MODULO7 : A FULL STACK MUSIC INFORMATION RETRIEVAL AND STRUCTURED QUERYING ENGINE

First Author
Affiliation1
author@ismir.edu

Second Author
Retain these fake authors in submission to preserve the formatting

Third Author
Affiliation3
author3@ismir.edu

ABSTRACT

This paper describes a novel Music Information Retrieval and Structured Querying framework named Modulo7. Modulo7 is a full stack implementation (both client and server side software) which facilitates indexing variegated sources of music (midi, mp3, music xml and digitized sheet music files). Modulo7 implements a similarity search engine based on customized vector space representations of songs, an efficient indexing and persistent storage mechanism and an interface for querying attributes based on SQL(Structured Querying Language) like principles. The papers describes the implementation details and outlines speed up and scale up results over input sources and other MIR frameworks and discusses potential improvements.

Keywords : MIDI, Music XML, MP3, Music Retrieval, SQL

1. INTRODUCTION

Given the explosive growth of Music Information Retrieval research, several approaches and software suits have been designed to tackle generic problems such as efficient indexing, similarity searches, archival methods, structured and un-structured querying,feature extraction, audio and digital signal processing. A vast majority of the MIR frameworks in academia tend to approach very specific problems and does not support scalability as a significant end goal in itself [5]. Moreover, industry applications predominantly treat MIR applications based on collaborative filtering approaches [9] and/or manual annotation [6] instead of structural analysis of music sources yet scales very well to large datasets.

Modulo7 is an attempt to capture the best features of both worlds. Modulo7 converts different music sources (midi, musicxml files, sheet music png/jpeg files and mp3) into a single unified symbolic representation. It indexes songs on global properties (artist, key signature, time signature etc) and provides a vector space implementation for similarity searches and querying based on the similie [8] platform and extended to include chords and polyphony. As a consequence, Modulo7 acts as an end to end software suite for Music Information Retrieval including client side querying, searching, indexing and persistent store mechanism. The broad novel goals of Modulo7 can be classified as follows

1. To create a framework for retrieval from multiple digitized music formats and in effect facilitate comparing songs of different formats.
2. To efficiently store and index variegated data formats into a unified format while keeping symbolic information intact.
3. To query/search relevant songs based on a quantified version of music theoretic principles.

2. RELEVANT WORK

Music Information Retrieval is a vast and interdisciplinary body of work. In order to facilitate research in MIR, several software suits and frameworks have been developed in the academic community in the past. Notable amongst them are software suits like jMIR [11] for automatic feature extraction from audio and midi sources to be used as input into machine learning algorithms for genre classification, marsyas [15] and essentia [1] for audio processing, humdrum [2] for automated musicological research, gamera [7] for optical music recognition and symbol identification and SIMILIE [8] for melodic similarity analysis.

On top of the effort done by there has been significant effort undertaken by companies in the field of MIR with an emphasis on music recommendation. For instance Amazon implements a collaborative filtering approach by quantifying a consumer's shopping behavior [9]. Pandora on the other hand utilizes an approach in which trained musicologists manually classify similarity between artists and songs [**?**]

3. SOFTWARE ASPECTS

This section details the software aspects of the Modulo7 most notably its architecture and a typical work-flow description.

3.1 Software Architecture and Design

The Modulo7 architecture can be visualized as 1 and is broadly divided into the following components.

Figure 1. A block diagram of the Modulo7 Architecture.

1. **Music Parsers:** Components that individually parse different music sources into a unified symbolic representation. as described in 2. This object can be stored in memory or in a persistent serialized state via Apache Avro. Depending on the source of the file, the parser utilizes a different approach.

   (a) For mp3 files, the input is parsed into chromagrams using echonest API and then converted into symbolic format using the KKTonality profiles algorithm [10].

   (b) For Midi and MusicXML files, an in house parser is built.

   (c) For Sheet Music, the Audiveris library is used to convert digitized sheet music to

2. **Music theory Models:** Modulo7 implements vector space models described in [8] with extensions to polyphonic music and chords using a template matching algorithm described in [14]. This forms a mathematical formulation of music sources described in detail in section 4 which is leveraged for querying and similarity searches.

3. **Client side Querying:** Modulo7 implements client side abstractions that exposes two modes of information retrieval.

