Modulo7 : A Full Stack Music Information Retrieval and Structured Querying Engine

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, 2016

© Arunav Sanyal 2016

All rights reserved

Abstract

Music Information Retrieval (MIR) is an interdisciplinary science of extracting non trivial information and statistics from different sources of music. In today's computerized age, music is stored in a variety of digitized formats - e.g midi, musicxml, mp3, digitized sheet music(in the form of images in .png and .jpeg formats) etc. Music Information Retrieval(MIR) systems aim at extracting features from one or more of these sources. MIR research helps in solving problems like automatic music classification, recommendation engine design etc.

In this thesis, the author proposes and implements a new Music Information Retrieval and Structured Querying Engine called Modulo7. Unlike other MIR software which primarily deal with low level audio features,[1] Modulo7 operates at a higher abstraction level, on the principles of music theory and a symbolic representation of music(by treating musical notes instead of acoustic pitches as the basic blocks of representation of musical data). Modulo7 is implemented as a full stack deployment, with server components that parse various sources of music data into its own efficient

ABSTRACT

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), along with a traditional search model based on non trivial similarity metrics for symbolic music. Modulo7 also implements a lyrics analyzer, which supports functions such as lyrics similarity and meta data prediction (e.g genre prediction).

Primary Reader: Dr David Yarowsky

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 Natalie Draper in the Peabody Conservatory, my instructor for music theory for teaching me the basics of the subject. I would like to thank Dr Cory Mckay from McGill University for his help with understanding concepts in symbolic Music Information Retrieval and helping me with the implementation specifics for midi processing, Dr Ichiro Fuginaga from McGill University for his guidance and help with Optical Music Recognition concepts and Dr Dan Ellis from Columbia University on helping me access and set up the million song data set.

I would like to thank my fellow graduate students Aakash Bhambhani for his key insights in ground truth estimation in experiments, Ankit Garg for his immense help in the application of genomics alignment algorithms in Modulo7, Satya Prateek for his deep expertise Natural Language processing which allowed me to finish the Lyrics

analyzer portion of the thesis and my friend Japneeth for his unending support and encouragement.

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

Dedication

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

Contents

Abstract **ii**

Acknowledgments **iv**

List of Tables **xiii**

List of Figures **xiv**

1 Introduction **1**

2 Literature Review **5**

2.1 Current MIR Software 5

2.1.1 jMIR 6

2.1.2 Marsyas 6

2.1.3 SIMILIE 7

2.1.4 Echo Nest APIs 7

2.1.5 Humdrum 8

2.1.6 Gamera 8
2.1.7 Audiveris 9
2.2 Music Representation Formats 9
2.3 Typical problems of MIR 10
2.3.1 Music Classification / Genre Identification 10
2.3.2 Music Similarity Analysis 11
2.3.3 Automated Musicological Research 11
2.3.4 Audio processing and feature extraction 11
2.3.5 Intelligent Music Archiving and Retrieval 12
2.3.6 Music Recommendation 12
2.3.7 Audio Fingerprinting and Song ID 13

3 Basics of Music Theory **14**
3.1 Building Blocks of Music 15
3.2 General Concepts in Music Theory 18

4 Mathematical Formulations and Models **22**
4.1 Basic Notation 22
4.2 Preprocessing Steps 23
4.2.1 Key Transposition 23
4.2.2 Voice to Melodic Representation Conversion 24
4.2.3 Contourization 25

4.3 Vector Space Models of Music 26
4.3.1 Vector Space Models for Monophonic Music 27
4.3.2 Vector Space Models for Polyphonic Music 28
4.4 Similarity Measures 30
4.4.1 N-gram Similarity Measures 31
4.4.2 Similarity Measures for Monophonic Music 32
4.4.3 Similarity Measures for Polyphonic Music 32
4.5 Sub melodic similarities and Tonal Alignment 33
4.6 Criteria Analysis 34
4.7 Statistics Analysis 36

5 Software architecture and Methodology 38

5.1 Server Side architecture 38
5.2 Client architecture 41
5.3 Song sources and Parsers 41
5.3.1 Midi format 42
5.3.2 Western Digitized Sheet Music 42
5.3.3 Music XML format 44
5.3.4 MP3 format 44
5.4 Modulo7 Internal Representation 45
5.5 Methodology 48
5.5.1 Modulo7 standard query set 48

5.5.2 Modulo7 SQL Language Specifications 49
5.5.3 Modulo7 Similarity Engine 51
5.5.4 Modulo7 Lyrics Analyzer Architecture 52
5.6 Lyrics Based Genre Estimation 53
5.6.1 Naive Genre Estimation 53
5.6.2 Weighted Genre Estimation 54
5.6.3 Max Frequency Tag Estimation 54
5.7 Meta data Estimation 55
5.8 Limitations of Modulo7 55

6 Experimental Evaluation **57**

6.1 Results of Index Compression 59
6.2 Million Song Dataset Experiments 61
6.2.1 Results on Melodic Similarity Analysis 61
6.2.2 Results on lyrics similarity and genre estimation 64
6.2.3 Results on exploratory query analysis 67
6.3 Results on KK Tonality Profiles algorithm for Key Estimation 69
6.4 Results on CPU and Memory and Disk space compared against jMIR 70
6.5 Results on melodic alignment and similarities over sub melodies . . . 74

7 Conclusions and Recommendations **76**

7.1 Conclusions 76

7.1.1 Conclusions on the Query Engine Implementation 78

7.1.2 Conclusions on the Similarity Search Engine Implementation . 78

7.1.3 Conclusions on Scalability and Speed 79

7.2 Recommendations for future research 80

7.2.1 Complete Music Models frameworks 80

7.2.2 Scalability Enhancements 81

APPENDICES **82**

A Software Engineering Aspects **82**

A.1 Third Party Libraries Used 82

A.1.1 Apache Lucene 83

A.1.2 Apache Avro 83

A.1.3 Echo Nest jEN API 83

A.1.4 Antlr 84

A.1.5 Jsoup 84

A.1.6 Audiveris 84

A.1.7 Alchemy 85

A.1.8 Apache JCS (Java Caching System) 85

A.1.9 Apache Commons IO and Math 85

A.1.10 JFugue 85

B Algorithms in use in Modulo7 **86**

B.1 Key Estimation Algorithm 86

B.2 Symbolic Transcription from Chromagrams 88

Bibliography **91**

Vita **98**

List of Tables

6.1 Average Precision and Recall for Melodic Similarity Measures 63
6.2 Results for the exploratory query analysis 68

List of Figures

5.1 Block diagram of Modulo7 software architecture 40
5.2 Jingle bells melody sheet music representation 43
5.3 Abstract representation of the Modulo7 internal representation 47

6.1 Modulo7 SMD Dataset compression 59
6.2 Modulo7 comparative file sizes 60
6.3 Precision Recall Curve for Weighted Genre Estimation 66
6.4 ROC curve for max frequency and naive genre estimation 66
6.5 Modulo7 vs jSymbolic for time taken to generate features 72
6.6 Modulo7 vs jSymbolic for average memory utilized 72
6.7 Modulo7 vs jSymbolic for maximum CPU utilized utilized 73

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 of the song or just a particular melodic pattern that they find catchy? More so if a person likes a song, can we predict if he/she will like a similar song? If yes, then how is this similarity judged?

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. Likewise, 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. Industry has predominantly favored approaches that look at user preferences and history as a basis of prediction and recommendation. For example Amazon Music recommendation works on consumer behavior (user's shopping, browsing history and related consumer behavior[2]). Pandora on the other hand utilizes musicologists to ascertain how a song is similar to another song and creates software that leverages this ad-hoc generated graph of similarity.[3] 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 have extensively extracted features from music sources[4] and mined cultural information on the web but leave it on the consumers to determine how best to leverage this extracted data. Hence symbolic MIR is not traditionally used in industry and music theory is an after thought in almost all industry applications.

Academia on the other hand attempts to solve very particular problems in MIR.

Typical examples would be cover song detection,[5] processing information via signal processing, audio feature extraction, optical music recognition[6] 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[7] (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 exists in academia which can satisfy the scale of industry applications.

This work is an attempt to bridge both communities. Modulo7 is a full stack deployment of Music Information Retrieval Software, providing both a server architecture, a SQL like client and a search engine functionality to find relevant songs on music theoretic criteria. Modulo7 does not attempt to solve very complex music theoretic problems (e.g study orchestral music to identify counter point class). Rather Modulo7 acts a framework on which such analysis can be built upon. Most importantly, Modulo7 addresses the issue of scale and allows for a fast and efficient comparison between songs. It also addresses deficiencies in existing software, such as predicting incomplete meta data information in music sources. Particular examples for this would be Key estimation, Tempo estimation etc.

Modulo7 implements a unique indexing scheme and a universal "document" rep-

resentation of music. This indexing scheme involves creating an inverted index for global properties of songs (key signature, the property of homophony, 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 discipline. 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,[5] estimating chords from chroma vectors[8]) etc, the Industry is primarily interested in solving problems like song recommendation and similarity searches for mass consumption. 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 following is an overview of such software used in production and

the problems they attempt to address.

2.1.1 jMIR

jMIR,[7] 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 and midi file formats) and mine cultural information from the web. jMIR extracts an 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 estimation etc). Nevertheless its one of the best open source MIR software in existence especially for MIR research involving machine learning approaches.

2.1.2 Marsyas

Marsyas[9] (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. Mod-

ulo7 has very different goals (multiple format support, music similarity, structured querying etc) as compared to marsyas.

2.1.3 SIMILIE

SIMILIE[10] is a set of tools for music similarity measures used for monophonic melodies and features multiple approaches to construct vector space models for melodies. The techniques used for melodic similarity analysis in SIMILIE are novel and derive from many subfields such as Natural Language Processing. Modulo7 uses a subset of these similarity measures as basis for an extended an 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[4] 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 described in 5.3.4

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[11] is a set of tools for computer based automation and assistance in musicology research. Humdrum has the capability for solving very complex questions using music theoretic concepts. It supports its own file format for analysis of music called the kern format.[12] Humdrum is specifically designed for musicologists for automating tasks that they otherwise would have required manual analysis but gathering statistics, music classification or music similarity analysis are not end goals for Humdrum. The fundamental difference of Modulo7 over humdrum is Modulo7 acts as a bulk analysis and querying tool while humdrum is designed for specific and in depth analysis of songs.

2.1.6 Gamera

Gamera[13] is Optical Symbol Recognition(OMR) 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 and is extensible to scriptures

and languages. Gamera also supports creating of new plugins for custom tasks and is widely used in academia for OMR.

2.1.7 Audiveris

Audiveris[14] 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 Optical Music Recognition System. 5.3.2.

2.2 Music Representation Formats

Modulo7 parses multiple formats for music described in 5.3. However there are many other sources prevalent in academia 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[12] is used in humdrum to symbolically denote events in columns while voices are represented in rows.[11] This facilitates a columnar represen-

tation of music on which humdrum can perform different kinds of music theoretic analysis.

