tag estimation**.

This scheme does not capture the strength of similarity a song has to another, or the relative prevalence of tag membership to a song. In order to address this issue, we introduce two non trivial schemes.

5.5.6 Weighted Tag Membership

In the previous scheme, there are no considerations for the weights associated with the tags. In order to accommodate we assume the existence of tag weights associated for tags in the song meta data. set $WT(S_i)$ be defined as set of tag weight pairs (tag with associated weight) for a given song and w_k is the weight of k^{th} tag. Let $WS_{sim} = sort_v(S_i, \alpha_i WT(S_i) | isSim(S_i, S_{new}) = v)$ be the weighted set of Song and tag pairs where α_i is a scaling factor which is defined $\alpha_i = \frac{sizeof(WT(S_i))-i}{sizeof(WT(S_i))}$. α_i is a fraction which is multiplied with every weight associated with tags of song S_i.

This scheme takes into account both the rank of the songs in based on a similarity metric and and also the weight associated with each individual tag. The scheme can retain only a subset of the maximal weighted tags in the resulted weighted tag set for the input song.

5.5.7 Most frequently occurring tags

In the previous scheme, the frequency of tags occurring inside the dataset is ignored. In order to accommodate that let the frequency of tags along with the tags for a song be defined as $FT(S_i)$. Let $f_k(S_i)$ be the total frequency of tag k for the set S where S = $S_{sim} = S_i | isSim(S_i, S_new) = true$. Hence we can define the set of estimated tags as $FT_{sim}(S_i) = sort_{f_k(S_i)}(k)$

5.6 Limitations of Modulo7

While Modulo7 attempts to solve a large set of problems (custom query specification based on structural aspects of music, similarity analysis, logical indexing scheme), there are some fundamental limitations to what Modulo7 can or cannot do. Some of the notable ones are listed as follows:-

1. Modulo7 does not perform any kind of timbral analysis. This limitation is by design, since all formats of music do not convey timbral information faithfully (for instance sheet music is a specification of music to be played and not an actual recording), hence Modulo7 has not been designed with timbral analysis in mind.

2. Modulo7 does not take into account varying time and/or key signatures. This is due to the fact that in most western music, these two global parameters stay constant for songs.

3. Modulo7 assumes input mp3 files are monophonic. This is due to the fact that the state of the art in audio processing techniques have not solved the problem of polyphonic symbolic transcription faithfully[10]

Chapter 6

Experimental Evaluation

For the purposes of evaluating Modulo7, test cases have been designed into two formats. One category of testing is micro testing, for validating correctness and precision recall for small sets of data. This ensures verifiability of algorithms and similarity measures on small datasets as well as novel explorations of data. Most MIR research is done on small scale datasets and hence falls in the purview of micro testing. The other format is macro testing which involves large datasets such as the million song dataset.[20]

A few assumptions that are made in testing are as follows :-

1. In order to estimate ground truth values, the author assumed ground truth values presented in datasets used / or subjective judgments about which songs are similar to each other. These subjective judgments are procured from existing

literature.

2. If the song meta data (such as key-signature, time-signature, total duration of song) is not encoded, its estimated by the individual parsers for the data source. This estimation id done by existing algorithms in literature. However if meta data is encoded in the input, its assumed to be correct and no such estimations are carried out.

3. Most tests are against file formats of the similar types (for example midi is tested against other symbolic files). This is due to the inherent complexity of symbolic decoding of audio formats like mp3. Also its easier to compare symbolic data against other symbolic data.

4. In the event of parsing data, there can be legal issues (e.g. the song can be copyrighted). For that reason custom parsers to build alternate research datasets (e.g the million song dataset has already derived features that Modulo7 intended to derive for Mp3 files and has its own parser written by the authors.[20])

5. All evaluations are done against research datasets which are published in academia or exposed as public data sets in industry. As such no proprietary data sets are used for the purpose of any evaluation metric.