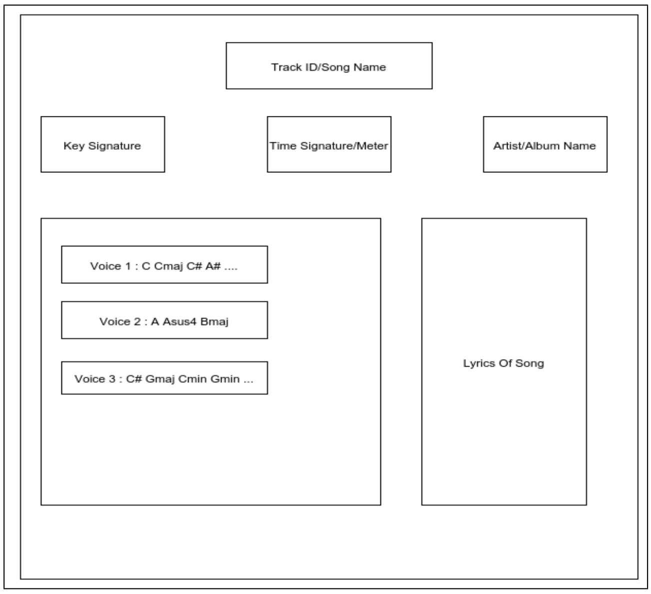

Figure 2. Modulo7 internal representation.

   (a) **Structured Querying:** The structured querying mode allows a user to make SQL like queries based on certain criteria or statistic that is quantified on music theoretic criteria.

   (b) **Custom Search Engine:** Modulo7 exposes a search engine functionality but allowing for customized similarity metric choices as described in section 5.1

4. **Lyrics Engine:** On top of providing support for music sources, Modulo7 also implements a standard text based retrieval model on lyrics of songs written in Apache Lucene.

3.2 Workflow

The workflow can be defined as the steps of operations the user takes to initialize the Modulo7 system and expose the consumer to client facing querying options

1. Modulo7 is initialized by specifying a source directory. Every music source file inside the directory is parsed and converted to either an in memory or persistent store version. It also indexes these songs on global properties (artist, key signature, time signature etc). It also guesstimates certain properties if not explicitly present (e.g key signature via the [10] algorithm)

2. It then exposes a prompt which asks the user for two modes (search engine/structured querying). (In either case the appropriate vector space model is built)

   (a) If the user selects the search engine functionality, Modulo7 similarity engine then asks for an input query song and a similarity metric (from one of the choices described in section 5.1). Then the engine proceeds to rank the query song with each of the indexed songs based on the similarity metric and then returns an ordered list of ”relevant” song.

(b) If the user selects the structured querying mode, Modulo7 asks for an input query along with a set of predicates songs must satisfy (details in section 5.2. Any song that satisfies this query is deemed relevant and returned to the user.

3. Modulo7 optionally caches results in memory so that it does not have to recompute queries.

4. VECTOR SPACE MODELS FOR MUSIC SOURCES

The vector space models form the core of the search and query functionality of Modulo7. Vector space models are derived from quantification of music theoretical representations of songs and extends functionality of the SIMILIE [8] melodic similarity library.

4.1 Building Blocks of unified internal representation

All vector space models are built from the unified internal representation of songs in Modulo7.

1. **Note:** A note is a pair of pitch(deterministic frequency played by an instrument) and onset (time instance at which a pitch is played). A **Chord** is a set of pitches played at the same onset.
2. **Interval:** The difference between two note pitches.
3. **Voice:** A set of notes or chords played in succession(usually on a single instrument).
4. **Song:** A song is a set of interleaved voices, along with some global properties (key signature, time signature, artist, album etc)

4.2 Global Vector Space Models

Now once the above building blocks are ascertained for songs, some vector space models can be defined over the entire song. Some notable ones are

1. **Normalized Tonal Histogram Vector** : For each of the 12 intervals in western music, compute the frequencies of each and divide by total number of intervals over the song. This forms a 12 dimensional vector irrespective of length of song. Mathematically

$$NTH(S) =< \Delta p^f_1, \Delta p^f_2, ..., \Delta p^f_{12} > \quad (1)$$

Here Δp^f_i is the fraction of the number of occurrences of the i^{th} interval.