2.3 Typical problems of MIR

On top of the generic software created by researchers and industry experts, experts have tackled specific problems in Music Cognition,[15] classification,[7] query by Humming Systems[16] etc. Broadly speaking, 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 of 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 based on a coarse criteria of what "type" a particular song is. Companies like Pandora and Microsoft assign genres to songs via musicologists[17] which means highly trained people manually classify music. Such approaches are expensive in terms of human labor and prone to errors. Automatic Music Classification takes a different approach using algorithms and machine learning approaches like jMIR[7] 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 monophonic melodic similarity[18] and not on generic polyphonic similarity. There are many systems[10] and music databases in existence[12] for the purpose of music similarity analysis.

2.3.3 Automated Musicological Research

In many cases musicological research is conducted manually by applying rules and music theoretic criteria. An example would be applying counterpoint analysis techniques given the rules in a treatise[19] to music sheet manually. This is labor intensive and the research community tries to address this inefficiency via techniques to automate analysis of music. A significant effort is done by the Humdrum community[11] 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 task would be music transcription(also known as melody extraction[20]) to convert audio to symbolic formats which allows for subsequent symbolic analysis. However researchers

have only found success in melody extraction where one voice is clearly dominant in a recording.[20] Researchers have also worked on quantitatively defining the concept of timbre (a peculiar tonal quality of a voice independent of pitch and loudness which characterizes the source of the sound) with varying degrees of success both qualitatively[21] and computationally.[22]

2.3.5 Intelligent Music Archiving and Retrieval

Key to music information retrieval are efficient and novel techniques to archive musical sources so that meaningful queries can be made against these archived sources. 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"[23] . 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[24] 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/or 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[12,25] and query systems are built and comparisons are based on lyrics genre tags and other properties of music data.[1] Most approaches have been based on collaborative filtering[2] based on contextual meta data (information extracted from a community of user's judgments and comments on music) and sparingly from low level audio features extracted from a song.[1]

2.3.7 Audio Fingerprinting and Song ID

A very industry relevant problem statement involves fingerprinting audio files and matching these finger prints to an input(melody or fragment of a song) fingerprint. These systems stress on an exact match as an end goal. Many commercial systems are in existence including companies like Shazam[16] which have developed sophisticated 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 possibilities 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). Traditionally music theory is used for providing directives to a performer to play a particular song/score or for a composer for producing novel 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.

This chapter is primarily meant for readers with a weak or lack of understanding of western music theory and can be skipped if the reader is familiar with these concepts.

3.1 Building Blocks of Music

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

Pitch/Note: A pitch is a deterministic frequency of sound played by a musical voice (instrument or a singer). 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,[26] 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 consecutive notes. There are 8 octaves numbered 0 to 7 which are played by traditional instruments or vocal ranges. Notes are categorized into 7 major notes types (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 defined as the incremental or decremental distance between two consecutive 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 the distance between two consecutive note types. For example there is one tone in between A and B.

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/Note duration: A pitch/note 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 which will persist twice as long as a quarter 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 which is covered in the end of 3.2.

Rests: Rests are pauses in between notes (with no sound being played at that point of time) for a fixed duration, generally in the same unit of measurement as a pitch duration. For example a whole rest is of the same duration as a whole note.

Melody: A melody is a succession of notes and rests which sound pleasing(which is subjective to a listener).

Chord: A chord is a set of notes stacked together (being played on or almost on the same time). Chords are the basic building blocks of the concept of harmony. 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.

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 rests by a single instrument/vocalist. The reader can think of a voice as a hybrid or generalization of the melody and harmony concepts.

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).

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 good.

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 sing/play.

Score/Song: A score or a song is an interplay of voices. It is the final product of music that is delivered to an audience. Songs can be categorized 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).

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 discuss few such concepts

Polyphony/Monophony: A monophonic 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 or a band performing a chorus section of a song.

Phrase: A musical phrase is a contiguous part/snippet of a 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. 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 relationships 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 pitch) to inculcate a sense of completeness. Moreover other pitches in relation to this pitches have various degrees of stability and serve different functions.

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 fre-

quency 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 defines the set of notes that can be played for a particular piece of western music.

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 are 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.

Dynamics: The dynamics of a song is a coarse idea which indicate the relative loudness of notes, speed or pace of notes being played across phrases etc.

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

Mathematical Formulations and Models

This chapter describes the mathematical modules formalizing the concepts described in 3. A significant chunk of the ideas described here are derived from[10] and[27] with some simple novel extensions, in particular to polyphonic music.

4.1 Basic Notation

In this section we define some basic notation that is utilized in subsequent sections in this chapter

1. Pitch : A pitch p is a quantitative representation of the concept of pitch defined in 3.1. Given an octave number x and a note type d (index of note in the

western notation scale), $p = 12 \times x + d$

2. Pitch Onset : Onset t_i is the absolute time at which p_i begins in a song.

3. Interval : An interval is the difference between two consecutive pitches. Mathematically an interval can be defined as $\Delta p_i = p_i - p_{i-1}$

4. Pitch duration : The pitch duration is the time difference between two consecutive pitch onsets. Mathematically a pitch duration can be defined as $\Delta t_i = t_i - t_{i-1}$.

4.2 Preprocessing Steps

It might be that the input sources require certain preprocessing steps for certain mathematical models 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.

4.2.1 Key Transposition

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 define the intervalic distance for songs S_1 and S_2 as

$$Int(S_1, S_2) =\mid K_1 - K_2 \mid \tag{4.1}$$

Here K_1 and K_2 stand for the key signatures for songs S_1 and S_2 respectively and their difference stands for the number of semitones in between the two keys. Consider a pitch p_j in a song a song S_i. Define the intervalic shift operation as $p_j^{shifted} = p_j - Int(S_1, S_2)$. The new transposed Song $S_2^{transposed}$ can now be formulated as

$$S_2^{transposed} = \{p_j^{shifted} \mid p_j \in S_1\} \tag{4.2}$$

4.2.2 Voice to Melodic Representation Conversion

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. Given a chord c_j in song S, and define the procedure which gives the root note of a chord (melodic representation of a chord) as $r(c_j)$ define the conversion as follows

$$S_{melodic} = \{S | c_j \rightarrow r(c_j) \quad \forall \quad c_j \in S\} \tag{4.3}$$

Here $c_j \rightarrow r(c_j)$ denotes converting the chord c_j into its melodic representation.

4.2.3 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 pre-processing steps of converting generic voices to pure melodies.4.2.2 is necessary before any contourization can be applied. 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. If p_i denotes the i^th pitch that occurs in a voice V, the i^{th}Gross Contour can be mathematically defined as

$$GC_i(V) = \begin{cases} 1 & p_j > p_{i-1} \\ -1 & p_j < p_{i-1} \\ 0 & otherwise \end{cases} \tag{4.4}$$

The Gross Contour is defined as $GC(V) =< GC_1(V), GC_2(V)...GC_n(V) >$ where n is the number of pitches in voice V.

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 pitches p_i and p_j and their onsets t_i and t_j as[27]

$$m = \frac{p_j - p_i}{t_j - t_i} \tag{4.5}$$

Define the concept of contour extrema as any two pitches p_i and p_j s.t $i < j$ where every pitch $p_k \forall k \in \{i, j\}$, p_k is either greater than or less than both p_i and p_j.
Once all the contour extremum are ascertained, natural contour can be defined as a semi tone transposition of every note in between two consecutive extremum notes p_i and p_j as

$$p_k = p_i + m(t_k - t_i) \quad \forall \quad k \in \{i, j\} \tag{4.6}$$

4.3 Vector Space Models of Music

In traditional text based information retrieval retrieval systems, documents are indexed and vector space representation of documents are created which facilitate in comparison of documents. Typical approaches for this 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[27] but generalized for polyphonic music.

4.3.1 Vector Space Models for Monophonic Music

Certain vector space models can be naturally defined for monophonic music. These vector space models can be represented as simple arrays. Given the pitches and onset times 4.1 of notes played in a song we can define monophonic vector space models as follows

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

$$P =< n_1, n_2, ...n_n > \tag{4.7}$$

A similar vector representation could be when the time information is eschewed in favor of only the pitch information. This vector is called the raw pitch vector and is defined as

$$R =< p_1, p_2, ..., p_n > \tag{4.8}$$

Pitch Interval Vector: Another way to look at elements is the interval spacing between elements. Given the definition of interval in 4.1 the pitch interval vector is defined as

$$PI =< \Delta p_1, \Delta p_2, ..., \Delta p_n > \tag{4.9}$$

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

$$RPI =< rp_1, rp_2, ...rp_n > \tag{4.10}$$

4.3.2 Vector Space Models for Polyphonic Music

This section discusses the mathematical formulations behind vector space model implementations for polyphonic music. These models can directly be utilized for similarity measures for songs in 4.4.3.

Normalized Tonal Histogram Vector: The tonal histogram is a vector or map of twelve distinct intervals present in western music theory. Each position in the

vector corresponds to the total number of times that interval has occurred in a song. Mathematically define $\Delta P^{voice_j} = \sum_{i=1}^{len(voice)} \Delta p_i^{voice_j}$ 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 $\sum_i \delta(p_i)$ stands for the number of pitches p s.t $\Delta(p) = i$. Now we define the normalized tonal histogram vector as

$$NTH(S) =< \Delta p_1^f, \Delta p_2^f, ..., \Delta p_{12}^f > \tag{4.11}$$

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 : $\Delta t_i^f = \frac{\delta(t_i)}{\Delta T^{song}}$ where $\delta(t_i)$ is defined as $\sum_{\Delta p_j = i} \Delta p_j$ where Δp_j is the interval duration as defined in 4.1. We can now define the normalized tonal duration histogram vector as

$$NTDH(S) =< \Delta t_1^f, \Delta t_2^f, ..., \Delta t_{12}^f > \tag{4.12}$$

Normalized Pitch Duration Histogram Vector: The pitch duration histogram is a vector or map of twelve 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 : $\Delta t_i^p = \frac{\mu(t_i)}{\Delta T^{voice}}$ where $\mu(t_i)$ is defined as $\sum_{\Delta t_j = i} \Delta t_j$ where Δt_j is the pitch duration as defined in 4.1. Thus we can define the normalized tonal duration histogram vector as

$$NPDH(S) =< \Delta t_1^p, \Delta t_2^p, ..., \Delta t_{12}^p > \tag{4.13}$$

4.4 Similarity Measures

Similarity is defined in Modulo7 as a function which takes as input two voices or songs and outputs a real number 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 ranked similarity measures. Mathematically

$$Sim_{song}(S_1, S_2) \in (0, 1) \tag{4.14}$$

$$Sim_{voice}(V_1, V_2) \in (0, 1) \tag{4.15}$$

4.4.1 N-gram Similarity Measures

String representations of voices can be treated as text documents, and as a result n gram representations of voices can be used for developing similarity models. The following n gram models are implemented which are described in.[27]

Count Distance Measure: Let t_n and s_n denote the set of distinct n-grams in the string representations of voices t and s respectively, and let τ denote an n-gram. The count distance is defined as

$$\sigma(s,t) = \frac{\sum_{\tau \in s_n \cap t_n} 1}{max(\mid s_n \mid, \mid t_n \mid)} \tag{4.16}$$

Sum Common Measure: Given the above definition of s_n, t_n and τ, Let $f_s(\tau)$ and $f_t(\tau)$ denote the frequencies of of n-gram τ in s_n and t_n respectively, the Sum Common Measure is defined as

$$\mu(s,t) = \frac{\sum_{\tau \in s_n \cap t_n} f_s(\tau) + f_t(\tau)}{\mid s \mid + \mid t \mid -2(n-1)} \tag{4.17}$$

Here $\mid s \mid$ and $\mid t \mid$ are lengths of string representations of voices s and t.