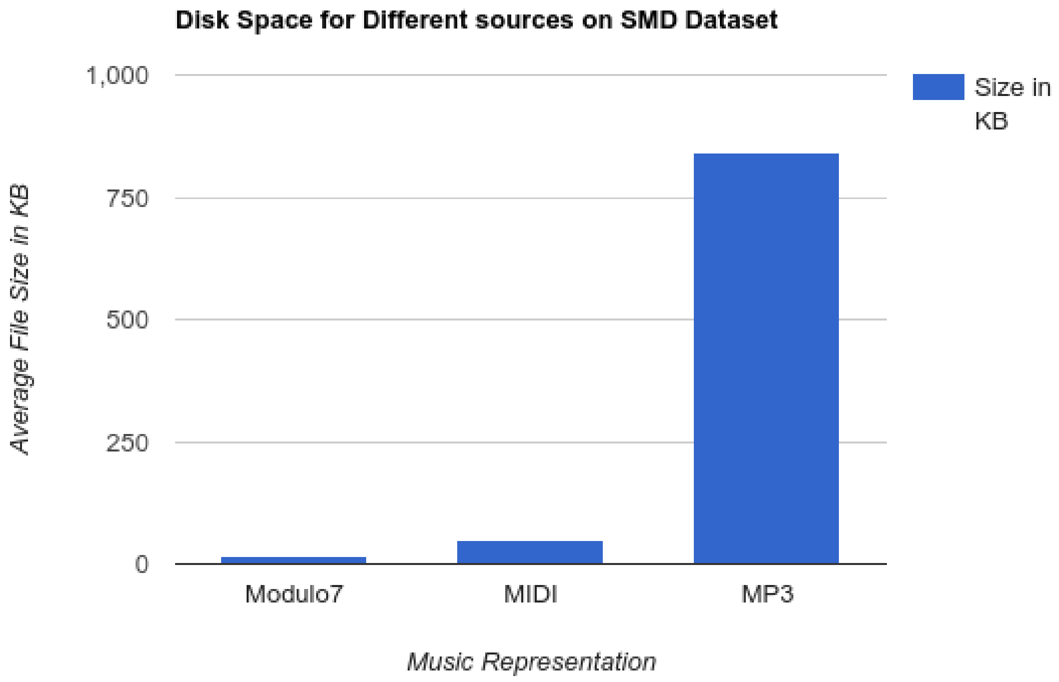

Figure 6.1: Modulo7 architectural design

6.1 Results of Index Compression

The Modulo7 representation can be thought of an indexed meta data version of the song with the symbolic information of the song intact. True to all indexed data, Modulo7 represents the song in a much smaller size than the original source. The following chart demonstrates the average compression of indexed data as compared to source files on the Saarland Music Data (SMD) Dataset:^{21}- As expected Modulo7's serialized format expresses a song in less disk space than its source formats while keeping the symbolic information intact. The results are positive as there is a 4 time decrease in size of expressing symbolic information as compared to midi files.