2. **Normalized Tonal Duration Histogram Vector** : For each of the 12 intervals in western music, compute the cumulative durations each of these intervals are played and divide by total duration of the song. This forms a 12 dimensional vector irrespective of length of song. Mathematically

$$NTDH(S) =< \Delta t^f_1, \Delta t^f_2, ..., \Delta t^f_{12} > \quad (2)$$

Here Δt^f_1 is the fraction of the duration that the i^{th} interval is played.

4.3 Voice specific vector space models

Considering each voice as a separate entity, a voice can be modeled as a vector independent of other voices in a song. A particular song can then be represented as a set of these vectors. Some notable examples of these are

1. **Pitch Interval Vector:** If a voice has k pitches played one after the other, there are k - 1 intervals (for each pitch its immediate subsequent pitch). The pitch interval vector is the chronological succession of these intervals. Mathematically

$$PI =< \Delta p_1, \Delta p_2, ..., \Delta p_n > \quad (3)$$

Here Δp_i is the i^{th} interval which is the difference between the i^{th} and $(i-1)^{th}$ pitches.

2. **Rhythmically weighted Pitch Interval Vector:** Similar to the pitch interval vector, except it also takes into account the time t_i an interval Δp_i is played for. Mathematically

$$RPI =< rp_1, rp_2, ..., rp_n > \quad (4)$$

Here rp_i = $\Delta p_i \times t_i$

Its important to note that these vectors have a dimensionality influenced by the length of the voices in a song.

5. CLIENT ABSTRACTIONS

Modulo7 exposes two client facing

5.1 Search Engine Functionality

Based on the vector space models defined in sections 4.2 and 4.3, customized similarity metrics can be defined to form ranked search functionality for a given query song against an indexed data set. These similarity metrics are derived from the SIMILIE project [8] and extended ideas in [13]. Given two songs S_1 and S_2, we can define similarity to be

$$Sim(S_1, S_2) \in \{0, 1\} \quad (5)$$

The definition of $Sim(S_1, S_2)$ can be implemented in various ways and Modulo7 allows the consumer to choose the similarity metric as a custom argument. So for a given query song S_{query} and a given similarity metric Sim, we can define the order list $Or(S_{query}, Sim)$ given the indexed set of songs S_{ind}

$$Or(S_{query}, Sim) = desc\_sort_k \{S|Sim(S_{query}, S) = k, S \in S_{ind}\} \tag{6}$$

This order list is presented as an output to the consumer. Following are some notable similarity measures implemented in Modulo7.

1. **Smith Waterman Song Similarity**: Given two voices V_1 and V_2, Modulo7 uses the smith waterman genome alignment algorithm [13] to estimate similarity based on notes as units. In order to extend this to songs, pair wise voice similarities are calculated and the max similarity is returned. Mathematically

$$SM(S_1, S_2) = arg_{max} k\{SMV(V_i, V_j) = k|V_i \in S_1, V_j \in S_2)\} \tag{7}$$

2. **N Grams Similarity** :

5.2 Structured Querying Functionality

Modulo7 implements a custom SQL like language with its own grammar definition and implementation. Any query can be constructed as

$$select\ input\_types\ from\ DATABASE\_NAME\ where(boolean\_formula) \tag{8}$$

Here the individual components stand for

1. **Input Types:** A list of the input types we are interested in. For example if we are only interested in midi files then this value is "midi". If we are interested in more than one type(e.g midi and musicxml) then Input Types = "midi, musicxml".
2. **DATABASE_NAME** : A placeholder for the of the data base (created when Modulo7 server starts up)
3. **boolean_function** : A boolean function is a well formed propositional formula (chain of propositions or predicates connected by AND/ORs). Each predicate can either be defined as a criteria or statistic condition.