Ukkonnen Measure. Ukknonen measure is similar to Sum Common Measure, except it takes the absolute difference of trigram frequencies that are not present in either string. Mathematically

$$\sigma(s,t) = 1 - \frac{\sum_{\tau \in s_n \cup t_n} \mid f_s(\tau) - f_t(\tau) \mid}{\mid s \mid + \mid t \mid -2(n-1)} \tag{4.18}$$

N gram models are readily extensible to polyphonic music by the generic similarity technique described in 4.4.3.

4.4.2 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 4.5 we extend standard similarity measures to vectors of unequal length.

Edit Distance on Raw Pitch Vector Representation: Consider the raw pitch vector in equation 4.8. 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, Wagner-Fischer algorithm etc[28]).

4.4.3 Similarity Measures for Polyphonic Music

In order to incorporate vector space models to polyphonic similarity, monophonic measures can be extended in order to accommodate for polyphony.

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 and acts as a generic polyphonic extension to models in 4.4.1. Mathematically

$$GMVS(S_1, S_2, VSim) = arg_{max}(VSim(V_i, V_j))\ s.t\ V_i \in S_1\ and\ V_j \in S_2 \tag{4.19}$$

Standard Document Similarities: Given the document representations in 4.3.2, standard document similarity measures like cosine similarity can be applied. We can define certain measures such as the tonal histogram similarity as

$$THS(S_1, S_2) = cosine_{sim}(NTH(S_1), NTH(S_2)) \tag{4.20}$$

where NTH is defined as equation 4.11. In the same manner we can define similarity measures for 4.12 and 4.13.

4.5 Sub melodic similarities and Tonal Alignment

Often its important to judge which regions of one melody are maximally similar to other regions of a different melody (also called as tonal alignment) instead of judging

overall similarity. Modulo7 takes inspiration from bio informatics domain and uses the smith waterman algorithm modified for voice similarity.[29] The algorithm is as follows:-

```
1: procedure SMITH WATERMAN VOICE SIMILARITY(V1, V2, INSIM)
2:     Define WM = Array[len(V1)][len(V2)]
3:     for i in 1 to len(V1) do
4:         WM[i][0] = 0
5:     end for
6:     for j in 1 to len(V2) do
7:         WM[0][j] = 0
8:     end for
9:     for i in 1 to len(V1) do
10:        for j in 1 to len(V2) do
11:            WM[i][j] = max(0, WM[i - 1][j - 1] + InSim(V1(i), V2(j)), WM[i - 1, j]
   + InSim(V1(i), φ), WM[i, j - 1] + InSim(φ,V2(j))
12:        end for
13:    end for
14:    return WM[len(V1), WM(len(V2))] / max(len(V1), len(V2)
15: end procedure
```

Here InSim(Vx, Vy) is a customizable similarity function for two voice instants (pitch/chord) and ϕ stands for absence of a pitch/chord and as a consequence InSim(Vx, ϕ) denotes the similarity between a pitch/chord to a rest or no pitch/chord.

4.6 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 criteria or predicate. Some examples would be if the piece has a species 1 counterpoint (i.e.

the voices move with the exact same speed[19]) or if the piece has voices in the SATB criteria (with exactly 4 voices and their ranges being in particular range of high and low notes).[30] This allows a consumer to build complex queries based on pieces satisfying selectivity requirements by compounding such criteria. Following are the criteria implemented in Modulo7.

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.

Time Signature Criteria: Its a simple criteria that checks if a song has a particular time signature/meter or not.

SATB Criteria: Whether the song satisfies the STAB voice classification as defined in.[30]

4.7 Statistics Analysis

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

$$Statistic(Song) = x \ s.t \ x \in \mathbb{R} \tag{4.21}$$

The following are the statistics 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 defined in 4.2.2, since its only applicable to pure melodies.

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.[31]

2. Sadness Index : The sadness index of a song is the number of minor intervals[32] 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.[31]

3. Power Index : The power index of a song is the number of perfect intervals in a song divided by the total number of intervals. Perfect melodic intervals are very prevalent in rock and metal songs and are an expression of a neutral/powerful tone. This stems from the fact that perfect fifths along with perfect unison or perfect octaves form power chords, which are very common in rock music[33]

Max Range of a song The maximal range that is occurring within a given song. This is the max range over all voices of a song.

Most frequent interval/ pitch The pitch / interval value of the most commonly occuring pitch/interval.

Average pitch/interval duration The average pitch and duration interval for all given pitches, durations.

Chapter 5

Software architecture and Methodology

This chapter provides the details of the software architecture and the methodology of Modulo7 and also lists the limitations of the Modulo7 software implementation.

5.1 Server Side architecture

Modulo7 is designed with the purpose of scalability. Modulo7 is built up of the following modules

1. **Source Converters** : Converts music sources (e.g. music XML, midi etc) into Modulo7's binary representation.

2. **Music Theory Models** : These models are implementations of the music

theoretic criteria formalized in 4.4, 4.7 and 4.6 as well as the vector space models defined in 4.3.2

3. **Persistent Storage Mechanism** : The Modulo7 internal representation is implemented as a hierarchical class design as shown in 5.3. This representation is then serialized via Apache Avro A.1.2 and stored to disk.

4. **Lyrics Indexer** : An inverted index of song 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. Apache Lucene was used for developing the document index for lyrics A.1.1 and alchemy A.1.7 was used semantic intent and language ID feature implementations.

5. **Lyrics similarity models** : A set of similarity models that can be applied to indexed lyrics objects. Modulo7 also implements meta data predictor models described in 5.5.4.

6. **Query Engine** : An SQL like interface to a client that allows you to gather and ascertain useful information (based on music theoretic criteria). Antlr was used for developing a lexer and parser for this engine A.1.4.

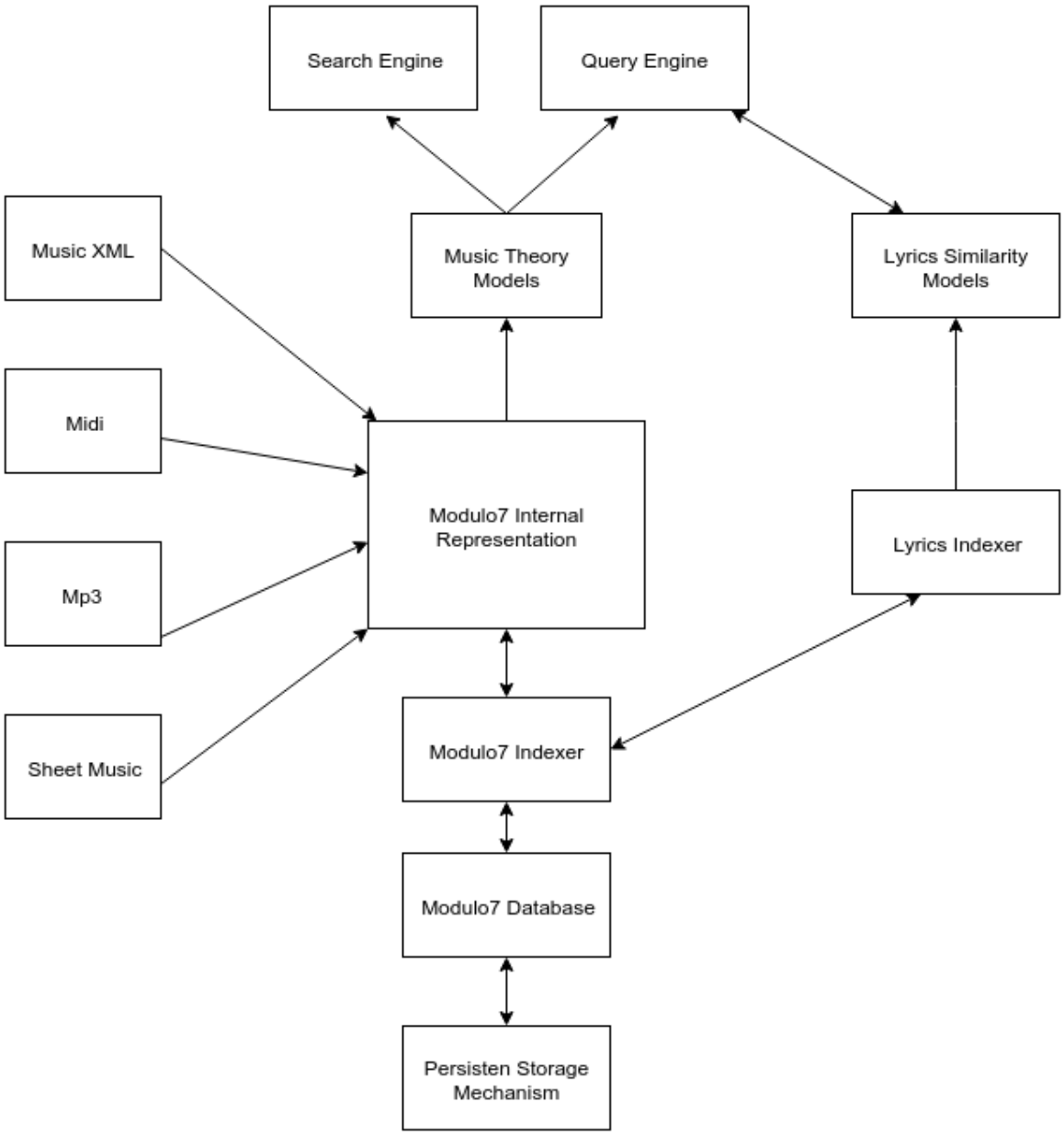

Figure 5.1: Block diagram of Modulo7 software architecture

5.2 Client architecture

The server exposes a sql like interface with a query syntax defined in 5.5.2 and a standard querying set defined in 5.5.1.