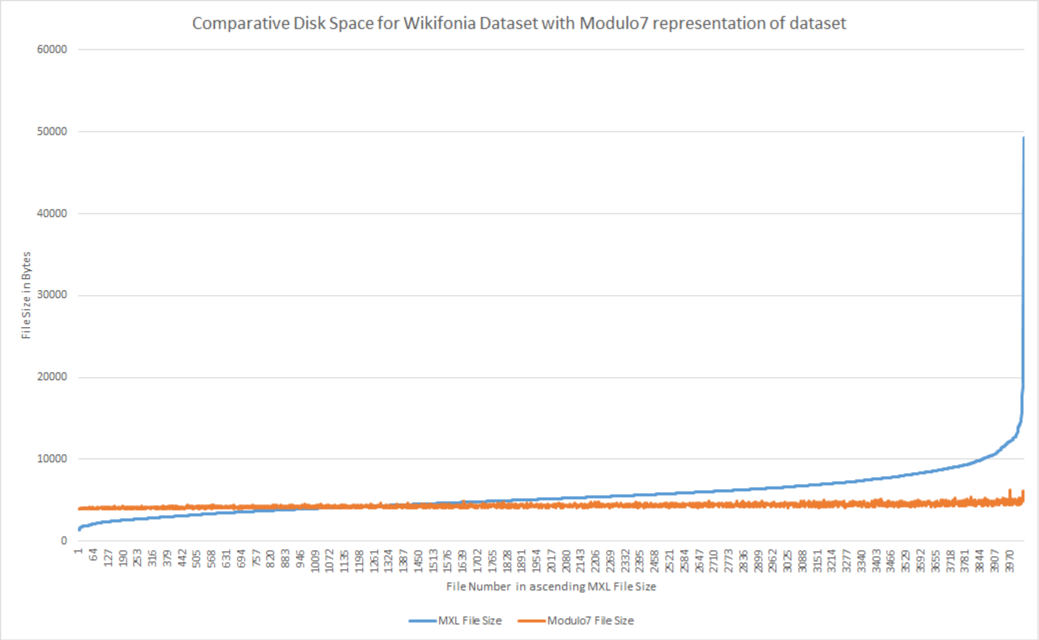

Figure 6.2: Modulo7 comparative file sizes

A similar transformation was also done on a direct download able subset of the wikifonia dataset in order to compare Modulo7 internal representation against the compressed xml representation of the wikifonia dataset. A plot of disk space requirements are plotted in ascending order:-

6.2 Results on similarity measures

This set of experiments determine the precision and recall values for the similarities defined in 4.3 on ground truth data extracted from.[20] Modulo7 does not claim to improve on the state of the art when it comes to similarity metrics or does not intend to create a new similarity metric. Rather this set of experiments are a test of efficiency in execution and accuracy of existing methods on large scale datasets. The million song dataset was chosen for experimental evaluation[20] for pre computed symbolic transcriptions of mp3 data and the last fm data set for ascertaining similar songs to build a ground truth for evaluation. Due to the constraints of hardware for evaluation, we took a scaled down subset of the original 584,897 songs to a more manageable 10000 songs, with similar songs edited to fit this dataset. The songs were chosen randomly from the dataset with the only criteria being that they are monophonic(since polyphonic transcription from audio files is not a fully solved problem[10]). As a consequence only the monophonic similarity measures are used for these experiments.

6.3 Results on KK Tonality Profiles algorithm

In order to test the KK Tonality algorithm given in B.1, Modulo7 is benchmarked against a big subset of the Wikifonia data set of lead sheets in the compact mxl format (variant of the music xml format).[22] The original dataset of the Wikifonia is now no longer available but a sizable subset of 6715 songs are currently downloadable and copyright free. Out of this set, 1314 have key signatures embedded in the song sources. The experiment involves comparing the key signatures embedded inside the key signatures versus the implied key signatures the KK Tonality estimates from the pitch histogram of the songs parsed from this source. A special MXL parser (a minor variant of the music xml parser) was developed for this purpose. The scoring scheme for this experiment was simple, if the key signature was correctly identified then score of 1 otherwise score of 0. In this particular dataset, key signatures are partially known (**since the number of sharps or flats in the key signatures are always encoded in music xml files** so only relative major/minor are needed to be ascertained). As a consequence only two choices are to be made between key signatures for each file giving a baseline of 50 percent. In this particular example, KK Tonality's performance is how well it can distinguish between relative minors and majors.

After running the KK Tonality algorithm on the wikifonia dataset, 1129 out the total 1314 key signatures are correctly identified leading to an accuracy of 85.9 percent. This is commensurate with the reported accuracies in.[23] The novelty of this experiment stems from the fact that KKTonality profiles algorithm was not successfully run against a large scale database successfully in literature.

6.4 Results on lyrics similarity and statistics analysis

On top of the experiments done for Song sources incorporating tonal information, there were specific experiments that were carried out for lyrics similarities in general. The ground truth for these experiments is the musix match lyrics dataset present in the million song data set.[20] The dataset decomposes lyrics into bag of words formats (the frequencies of the top 5000 words in lyrics) along with bag of words representation of 210,519 lyrics of songs. This dataset acts like a great baseline for set based similarities of lyrics. The experiment involved calculating the expected word count from the ground truth data and with that form a basis for comparing songs with the ground truth data. There are measures defined in literature[24] which define similarity and accuracy of lyrics based on expected counts of words and observed counts of words in lyrics. However for this experiment we have decided to extract the tags from the last fm dataset of the Million Song Data set[20] to acquire the tags that occur frequently for a given song and then build a predictive model that outputs the tags for a newly seen song.