5.3 Criteria

A criteria is a particular music theoretic condition that a song either satisfies or not. Some notable supported criteria are

1. Polyphony criteria : Whether a song is polyphonic or not
2. Key Signature criteria : Whether a song has a particular key signature
3. STAB Voice Criteria : Whether a song follows the Soprano, Alto, Tenor, Bass Voice structures as defined in [3]

5.4 Statistic

A statistic is a real number generated when applied to a particular song. A statistic condition is defined when this generated statistic satisfies a relational operator with a given thresh-hold value. For example PowerIndex(song) ¡ 0.5 is a valid statistic condition. Some notable supported statistic conditions are

1. Average Pitch Duration over the songs
2. Happiness Index : Fraction of major intervals over the total number of intervals
3. Power Index : Fraction of perfect intervals over the total number of intervals
4. Sadness Index : Fraction of minor intervals over the total number of intervals
5. Number of voices in the song

6. EXPERIMENTAL EVALUATION

This section includes improvements that have been attained over existing frameworks and input sources. For the purposes of this experiment, existing datasets such as the Saarland dataset [12], the Wikifonia Dataset [16] and the million song dataset [4].

6.1 Indexing and Persistent Store Improvements

Modulo7 achieves significant storage improvements over the input formats, while not losing any symbolic information. Figure 3 illustrates this for the Saarland dataset [12]

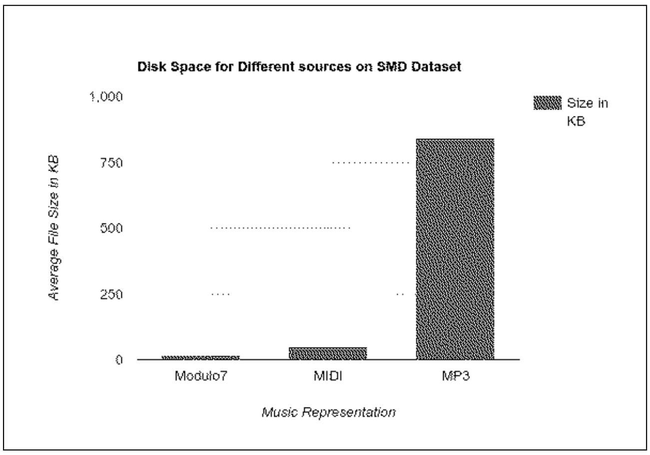

Figure 3. Space storage improvements over Saarland Dataset.

Similarly an experiment was carried out to ascertain compression as a function of data set size for the Wikifonia Dataset [16]

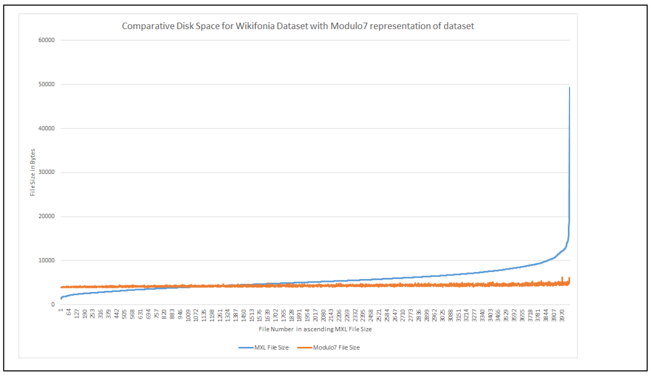

Figure 4. Modulo7 compression of Wikifonia Dataset over increasing subset sizes

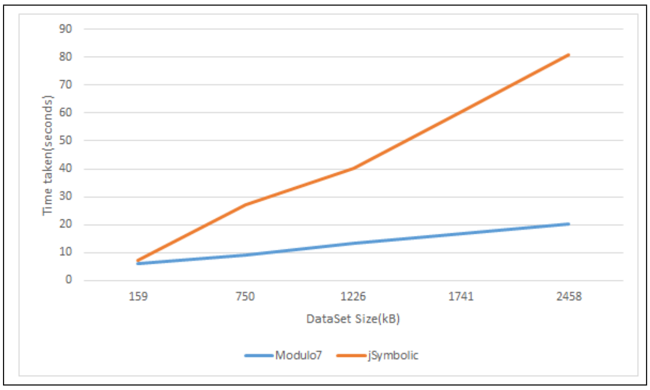

Figure 6. Total time taken for feature extraction, jSymbolic vs Modulo7

6.2 Comparisons against JSymbolic

Some experiments were conducted against the jSymbolic component in the jMIR [11] software suite. Like the jMIR suite, Modulo7 extracts features from songs, but with a different goal (relevance querying and similarity against jMIRs genre classification goals). Modulo7 implements **23** of jSymbolic's **111** features. Moreover both frameworks are written in Java and hence language features do not influence the evaluation.