Moreover the client also exposes a highly customized search engine based on the vector space representations defined in4.3.2. The search engine implements the ranked order search based on the similarity measures defined in 4.4.

5.3 Song sources and Parsers

At the heart of Modulo7's design is its song sources parsers (or converters) which converts different song sources into its own internal binary format 5.3. Each music source is a different representation and while certain sources ascribe what how music should be played (e.g music-xml, 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 kern format 2.2 etc), but for the purposes of breadth and ubiquity, four sources have been targeted as input for Modulo7(mp3, sheet music as png, jpeg etc, music xml file and midi files). Its important to note that acquiring features from each format is a domain specific challenge and inaccuracies are inherent because of that. Moreover Modulo7 does not attempt to improve on state of the art feature extraction techniques. 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.[34] 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 1990s.

Midi is one of the easier formats to parse for symbolical music information. 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 Digitized 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

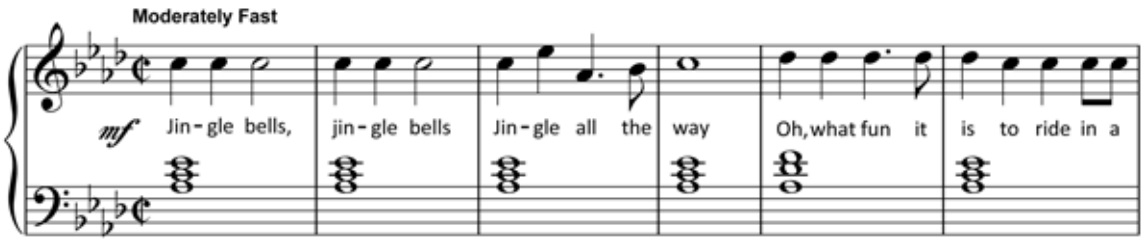

Figure 5.2: Jingle bells melody sheet music representation

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.[19]

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 of computer vision algorithms and even the state of the art software in existence today cant handle all scores (especially for poorly digitized formats[13]). Given the amount of domain knowledge required, Modulo7 uses a third party library called Audiveris 2.1.7 for the purposes of Optical Music Recognition.

5.3.3 Music XML format

Music XML format is an xml based standard open format for exchanging digital symbolic music.[35] 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.[36] 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 de-facto standards of digital music compression and transfer and playback on most digital audio players. In order to parse this format, Modulo7 uses the technique developed in B.2 and also

utilizes the Echo Nest jEN API for directly processing mp3 files into chromagrams A.1.3.

5.4 Modulo7 Internal Representation

Modulo7 consists of converters that convert data into Modulo7's internal representation 5.3. This representation can be thought of a document representation on which similarity measures described in 4.4 can be applied on. 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 similar to the music xml and sheet music formats. 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 certain criteria and on top of these boolean queries can be formulated. The internal components are categorized as the following:-

Song Metadata: The meta data aspects of 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 a particular song.

Voices in a song: An implementation of the voices described in 3.1, voices in Mod-

ulo7 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 the lyrics from them.

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[25]). Modulo7 accomodates such formats as well.

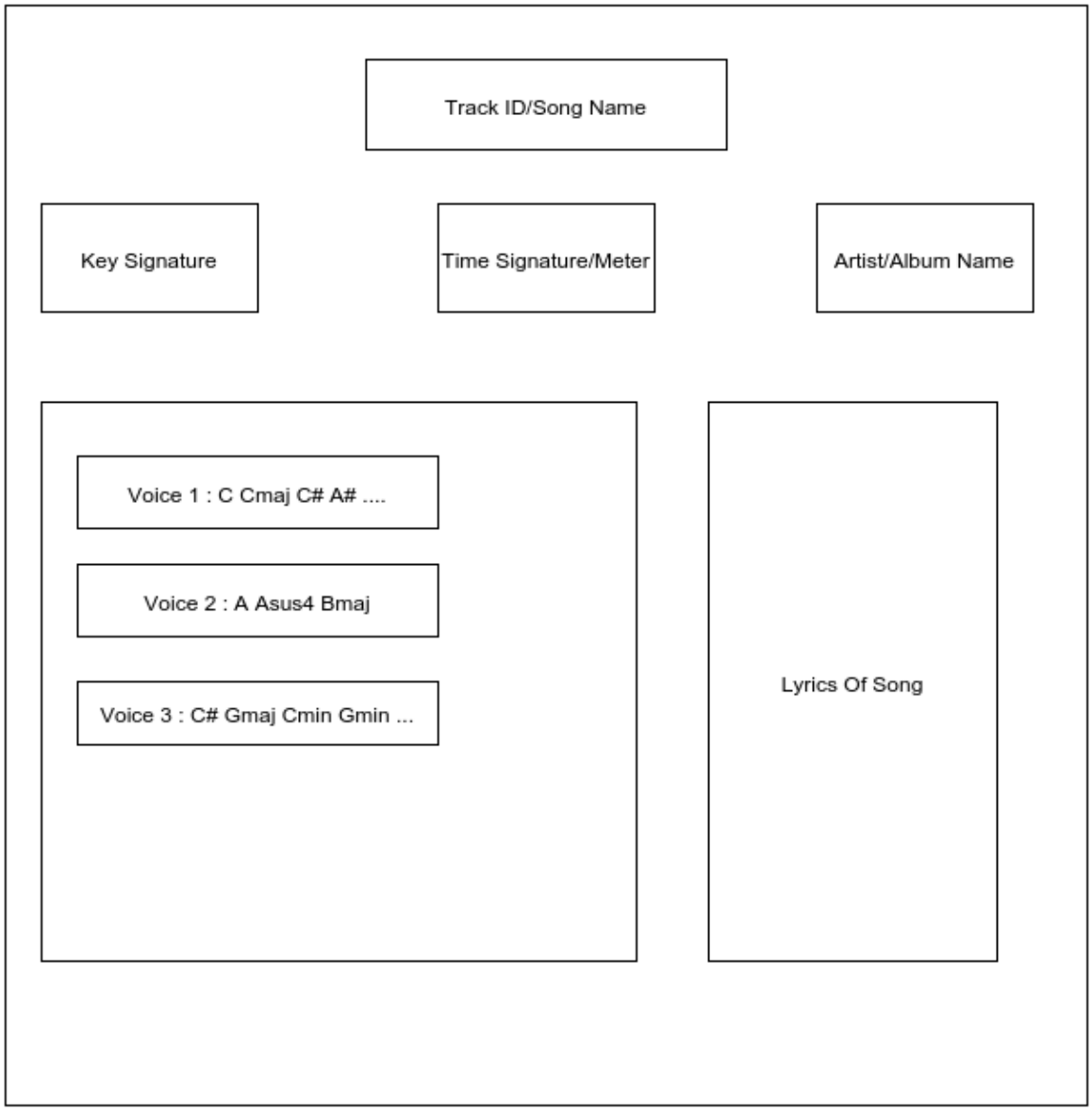

Figure 5.3: Abstract representation of the Modulo7 internal representation

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 a standard query set 5.5.1, or a customized structured query prompt 5.5.2 or a customized similarity based search engine 5.5.3.

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 some 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 similar to an input string (for example a verse in the song).
4. Return all songs that are performed by Led Zepplin
5. Return all polyphonic songs in the database

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 more customized and expressive SQL like query syntax to concatenate boolean expressions of these example queries and more (boolean combinations of all criteria and statistics defined in criteria 4.6 and statistic 4.7 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 as it does not operate on a traditional database (it rather interacts with the Modulo7 indexer). A generic expression can be expressed as follows

select input_src_list from DATABASENAME where expr_list (5.1)

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

2. **DATABASENAME** : The name of the Modulo7 Database. Its acts as 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 4.6 and 4.7. 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 set which either satisfy or do not satisfy a given criteria. The argument criteria is replaced by an implemented criteria in 4.6

   (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 relation a given value defined by the relational_op argument. The arguments to this expression is a statistic implemented in 4.7, 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 value 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 4.4 and the songs themselves are represented as vector space models defined in 4.3.2.

The similarity engine functions by asking the user for a reference "query" song and a similarity measure implemented in 4.4.3. The engine computes the similarity of the query song to each song in the indexed databased and returns a ranked order based on relevance to that particular similarity measure.

5.5.4 Modulo7 Lyrics Analyzer Architecture

The modulo indexer also indexes lyrics, but treats lyrics objects as standard text documents. So the standard model of text Information Retreival techniques can be used to directly 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 Apache Lucene A.1.1 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.1.7.

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). It does this via a sentiment analysis call to alchemy. A.1.7

5.6 Lyrics Based Genre Estimation

On top of these features, the lyrics analyzer provides support for genres prediction. Given a data set with genre annotations to songs along with lyrics, Modulo7 can predict genre annotations for new input lyrics. The following genre estimation schemes are implemented in Modulo7:-

5.6.1 Naive Genre Estimation

Consider $T(S_i)$ be defined as the set of genre annotations for the song S_i which is the i^{th} song in a tag annotated data set. Let S_{new} be a new song for which genre annotations need to be predicted and let L(S) represent the lyrics of a song. Hence L_{new} should be similar to some $L(S_k)$ for their genres to be deemed identical. Let $S_{sim} = \{S_i |\ \ isSim(S_i, S_{new}) \geq \epsilon\}$ be the set of all the songs similar to S_{new}(Here ϵ is some thresh hold value and isSim is a similarity function that compares lyrics of two songs). We define $T_{new} = \{\cup\ T(S_i) \mid S_i \in S_{sim}\}$. In other words the genres of the new song is the union of the genre labels in the songs similar to the new song up to a particular thresh hold.

5.6.2 Weighted Genre Estimation

In the previous scheme, there are no considerations for degree of importance of each tag for a give lyrics or about the degree of similarity between lyrics. In order to accommodate these we assume the existence of tag weights associated for tags in the song meta data. Let $T(S_i)$ be defined as the weighted genre annotations for song S_i. Let $S_{sim} = sort_v^{desc}(S_i, |isSim(S_i, S_{new}) = v)$ be the rank ordered set of genre labels based on descending order of similarity values, where isSim may choose to leverage the weights of tags to compute similarity. Out of these top we choose the top k $S_{sim(k)} = first_k(S_{sim})$ similar songs. The genre estimation can then be defined as $T_{new} = \{\cup\, T(S_i) \mid S_i \in S_{sim(k)}\}$

This scheme takes into account the rank of the songs in based on a similarity metric. The scheme can retain only a subset of the maximal weighted tags in the resulting tag set for the input song and the size of this subset depends on the chosen value of k.