These tags could indicate the language, genre etc of the songs. As such this comparison allows for predicting various meta data regarding a newly seen lyrics of a song and allow for building a model that predicts tags based on lyrics similarities.

Out of the 210,519 songs with lyrics provided in the million song data set, 2296 have tags extracted for them in the dataset, so this set of songs are considered the ground truth for estimating tags for novel lyrics. The lyrics in this dataset are in the bag of words document representation format and hence standard similarity measures like cosine and dice similarity can be used for similarity measurement between lyrics. The lyrics in the million song dataset are already stemmed via the Porter stemmer[20]

In order to estimate the accuracy of the tag prediction models, the extracted data was divided into 10 percent test data and 90 percent training data and 10 fold cross validation was performed. Each lyrics in the test data was compared to the training data and a ranked order of the trained songs are presented based on the similarity metric used. Tags are then estimated based on the tag estimation mechanisms presented in 5.5.4. The similarity metrics for comparing lyrics that were chosen for this experiment were standard document similarity measures that are word transpose invariant (cosine similarity, dice similarity etc). In order to estimate the degree of agreement of the estimated tag set with the observed tag set for a given song, we use the Jaccard Similarity which is defined as

$$J(T_{est}, T_{obs}) = \frac{\mid T_{est} \cap T_{obs} \mid}{\mid T_{est} \cup T_{obs} \mid} \tag{6.1}$$

The following table lists out the observed average agreement over all songs of the test set:-

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

6.5 Result on Memory and Disk space usage against jMIR

In order to compare the memory and disk space requirements, Modulo7 was tested against its closest competitor jMIR's[1] jSymbolic component. Both frameworks are written in Java and both involve extraction of features. However jMIR is more exhaustive in what features it extracts so only a subset of those that are also extracted by Modulo7 are considered.

Chapter 7

Conclusions

Appendix A

Third Party Libraries Used

Modulo7 is a significant software engineering effort. This is partly due to the fact that Modulo7 tends to address speed related issues that are prevalent in other frameworks and partly due to the disparate sources of music that it supports. As such Modulo7 utilizes a number of third party libraries in its operations. These libraries and their roles are mentioned below:-

A.1 Apache Lucene

Apache Lucene is a full text search engine library written in Java. Apache Lucene is used for indexing text documents, spelling correction and other such functionality. In context of Modulo7, Apache Lucene is used to maintain inverted indices of lyrics either independently acquired from text files containing lyrics or from emdedded lyrics in the Modulo7 supported sources.

A.2 Apache Avro

Apache Avro is a serialization library used to store Modulo7 objects to disk. This allows for faster retrieval of parsed objects instead of having to reparse entire song sources again and again.

A.3 Echo Nest jEN API

The toughest challenge in all of Modulo7 was to parse symbolic information from audio sources. In order to accomplish this, Modulo7 relied on the Echo Nest's client library to convert mp3 files into chromagram representation of music.[25] The chromagram representation is acquired directly by converting mp3 representation into the frequency domain by Echo Nest. Modulo7 treats this process as a black box, as it is interested in finding out only the chromagram representation (from which identifying notes and chords become much simpler).

A.4 Antlr

Antlr (Another language recognition tool) is a framework used to develop lexers and parses for custom programming languages. In case of Modulo7, Antlr was used to develop the Modulo7SQL Custom query language.

A.5 Jsoup

Jsoup is a library used for parsing XML documents written in Java. In case of Modulo7, Jsoup is used to parse music xml documents and present song representations to the Modulo7 engine.

A.6 Audiveris

Audiveris is a OMR (Optical Music Recognition System) written in Java which converts digitized sheet music files into musicxml files. Audiveris is used to parse sheet music files into Modulo7 song representations.