The experiment involved comparing memory footprints and time taken for extraction of 23 features for both libraries over increasing subset sizes of the Saarland [12] dataset.

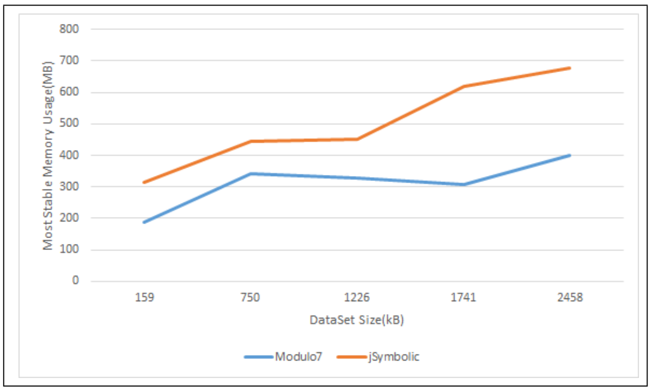

Figure 5. Average Memory taken for feature extraction, jSymbolic vs Modulo7

It can be inferred that Modulo7 outperforms jSymbolic in feature extraction and scales better with increasing data set sizes.

6.3 Similarity Search Experiments

In order to estimate the efficacy of the similarity search algorithms, the million song dataset or MSD [4] was chosen to act as ground truth. MSD contains tags associated with songs (genre, comments about mood of the song and other such keywords) which was extracted by the researchers using the Last FM API. The ground truth of similarity was based on the number of matching tags between two songs. Consider tag sets T_i and T_j for songs in the MSD S_i and S_j. The ground truth similarity is defined as

$$GTSim(S_1, S_2) = \sum_{t_i \in T(S_1)} \sum_{t_j \in T(S_2)} \begin{cases} 1 & t_i == t_j \\ 0 & otherwise \end{cases} \tag{9}$$

The measure was not normalized as MSD had a large number of "junk" tags (irrelevant comments about songs) which would have influenced the ground truth otherwise.

A direct download-able subset of the million song data set was used which contained 10000 songs. A ten fold cross validation was performed (1000 query songs, 9000 data hold out) and average precision and recall are calculated.

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 1. Average Precision and Recall for Melodic Similarity Measures

6.4 Exploratory Query Search Experiments

In order to ascertain the effectiveness of the query engine, certain custom experiments were ran. Queries built from concepts

Purpose	Query	Precision	Recall
Rock Song ID	Q1	0.13	0.98
Sad Song ID	Q2	0.02	0.44
Happy Song ID	Q3	0.018	0.4

Table 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; with a ground truth estimate as song tags being either "rock" or "pop_rock"

Q2 select mp3 from default_database where sadnessindex > 0.15 and scale = minor; with the ground truth estimate being song tags either "sad" or "sad_songs"

Q3 select mp3 from default_database where happinessindex > 0.11 and scale = major; with ground truth being song tags as either "happy" or "happy_song"s

7. CONCLUSION AND FUTURE WORK

8. REFERENCES

[1] *ESSENTIA: an Audio Analysis Library for Music Information Retrieval*.

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

[3] Anon. Satb wikipedia link.

[4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[5] Donald Byrd and Tim Crawford. Problems of music information retrieval in the real world. *Inf. Process. Manage.*, 38(2):249–272, March 2002.

[6] W.T. Glaser, T.B. Westergren, J.P. Stearns, and J.M. Kraft. Consumer item matching method and system, February 21 2006. US Patent 7,003,515.

[7] Ichiro Fuginaga Karl MacMillan, Micheal Droettbroom. Gamera: Optical music recogntion in a new shell.

[8] Daniel Mllensiefen Klaus Frieler. The simile algorthms for melodic similarity.

[9] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[10] Sren Tjagvad Madsen, Gerhard Widmer, and Johannes Kepler. Key-finding with interval profiles.

[11] Cory Mckay. Automatic music classification with jmir, 2010.

[12] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlora 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.

[13] Pascal Ferraroa Pierre Hannaa and Matthias Robinea. On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences. 2007.

[14] Adam M. Stark and Mark D. Plumbley. Real-time chord recognition for live performance. 2009.

[15] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Org. Sound*, 4(3):169–175, December 1999.

[16] Wikifonia. The wikifonia lead sheet collection, 2013.