5.6.3 Max Frequency Tag Estimation

In the previous scheme, the frequency of genre labels occurring inside the dataset is ignored. In order to accommodate that let $f_x(S_i)$ be the total frequency of genre label x for the set S_{sim} where $S_{sim} = \{S_i|\ \ isSim(S_i, S_{new}) \geq \epsilon\}$ where isSim and

ϵ is defined identically in 5.6.1. Hence we can define the set of estimated tags as $T_{new} = first_k sort_{f_x(S_i)}(x)$ where k is defined identically in 5.6.2. The tags are sorted according to descending cumulative frequencies of tags in the similar set.

5.7 Meta data Estimation

Unlike other frameworks, Modulo7 attempts to attempts to guess and fill certain meta data that are missing in song sources. One such meta data is key signature of a song which it guesses is the Key Signature of a song via the algorithm described in B.1

5.8 Limitations of Modulo7

While Modulo7 attempts to solve a large set of problems, 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. Also Modulo7 does not take into account atonal music (as a key signature leads to transforms that are needed in certain similarity metrics 4.2).

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.[20]

Chapter 6

Experimental Evaluation

For the purposes of evaluating Modulo7, test cases have been designed into two categories. One category of testing is micro testing, for validating correctness of certain concepts and algorithms for small sets of data 6.3 6.5. 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 larger datasets such as the million song dataset.[25] Due to computing resources, subsets of larger datasets were chosen such that memory and disk requirements could be contained in one PC. No distributed test cases were run as a part of the evaluations.

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

1. Ground truth values presented in datasets (such as tagged meta data or sub-

jective jugdements for song similarity in[25]) are assumed as a base line for evaluations.

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 5.7. This estimation is 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 of the tests are against file formats of 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 5.3.4.

4. In the event of parsing data, there can be legal issues (e.g. the song can be copyrighted). For that reason custom parsers are used for alternate research dataset format (e.g the million song data set has already derived features that Modulo7 intended to derive for Mp3 files and a separate parser is written for this format[25]).

5. All evaluations are done against research datasets[25,37,38] 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.

6.1 Results of Index Compression

The Modulo7 representation 5.3 can be visualize as indexed meta data version of the song with the symbolic information of the song intact(which entails no core information is lost during the conversion). True to all indexed data, Modulo7 represents the song in a much smaller size than the original source when persisted to disk. The following chart demonstrates the average compression of indexed data as compared to source files on the Saarland Music Data (SMD) Dataset[38] when the Modulo7 representation is persisted on disk

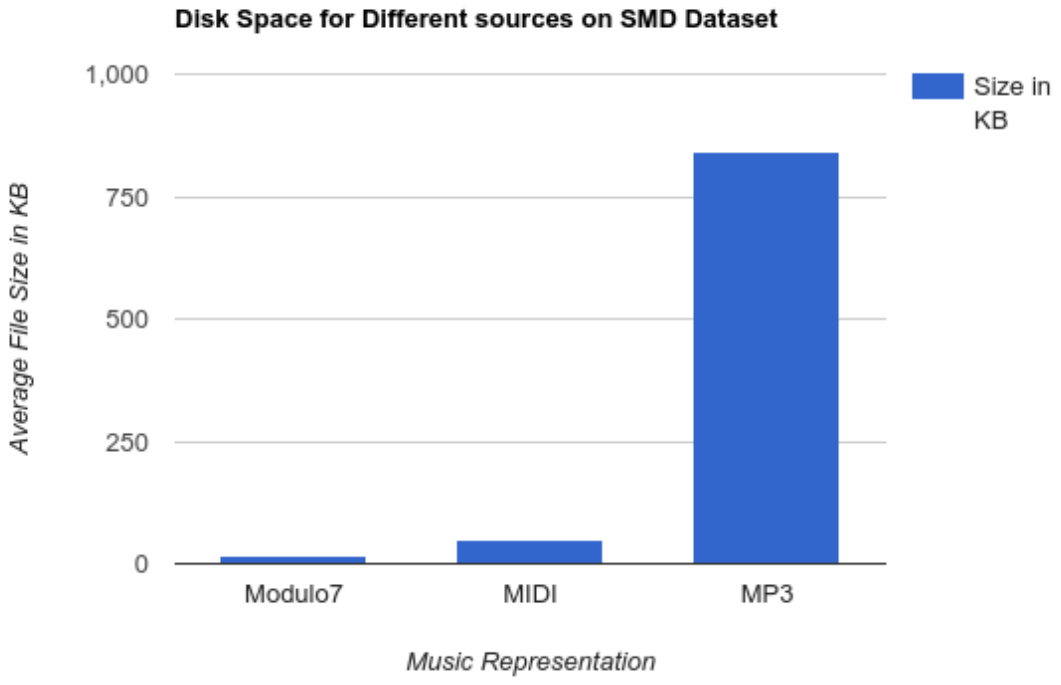

Figure 6.1: Modulo7 SMD Dataset compression

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 are significant as there is a 4 time decrease in size of expressing symbolic information as compared to midi files.

A similar transformation was also done on a direct download able subset of the wikifonia dataset[37] 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 of the Wikifonia dataset file sizes(in KB)

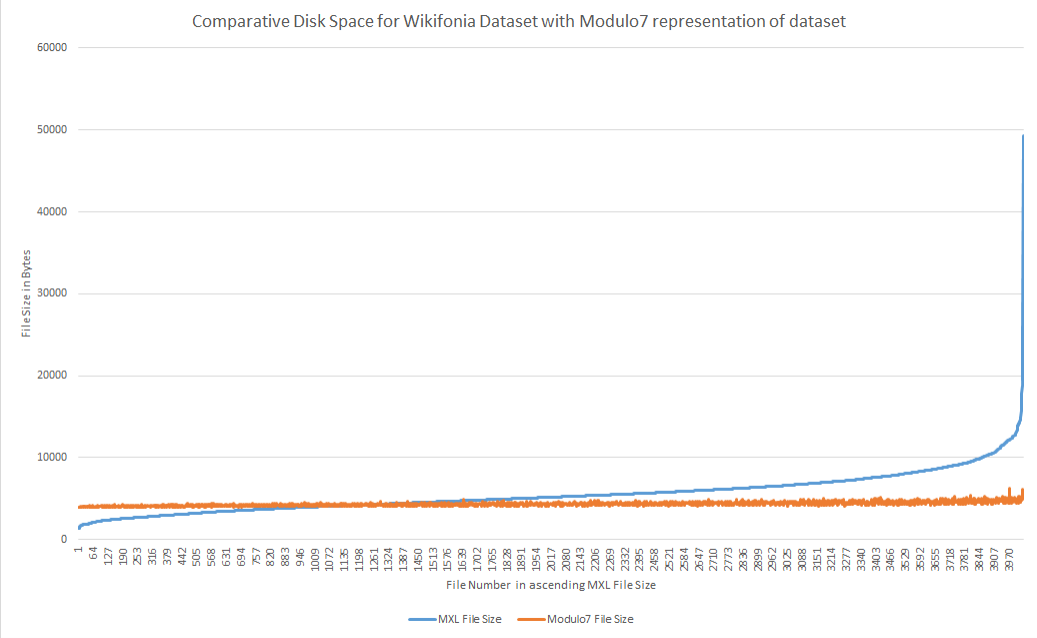

Figure 6.2: Modulo7 comparative file sizes

As expected, Modulo7 is extremely space efficient for storing symbolic information. One observation is that Modulo7 file size increases in a much smaller rate than the wikifonia data set, but the meta data storage requirement is higher for Modulo7 (approximately 4KB).

6.2 Million Song Dataset Experiments

The million song dataset was chosen for experimental evaluation[25] for the querying and similarity engines and the lyrics analyzer. MSD contains pre-computed chromagram transcriptions of Mp3 files and the last fm data set contains a set of tags and genres for building a ground truth for evaluation. Due to the hardware constraints, a scaled down subset was chosen from the original 584,897 songs to a more manageable 10000 songs(offered as a direct down loadable subset in the Million song data set website[39]).

6.2.1 Results on Melodic Similarity Analysis

This set of experiments determine the precision and recall values for the similarities defined in 4.4 on ground truth data extracted from.[25] Modulo7 does not attempt to improve on the state of the art when it comes to similarity metrics or does not attempt to create a new similarity metric. Rather this set of experiments are a test of efficiency in execution and accuracy of existing similarity models of 4.4.3 and 4.4.1

on large scale datasets.

For this experiment the songs that they were monophonic are retained(since polyphonic transcription from audio files is not a fully solved problem[20]) and hence subset of 3,784 songs were retained. These songs were mapped with the last fm similarity dataset and 838 songs out of the monophonic subset were identified to have at least one similar song listed in the last fm tags.

Only the monophonic similarity measures are used for these experiments from 4.4.2. The testing was done with a 10% test set (search queries) and 90% hold out set (data base) and 10 fold cross validation was used.

In order to estimate a song similarity ground truth that faithfully captures the user's sentiment about a song, a quantitative estimate was designed around the meta data associated with a song called the tag hit rate. Given a song S_1 with tags $T(S_1)$ and another song S_2 with tags $T_(S_2)$. The tag hit rate is defined as :-

$$THR(S_1, S_2) = \sum_{t_i \in T(S_1)} \sum_{t_j \in T(S_2)} \begin{cases} 1 & t_i == t_j \\ 0 & otherwise \end{cases} \tag{6.1}$$

This can be interpreted as an quantitative estimate of the agreement between tags of two songs, and as a consequence the song similarity based on a collaborative filtering

approach.

Based on this measure, each song in the test set can be compared against the songs in the hold out set to ascertain ground truth data, with any song have a tag hit rate score greater than 0 is considered to be a relevant song.

In order to compare the efficiency of each of the similarity measures implemented, the average precision and recall values are listed for melodies present in the million song data set. Only those similarity measures are selected which do not depend on melody length (in other words melodies of unequal length can be compared with these similarity measures)

Similarity Measure	Average Recall	Average Precision
SCM Trigram	0.308	0.299
Ukkonnen	0.339	0.291
Count Distance	0.294	0.283
Tonal Histogram	0.341	0.362

Table 6.1: Average Precision and Recall for Melodic Similarity Measures

From the following results and observations on the data set the following can be concluded

1. Similarities based on music theory (tonal histogram) marginally outperform those based on n gram models.

2. In general, the similarity metrics perform better on symbolic ground truth data[40] as compared to mp3 transcribed data,[25] as tested in this experiment. A potential explanation for this would be the inherent complexity associated with a faithful symbolic transcription of audio data,[20] which inadvertently reduce the precision and recall of the similarity measures.

6.2.2 Results on lyrics similarity and genre estimation

On top of the experiments done for song sources incorporating tonal information, there were specific experiments that were carried out for the lyrics analyzer 5.5.4 component. The ground truth for these experiments is the musix match lyrics dataset present in the million song data set.[25] 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 as baseline for set based similarities of lyrics. For this experiment the genre labels were extracted from the tag tratum genre annotations dataset of the Million Song Data set[25] to acquire the genre labels that are observed for a given song and then build a predictive model that outputs genre labels for a newly seen song.