A.7 Alchemy

Alchemy is an implementation of NLP(In general AI) as a service model by IBM. Alchemy provides support for language ID, semantic analysis of arbitrary documents and text. In Modulo7, Alchemy is used for analyzing lyrics.

A.8 Apache JCS (Java Caching System)

Apache JCS is used as a distributed in memory cache to cache the results of Modulo7 custom queries and similarity results for the queries made in the past.

Appendix B

Algorithms in use in Modulo7

There are certain algorithms in literature that are directly implemented in Modulo7. These algorithms facilitate the smooth functioning of Modulo7's indexing in face of incomplete metadata. Some notable algorithms that have been used are briefly described in the following subsections

B.1 KK Tonality Profiles and a Key Estimation Algorithm

Many music sources have the key signature inscribed in it. For example a midi file might have the key signature bytes transcribed. In the event that this information is not present, it must be inferred from the recording. This is required for certain similarity measures that need the key signature of the song for preprocessing steps in particular for tonality alignment (4.3.3). There are many methods for achieving this including non trivial tree representations of polyphonic music to estimate key.[26] However in Modulo7, the author has implemented a simpler model for tonality estimation based on templates called KK tonality profiles[23]

The premise of the KK tonality profile stems from experiments done in[23] and[27] which estimate how likely a user is to ascribe a note to a series on notes played on a melody or an incomplete harmonic element in different keys. The notes guessed correlate to the relative prominence of a note in a given key(what this the frequency and total duration a note is played in a song in a given key). After many experiments, the experimenters collected the aggregate duration for each note for each key. This experiment was repeated for all 12 major and 12 minor keys. They were able to acquire 24 profiles (vectors of real numbers) which represent a quantitative measure of the key. For example the profiles for C Major and C Minor are respectively.[27]

$$CMajor =< 6.35, 2.23, 3.48, 2.33, 4.38, 4.09, 2.52, 5.19, 2.39, 3.66, 2.29, 2.88 >$$

$$CMinor =< 6.33, 2.68, 3.52, 5.38, 2.60, 3.53, 2.54, 4.75, 3.98, 2.69, 3.34, 3.17 >, \tag{B.1}$$

The profiles of the other keys can be achieved by rotating the vector by the intervalic distance of the root notes of the key and root note their reference Key(CMajor for major keys and CMinor for minor keys).

The key estimation algorithm leverages the kk tonality profiles as input. The algorithm is as follows:-

```
procedure Predict Key Signature(song)
    Define CMaj and CMin as per eqn B.1
    Define MajProf and MinProf = []
    MajProf.add(CMaj) and MinProf.add(CMin)
    Define prev_Key = C
    for key in western keys [D to B] do
        MajProf[key] = left_shift(MajProf[prev_Key])
        MinProf[key] = left_shift(MinProf[prev_Key])
        prev_Key = key
    end for
    song_Pitch_Hist = compute_song_tonal_histogram(song) as per 4.9
    best_Key = CMin, best_Corr = $-\infty$
    for key, maj_prof in MajProf do:
        if correlation(maj_prof, song_Pitch_Hist) ¿ best_Corr then
            best_Key = key
            best_Corr = correlation(maj_prof, song_Pitch_Hist)
        end if
    end for
    for key, mij_prof in MijProf do:
        if correlation(min_prof, song_Pitch_Hist) ¿ best_Corr then
            best_Key = key
            best_Corr = correlation(min_prof, song_Pitch_Hist)
        end if
    end forreturn best_Key
end procedure
```

B.1.1 Chord Identification from Chromagram

A chromagram[25] is a representation of a song in frequency domain with relative intensities of notes in a short window frames of analysis in songs. This chromagram representation is central to acquiring symbolic description from audio sources. Once a chromagram is acquired, ascertaining chords in it becomes important(in particular because harmonic elements are non trivial to ascertain in a given chromagram). Modulo7 implements an algorithm described in^2 in order to detect chords in chromagrams. This procedure is based on chromagram templates of different chords and correlation of current chromagram with these templates.