Out of the 210,519 songs with lyrics provided in the million song data set and 280,831

songs with corresponding genre labels annotated, 55726 songs were identified with both lyrics and genre labels present, so this set of songs are considered the ground truth for estimating genre labels for novel lyrics

The lyrics in this dataset are in the bag of words document representation format and hence standard set based similarity measures like cosine similarity can be used for comparing lyrics. The lyrics in the million song dataset are already stemmed via the Porter stemmer[25] so no explicit stemming is conducted as a part of this experiment.

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 ground truth 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 5.6.

Parameters which determines the degree of permissible agreement are the thresh hold value ϵ defined in 5.6.1 and 5.6.3 and top k songs chosen in 5.6.2 and 5.6.3. For the purposes of experimental evaluation, these hyper parameters were varied to produce a precision recall curve for the weighted genre estimation, as its an ordered ranked list and a ROC (Receiver operating characteristic curve) for max frequency and naive

genre estimation (as they are produced an un-ordered ranked list).

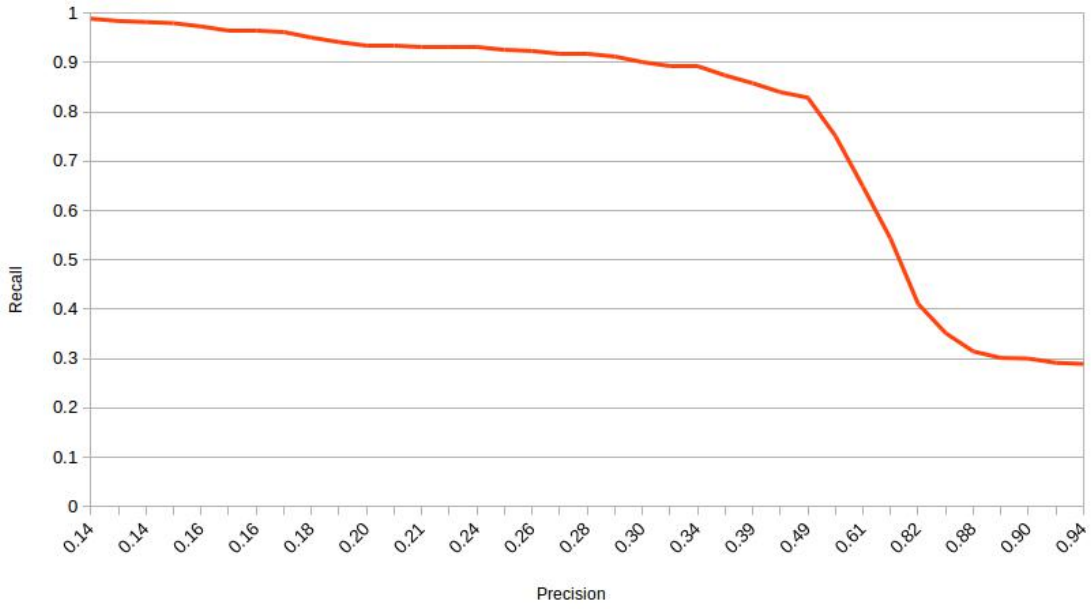

Figure 6.3: Precision Recall Curve for Weighted Genre Estimation

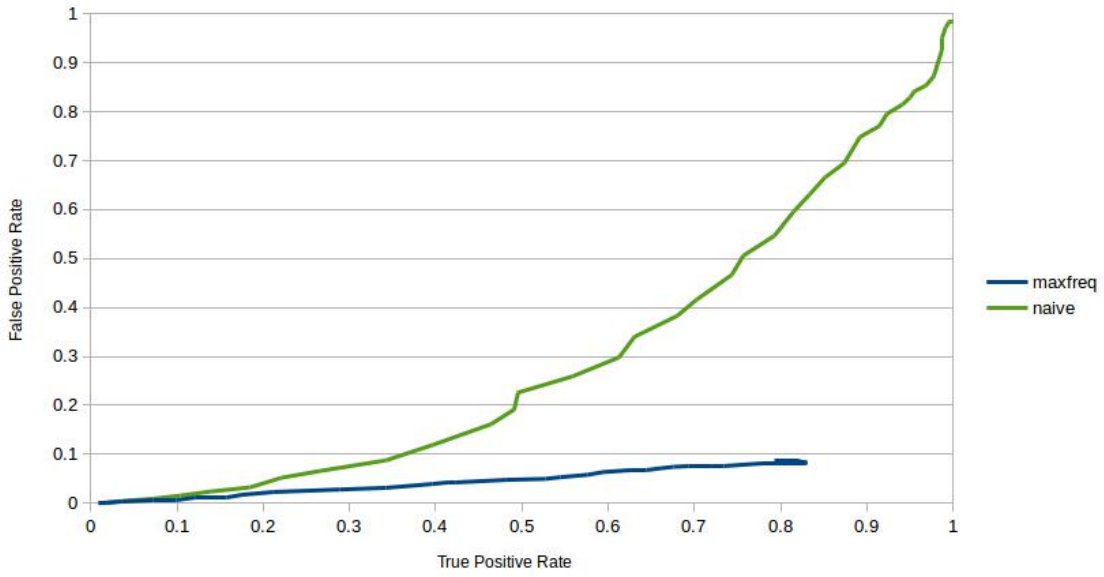

Figure 6.4: ROC curve for max frequency and naive genre estimation

6.2.3 Results on exploratory query analysis

In order to estimate the efficacy of the Modulo7 SQL querying, certain customized exploratory experiments are conducted. In order to ascertain the relevance of the statistic extraction 4.7 and criteria estimation 4.6, certain queries were designed and cross checked with the tags associated with that song (since meta data tags come along with the songs in the million song data set). For example based on a prior statement made about intervals expressing the mood of a song in 4.7 , we can estimate a rock song based on a query : select mp3 from database where power index $>$ k where k is some thresh hold. This particular experiment involves exploring for a reasonable estimate of k to ascertain rock songs from non-rock songs. The ground truth would be the genre labels extracted in 6.2.2 or the last fm dataset tags[25] depending on the query context. **Its important to note that this experiment is exploratory and novel in nature and hence there is no pre-existing framework/methodology or approach to compare against.**

The query and their equivalent statement are listed and the best precision recall (at highest F-measure value) for sample queries are listed below

Purpose	Query	Precision	Recall	Ground truth estimate
Rock Song ID	Q1	0.13	0.98	Song tags : "rock" / "pop_rock"
Sad Song ID	Q2	0.02	0.44	Song tags : "sad" / "sad_song"
Happy Song ID	Q3	0.018	0.4	Song tags : "happy" / "happy_song"

Table 6.2: Results for the exploratory query analysis

We define Q1, Q2 and Q3 as follows

Q1 select mp3 from default_database where powerindex > 0.61;

Q2 select mp3 from default_database where sadnessindex > 0.15 and scale = minor;

Q3 select mp3 from default_database where happinessindex > 0.11 and scale = major;

From these results we can conclude the following :

1. A cursory analysis of the data set revealed that 57% of all songs in the data set are classified as rock or pop rock. Hence the high optimal value of k for powerindex is justified given the higher concentration of rock songs.

2. While recall is high (especially for rock songs), precision is low for all queries in this analysis. This would entail that while the relevant songs are indeed retrieved, many irrelevant songs are also retrieved which satisfy the criteria. This could be resolved by compounding the query with criteria/statistics which filter out the false positives.

6.3 Results on KK Tonality Profiles algorithm for Key Estimation

In order to test the KK Tonality algorithm given in B.1, Modulo7 is benchmarked against a subset of the Wikifonia data set of lead sheets in the compact mxl format (which is just a zipped version of xml files).[37] The original dataset of the Wikifonia is now no longer available but a sizable subset of 6715 songs are currently down loadable 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 algorithm B.1 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 for all 6715 songs (**since the number of sharps or flats in the key signatures are always encoded in music xml files** so only relative major/minor were required to be ascertained). As a consequence only two choices are to be made between key signatures for each file giving a baseline of **50%**. In this particular example, KK Tonality's performance is how well it can distinguish between relative minor and major key(a minor and major key having identical scales but different keys).

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%**. This is commensurate with the reported accuracies reported in.[41]

6.4 Results on CPU and Memory and Disk space compared against jMIR

In order to compare the memory and disk space requirements, Modulo7 was tested against its closest competitor jMIR's[7] jSymbolic component. Both frameworks are written in Java and both involve extraction of features(although that is not an end goal for Modulo7). However jMIR is more exhaustive in what features it extracts so only a subset of those that are also extracted by Modulo7 are considered. Out of the total 111 features that are implemented in jSymbolic,[42] 23 features were identified as implemented as internal computation within the Modulo7 indexers and/or querying engine. A modified version of Modulo7's basic indexing engine was used for extracting the following features.

1. 1 feature for duration of song

2. 2 features for average melodic intervals, note duration

3. 1 feature for Meter classification (simple or compound)

4. 1 feature for lengths of melodic archs in midi files
5. 1 feature for initial tempo of song
6. 4 features for melodic intervals (thirds, fifths, octaves and intervals in the bass line)
7. 2 features for maximum and minimum durations of notes in the song
8. 3 features for most commonly occurring pitch, pitch class and melodic interval
9. 3 features for ranges, namely primary register, range of highest and lowest voices
10. 1 feature for time signature
11. 4 features for checking for voice equality in the following categories : melodic leaps, note duration, number of notes and range

In order to compare the frameworks, jProfiler was used to profile for max CPU utilization, average Java Heap Memory usage and time taken for both frameworks over different sized subsets of the Saarland Music Data (SMD) Dataset.[38] In order to protect against background process interference, the frameworks were ran on AWS EC2 m4x.large instances (dual core 2.4 GHz Intel Xeon E5-2676 v3 Haswell processors and 8 GB DDR3 RAM). We plot the average memory consumed, CPU load and time taken in seconds as a function of dataset size (over monotonically increasing subset sizes of the SMD dataset). We ignore IO performance since in this experiment, IO

is only utilized when pushing output to disk, which is not taken as a metric of evaluation. No data sets involving music xml files were chosen, as jSymbolic does not support music xml files.

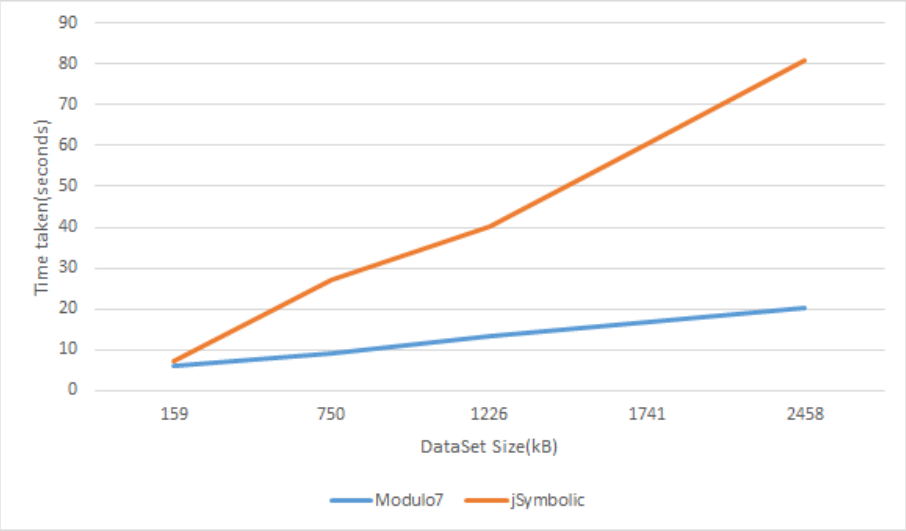

Figure 6.5: Modulo7 vs jSymbolic for time taken to generate features

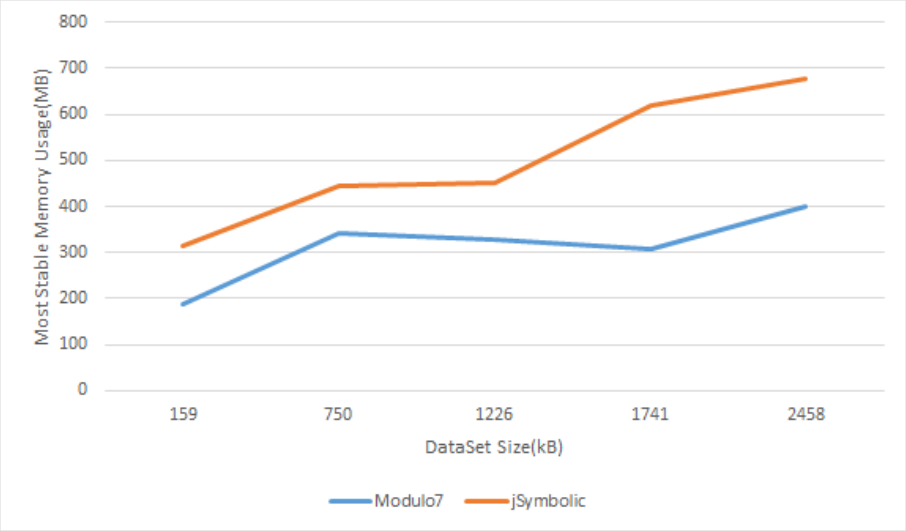

Figure 6.6: Modulo7 vs jSymbolic for average memory utilized

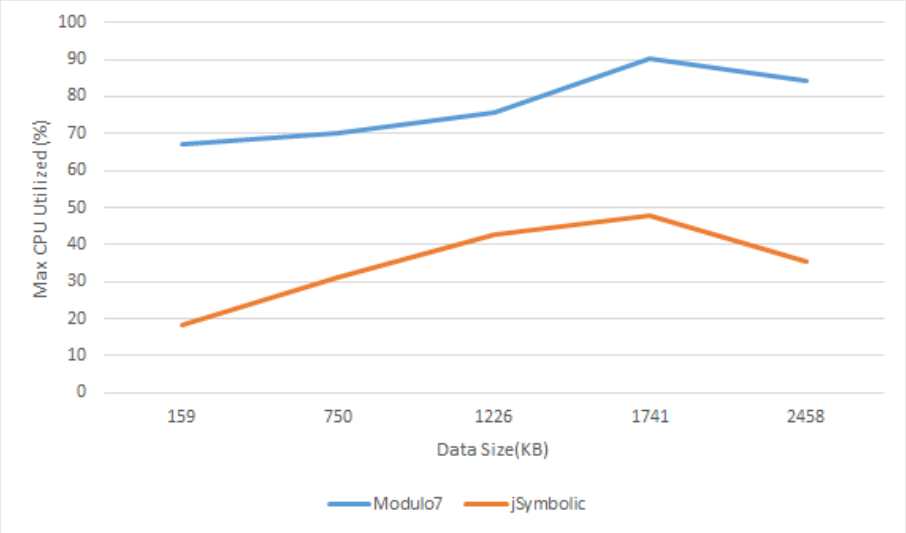

Figure 6.7: Modulo7 vs jSymbolic for maximum CPU utilized utilized

From these graphs we can conclude the following :-

1. Modulo7 is much faster than jSymbolic when computing core features and both of them scale linearly. The rate of increase for Modulo7 is lower, and hence Modulo7 scales better for larger datasets as compared to jSymbolic.

2. jSymbolic under utilizes CPU for computing features whereas Modulo7 is optimal in terms of CPU usage. The profiling results revealed that jSybmolic is single threaded and contains no caching mechanism for storing features (leading to re-computation of features that are dependencies of other features)

3. jSymbolic consumes more Java Heap memory during execution on average. While no conclusive evidence could be established, a code audit revealed jSymbolic uses a large number of primitive arrays[43] in contrast to a fewer number of dynamic allocations in Modulo7 which could lead to higher memory consumed.

6.5 Results on melodic alignment and similarities over sub melodies

A micro experiment was run to show the extensibility of Modulo7 for the purpose of melodic alignment. Its often important to ascertain which regions of a melody are similar to which other regions of a melody. For this experiment, the Smith Waterman algorithm 4.5 is used for similarity computation and representing regions of melodies that are similar to each other. A particular definition for that was used for the similarity between voice instants 4.5 in this experiment is

$$isSim(V_x, V_y) = \begin{cases} 2 & V_x = V_y \\ -1 & otherwise \end{cases} \tag{6.2}$$

Its important to note that this experiment is a demonstrative experiment instead of an evaluation to show the application of 4.5 on tonality alignment. For a more in depth study of tonality alignment and sub melodic similarity, the reader should refer to.[29]

For this experiment a couple of famous publicly available monophonic tunes(twinkle twinkle little stars and jesu joy of man's desiring by J.S Bach) are chosen and their corresponding music xml transcriptions are aligned both to C Major Key.

The following alignments are noticed (some deletions in between are omitted for brevity's sake, relevant substitutions are retained)

TW = D D - E E D

JM = D D - - - - - - - - - - - - - - - - E - - - - - - E D - - - -

This facilitates in visualizing the similar regions of songs. While the songs have widely different appeal (joy of man's desiring(JM) is a hymn and twinkle twinkle little star(TW) is a nursery rhyme), they have a couple of contiguous notes common in the beginning and end.

Chapter 7

Conclusions and Recommendations

In this chapter, we list our findings and conclusions about the work done in this thesis and also explore potential directions for further research.

7.1 Conclusions

In this thesis, a new Music Information Retrieval system is proposed and implemented which applies concepts of music theory for structured querying and search based on custom similarity measures.

The goals that Modulo7 was able to accomplish can be stated as follows

1. To implement an space efficient and an universal indexing scheme for variegated sources of music.

2. To implement and expose a querying language and a search engine which uses music theoretic criteria as its building blocks.

3. To implement a lyrics analyzer to support lyrics similarity. To evaluate the lyrics similarity engine and its ability to predict meta data (in particular the genre of a song).

4. To explore and quantify the efficiency and efficacy of applying music theoretic concepts for similarity judgments and querying.

In a nutshell, Modulo7 unifies disparate music sources into one cohesive framework which allows for a common ground for querying and similarity searches on a heterogeneous data set. While Modulo7 does not extend the state of art in any of the subproblems it tackled (e.g better optical music recognition algorithms, new similarity measures etc), it was able to successfully unite the best aspects of different frameworks while adding algorithms to fix missing meta data like key signature of a song B.1.

However, Modulo7 encountered many obstacles on the path to building a cohesive framework. One notable problem is the difficulty in faithful symbolic transcription for mp3 files 5.8 which hampered performance and accuracy.

7.1.1 Conclusions on the Query Engine Implementation

In this thesis, a novel querying framework was developed which facilitated a structured querying approach based on music theoretic features. The number of features that were implemented as a part of this work was limited and hence optimal queries were not identified (based on the results of the experiments done in 6.2.3). While relevant songs were being included, irrelevant documents were also being fetched as a part of the query. One obvious way to solve this problem to would be to implement more features and specialize queries to exclude irrelevant documents. As such, at present the query engine is not an exhaustive implementation of statistics and criteria.

7.1.2 Conclusions on the Similarity Search Engine Implementation

Modulo7 implements many of the similarity models defined in[27] along with simple extensions to polyphonic music in 4.4.3. This coupled with the variegated source parsing 5.3, allowed for similarity computations of different sources of music (e.g a midi file being compared with a music xml file). Existing frameworks in literature[7,9–11] focus on media specific Music Information Retrieval and hence Modulo7 distinguishes itself from other frameworks.

A key insight is gained about the efficiency and efficacy of implementing and testing the vector space models defined in 4.3.2 by testing it against the million song data set[25] in the experiments 6.2.1. We have proved that symbolic measures enjoy limited effectiveness on inherently non symbolic data (e.g. chromagrams extracted from mp3 files).

7.1.3 Conclusions on Scalability and Speed

As a part of the effort for the querying engine, a novel indexing and persistent storage mechanism is implemented. The persistent store mechanism has been shown to be extremely efficient in 6.1. A natural conclusion could be made in which music data sets could be maintained as persisted Modulo7 internal objects instead of the sources themselves. This would result in significant space savings and could be utilized as a de-facto storage mechanism for symbolic music.

Modulo7 is also proven to be significantly more efficient (in terms CPU, memory and time consumed) for acquiring features then existing state of the art frameworks such as jMIR 6.4. As such Modulo7 would scale more efficiently with bigger data sets and is an ideal framework for bulk processing.

7.2 Recommendations for future research

Modulo7 was an ambitious project and during the course of working on this thesis, several key ideas and concepts were identified as a potential directions for future research work. These extensions can be broadly be classified as creating exhaustive music models framework and scalability enhancements.

7.2.1 Complete Music Models frameworks

On top of the models implemented in 4. Many more mathematical models could be implemented. One problem that was not addressed was of time signature estimation(or alternatively estimating the tempo, meter of the song). Robust methods in literature exist like the one in.[44] Similarly significant extensions can be made on the key estimation procedure based on data tree based representations.[45] These methodologies can directly be implemented in Modulo7's meta data estimator.

Moreover more sophisticated vector space models can be implemented. An example would be the techniques described in,[46] which uses n-grams of acoustic/melodic and harmonic "events" as a vector space representation. But most importantly, a more in depth study of music is required to ascertain how musicologist's compare music and could that be mathematically formulated.

7.2.2 Scalability Enhancements

One of the original motivations for developing Modulo7 was to create a distributed framework for feature extraction, computing similarities and querying. While an in memory caching mechanism was implemented via the Apache JCS module A.1.8 it was never evaluated as the focus had shifted more towards the querying and similarity search engine functionality. However to the best of the author's knowledge, there are no server stack distributed Music Information Retrieval Engines in published academic literature.