Bibliography

[1] C. McKay, *Automatic Music Classification with jMIR.* Montreal: McGill University, 2010.

[2] M. D. P. Adam M. Stark, "Real-time chord recognition for live performance."

[3] P. G. Tzanetakis, "Marsyas a framework for audio analysis," *Organized Sound*, vol. 4(3).

[4] D. M. Klaus Frieler, "The simile algorthms for melodic similarity."

[5] "The humdrum toolkit: Reference manual. menlo park, california: Center for computer assisted research in the humanities, 552 pages, isbn 0-936943-10-6." p. 552 pages.

[6] I. F. Karl MacMillan, Micheal Droettbroom, "Gamera: Optical music recogntion in a new shell."

[7] D. M. N. Scaringella, G. Zoia, "Automatic genre classification of music content: a survey."

[8] K. F. Daniel Mllensiefen, “Melodic similarity: Approaches and applications,” in *Proceedings of the 8th International Conference on Music Perception and Cognition, Evanston 2004).*

[9] L. Cherubini, *A Treatise On Counterpoint and Fugue.* Novello, Ewer And Co, 2010.

[10] D. E. G. R. J. Salamon, E. Gomez, “Melody extraction from polyphonic music signals,” in *IEEE Signal Processing Magazine pp 118 - 134*, 2014.

[11] J. S. epnek, “Musical sound timbre: Verbal description and dimensions,” in *Proc. of the 9th Int. Conference on Digital Audio Effects.*

[12] T. Jehan, *Creating Music by Listening.* Massachusetts Institute of Technology: Media Arts and Sciences, 2005.

[13] L. S. Levy. (2013) The lester s. levy sheet music collection. [Online]. Available: http://levysheetmusic.mse.jhu.edu/

[14] S. S. o. M. McGill University. Distributed digital music archives and libraries lab. [Online]. Available: https://ddmal.music.mcgill.ca/

[15] P. Knees and M. Schedl, “A survey of music similarity and recommendation from music context data,” *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 1, pp. 2:1–2:21, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2542205.2542206

[16] A. L. chun Wang and T. F. B. F, “An industrial-strength audio search algorithm,” in *Proceedings of the 4 th International Conference on Music Information Retrieval*, 2003.

[17] D. M. . K. Frieler, *The Simile algorithms documentation 0.3*, 2006.

[18] G. Navarro, “A guided tour to approximate string matching.”

[19] J. D. Frey, *FINDING SONG MELODY SIMILARITIES USING A DNA STRING MATCHING ALGORITHM.* Ohio: Kent State University, 2008.

[20] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[21] M. Müller, V. Konz, W. Bogler, and V. Arifi-Müller, “Saarland music data (SMD),” in *Late-Breaking and Demo Session of the 12th International Conference on Music Information Retrieval (ISMIR)*, Miami, USA, 2011.

[22] Wikifonia. (2013) The wikifonia lead sheet collection. [Online]. Available: http://www.synthzone.com/files/Wikifonia/Wikifonia.zip/

[23] S. T. Madsen, G. Widmer, and J. Kepler, “Key-finding with interval profiles.”

[24] S. D. Robert Macrae, “Ranking lyrics for online search,” in *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR 2012).*

[25] B. Pardo. (2014) Northwestern university chromagram tutorial. [Online]. Available: http://www.cs.northwestern.edu/~pardo/courses/eecs352/lectures/MPM14-Chromagrams.pdf

[26] D. Rizo, J. M. Iñesta, and P. J. P. de León, “Tree model of symbolic music for tonality guessing,” in *Proceedings of the 24th IASTED International Conference on Artificial Intelligence and Applications*, ser. AIA’06. Anaheim, CA, USA: ACTA Press, 2006, pp. 299–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=1166890.1166941

[27] D. Professor, *Cognitive Foundations of Musical Pitch*, ser. Oxford Psychology Series. Oxford University Press, USA, 1990. [Online]. Available: https://books.google.com/books?id=aJDEVqyArr4C