In order to improve the performance and extend the scalability of Modulo7 the following recommendations could be made

1. Implement a fault tolerant distributed storage mechanism (e.g Hadoop Distributed File System) for indexed data.

2. Implement a big data framework for extracting features, building meta data based indices (based on Hadoop and/or Spark) typically as Map-Reduce jobs.

3. Expose clients with remote method invocation/Rest End point support for client and indexed data to reside on different computers.

Appendix A

Software Engineering Aspects

Modulo7 is a significant Software Engineering effort. This is partly due to the exhaustive coverage of different music sources and partly due to Modulo7 addressing speed, efficiency and scalability issues that are not addressed by other frameworks.[7,10,11]

At the time of submitting this thesis the Modulo7 source code is hosted at : `https://github.com/Khalian/Modulo7` along with a detailed wiki page describing the steps to use it : `https://github.com/Khalian/Modulo7/wiki`.

A.1 Third Party Libraries Used

Modulo7 utilizes a number of third party libraries in its operations. These libraries and their roles are mentioned below

A.1.1 Apache Lucene

Apache Lucene is a full text information retrieval 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 embedded lyrics in the Modulo7 supported sources.

A.1.2 Apache Avro

Apache Avro is a serialization library used to store Modulo7 representation 5.3 to disk. This allows for faster retrieval of parsed objects instead of having to re-parse entire song sources repeatedly.

A.1.3 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.[47] 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 notes and chords be ascertained based on the ideas developed in B.2).

A.1.4 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.1.5 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.1.6 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.1.7 Alchemy

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

A.1.8 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.

A.1.9 Apache Commons IO and Math

Apache Commons IO and Math libraries are helper libraries used throughout the Modulo7 code base for low level operations.

A.1.10 JFugue

JFugue is an open source playback library for various music sources, and is directly consumed by Modulo7 for providing playback support for different song formats.

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 data or meta data. Some notable algorithms that have been used are briefly described in the following subsections.

B.1 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 it as midi messages.[34] 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 (4.5). There are many meth-

ods for achieving this including non trivial tree representations of polyphonic music to estimate key.[45] However in Modulo7, the author has implemented a simpler model for tonality estimation based on templates called KK tonality profiles[41]

The premise of the KK tonality profile stems from experiments done in[41] and[48] which estimate how likely a user is to ascribe a note to a series of 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(for each note type, what is 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.[48]

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

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

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 algo-

rithm is listed below as follows[41]

```
1: procedure Predict Key Signature(song)
2:     Define CMaj and CMin as per eqn B.1
3:     Define MajProf and MinProf = []
4:     MajProf.add(CMaj) and MinProf.add(CMin)
5:     Define prev_Key = C
6:     for key in western keys [D to B] do
7:         MajProf[key] = left_shift(MajProf[prev_Key])
8:         MinProf[key] = left_shift(MinProf[prev_Key])
9:         prev_Key = key
10:    end for
11:    song_Pitch_Hist = compute_song_tonal_histogram(song) as per 4.13
12:    best_Key = CMin, best_Corr = −∞
13:    for key, maj_prof in MajProf do:
14:        if correlation(maj_prof, song_Pitch_Hist) > best_Corr then
15:            best_Key = key
16:            best_Corr = correlation(maj_prof, song_Pitch_Hist)
17:        end if
18:    end for
19:    for key, mij_prof in MijProf do:
20:        if correlation(min_prof, song_Pitch_Hist) > best_Corr then
21:            best_Key = key
22:            best_Corr = correlation(min_prof, song_Pitch_Hist)
23:        end if
24:    end forreturn best_Key
25: end procedure
```

B.2 Symbolic Transcription from Chromagrams

A chromagram[47] is a representation of a song in frequency domain with relative intensities of notes in 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[8] in order to detect chords in chromagrams. This procedure is based on chromagram bitmap representations of different chords and "similarity" of current chromagram with the various bit map representations.

A bit map for a chord is defined as a 12 dimensional vector in which there is a 1 entry for a present note[on its position on the chromagram] and a 0 entry for an absent note[on its position].[8] So for example the C major chord has three notes in it : C, E, G and as a consequence the bit mask for this chord would be : [1,0,0,0,1,0,0,1,0,0,0,0] as the positions for C, E and G are 1, 5 and 8 respectively in the chromagram representation.

Given a set of candidate chords T which contain bit mask representations of all chord and a chromagram, we define chromagram distance δ_i as:[8]-

$$\delta(T_i) = \frac{\sqrt{\sum_{n=0}^{P-1} T_i(n)C(n)^2}}{P - N_i} \tag{B.2}$$

Here C is the chromagram vector, n stands for the entry number/index in the vector, P = 12 (the number of semi tones in an octave), and T_i is the i^{th} element in the candidate chord set. The chord membership can then be defined as

$$MCC(C) = \{arg_{min_{T_i}} \delta(T_i) \quad \forall \quad T_i \in T\} \tag{B.3}$$

Modulo7 uses a heuristic extension for ascertaining a chord/note from a chromagram. Define max chromagram entry as (value of highest index in a chromagram)

$$MCE(C) = \{arg_{max_{C_n}}(C(n)) \quad \forall c \quad \in (1, P)\} \tag{B.4}$$

Let max chromagram index be defined as

$$MCI(C) = \{arg_{max_{C_n}}(n)\} \tag{B.5}$$

The voice instant (note/chord) assignment for a particular chromagram would be

$$VI(C) = \begin{cases} MCI(C) & MCE(n) \geq 0.5 \\ MCC(C) & otherwise \end{cases} \tag{B.6}$$

This equation is used to ascertain a symbolic transcription for a given set of chromagrams.

Bibliography

[1] 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

[2] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1109/MIC.2003.1167344

[3] W. Glaser, T. Westergren, J. Stearns, and J. Kraft, "Consumer item matching method and system," Feb. 21 2006, uS Patent 7,003,515. [Online]. Available: http://www.google.com/patents/US7003515

[4] B. Whitman. (2010, April) The Echo Nest Musical Fingerprint (ENMFP). [Online]. Available: http://blog.echonest.com/post/545323349/the-echo-nest-musical-fingerprint-enmfp

[5] J. Serra, E. Gómez, and P. Herrera, "Audio cover song identification and simi-

larity: background, approaches, evaluation, and beyond," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 307–332.

[6] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. Marcal, C. Guedes, and J. S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, 2012.

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

[8] A. M. Stark and M. D. Plumbley, "Real-time chord recognition for live performance," 2009.

[9] G. Tzanetakis and P. Cook, "Marsyas: A framework for audio analysis," *Org. Sound*, vol. 4, no. 3, pp. 169–175, Dec. 1999. [Online]. Available: http://dx.doi.org/10.1017/S1355771800003071

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

[11] Anon, "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.

[12] H. Schaffrath and D. Huron, "The essen folksong collection in the humdrum kern format," *Menlo Park, CA: Center for Computer Assisted Research in the Humanities*, 1995.

BIBLIOGRAPHY

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

[14] H. Bitteur. Audiveris handbook. [Online]. Available: https://audiveris.kenai.com/docs/manual/handbook.html

[15] W. J. Dowling and D. L. Harwood, "Music cognition," *Psychomusicology*, vol. 7, no. 1, p. 91, 1987.

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

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

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

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

BIBLIOGRAPHY

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

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

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

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

[25] 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.

[26] D. Keislar, "History and principles of microtonal keyboards," *Computer Music Journal*, vol. 11, no. 1, pp. 18–28, 1987. [Online]. Available: http://www.jstor.org/stable/3680175

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

[28] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, Mar. 2001. [Online]. Available: http://doi.acm.org/10.1145/375360.375365

BIBLIOGRAPHY

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

[30] Anon. Satb wikipedia link. [Online]. Available: https://en.wikipedia.org/wiki/SATB

[31] D. L. "Bowling, K. Gill, J. D. Choi, J. Prinz, and D. Purves, ""major and minor music compared to excited and subdued speech"," *"The Journal of the Acoustical Society of America"*, vol. "127", no. "1", "2010".

[32] M. E. Curtis and J. J. Bharucha, "The minor third communicates sadness in speech, mirroring its use in music," *Emotion*, vol. 10, pp. 335–348, 2010.

[33] W. Everett, *The Foundations of Rock : From "Blue Suede Shoes" to "Suite: Judy Blue Eyes": From "Blue Suede Shoes" to "Suite: Judy Blue Eyes".* USA: Oxford University Press, 2008.

[34] M. M. Association. Midi 1.0 detailed specification. [Online]. Available: http://oktopus.hu/uploaded/Tudastar/MIDI%201.0%20Detailed%20Specification.pdf

[35] I. MAKEMUSIC. Developer specifications for the music xml standard of music exchange. [Online]. Available: http://www.musicxml.com/for-developers/

[36] R. Finlayson", "A More Loss-Tolerant RTP Payload Format for MP3 Audio," "RFC 5219 (Proposed Standard)", "Internet Engineering Task Force", feb 2008. [Online]. Available: "http://www.ietf.org/rfc/rfc5219.txt

BIBLIOGRAPHY

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

[38] 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.

[39] D. P. Ellis. Million song downloadable subset. [Online]. Available: http://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset#subset

[40] (2007) Mirex symbolic melodic similarity results. [Online]. Available: http://www.music-ir.org/mirex/wiki/2007:Symbolic_Melodic_Similarity_Results

[41] S. T. Madsen, G. Widmer, and J. Kepler, "Key-finding with interval profiles."

[42] C. McKay, *Automatic Genre Classification of MIDI Recordings*. Montreal: McGill University, 204.

[43] C. Mckay. jsymbolic feature processor source code. [Online]. Available: https://github.com/DDMAL/jMIR/blob/master/jSymbolic/src/jsymbolic/processing/MIDIIntermediateRepresentations.java

[44] J. LaRoche, "Estimating tempo, swing and beat locations in audio recordings," in *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*. IEEE, 2001, pp. 135–138.

[45] 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

[46] N. C. Maddage, H. Li, and M. S. Kankanhalli, "Music structure based vector space retrieval," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. New York, NY, USA: ACM, 2006, pp. 67–74. [Online]. Available: http://doi.acm.org/10.1145/1148170.1148185

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

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

Vita

Arunav Sanyal obtained his Bachelor of Engineering (Honors) Computer Science Degree from BITS Pilani University in 2013 and is currently enrolled in the Master of Science and Engineering Program in the Department of Computer Science at the Whiting school of Engineering in Johns Hopkins University. His primary research interest is in the field of Music Information Retrieval and he has been supervised by Dr David Yarowsky from the Center for Speech and Language Processing and the Department of Computer Science in Johns Hopkins University.

His permanent contact information is : arunav.sanyal91@gmail.com