

# Expressive and efficient retrieval of symbolic musical data

Michael Droettboom, Ichiro Fujinaga,  
Karl MacMillan

The Peabody Institute  
The Johns Hopkins University

{mdboom, ich, karlmac}@peabody.jhu.edu

Mark Patton, James Warner,  
G. Sayeed Choudhury, Tim DiLauro

Digital Knowledge Center  
Milton S. Eisenhower Library

The Johns Hopkins University  
{sayeed, timmo, mpatton, jwarner}@jhu.edu

## ABSTRACT

The ideal content-based musical search engine for large corpora must be both expressive enough to meet the needs of a diverse user base and efficient enough to perform queries in a reasonable amount of time. In this paper, we present such a system, based on an existing advanced natural language search engine. In our design, musically meaningful searching is simply a special case of more general search techniques. This approach has allowed us to create an extremely powerful and fast search engine with minimal effort.

## 1. INTRODUCTION

This paper describes a system for music searching that is expressive enough to perform both simple and sophisticated searches that meet a broad range of user needs. It is also efficient enough to search through a large corpus in a reasonable amount of time. The music search system was created by extending an existing advanced natural language search engine with simple filters and user-interface elements.

This paper will describe the search engine in the context of our larger sheet music digitization project, and relate it to other musical search engines already available for use on the web. Then, the capabilities of the non-music-specific core of the search engine will be described, followed by the extensions necessary to adapt it to music.

## 2. BACKGROUND

The Lester S. Levy Collection of Sheet Music represents one of the largest collections of sheet music available online. The Collection, part of the Special Collections of the Milton S. Eisenhower Library at The Johns Hopkins University, comprises nearly 30,000 pieces of music (Choudhury et al. 2000). It provides a rich, multi-faceted view of life in late 19th and early 20th century America. Scholars from various disciplines have used the Collection for both research and teaching. All works in the public domain are currently available online as JPEG images. The user can browse the collection by category or search based on metadata, such as author, title, publisher, and date. Musical searches, such as finding a particular melodic or rhythmic pattern, will soon be

possible once the collection has been converted to symbolic musical data.

To convert this data, an optical music recognition (OMR) system is being developed (Choudhury et al. 2001). We chose GUIDO as the target representation language due to its simplicity and extensibility (Hoos and Hamel 1997). Having music in a symbolic format opens the collection to sound generation, musicological analysis and, the topic of the present paper, musical searching.

## 3. PRIOR ART

None of the available musical search engines we evaluated met the needs of the diverse user base of the collection, or could handle the large quantity of data in the complete Levy collection. In particular, we evaluated two projects in detail: Themefinder (Huron et al. 2001) and MELDEX (McNab et al. 1997).

### 3.1 Themefinder

Themefinder's goal is to retrieve works by their important themes. These themes are manually determined ahead of time and placed in an incipit database.

One can query the database using five different kinds of search queries: pitch, interval, scale degree, gross contour, and refined contour. These five categories served as the inspiration for a subset of our basic query types. The user can query within an arbitrary subset of these categories and then intersect the results. However, Themefinder does not allow the user to combine these query types within a single query in arbitrary ways. For instance, a user may know the beginning of a melodic phrase, while the ending is more uncertain. Therefore, the user may want to specify exact intervals at the beginning and use gross contours or wild-cards at the end. Unfortunately, in Themefinder, the user must have the same level of certainty about all of the notes in the query. Unfortunately, this is not consistent with how one remembers melodies (McNab et al. 2000).

In addition, Themefinder does not have a notion of rhythmic searching. While its invariance to rhythm can be an asset, it can also be cumbersome when it provides too many irrelevant matches. Figure 1 shows the results of a query where

one result is more relevant than the other. Such queries may return fewer false matches if they could include rhythmic information.

The searches themselves are executed in Themefinder using a brute-force method. The entire database is linearly searched for the given search query string. While this is acceptable for the 18,000 incipits in Themefinder’s largest database, it may not scale well for searching across a full-text database such as the Levy collection.

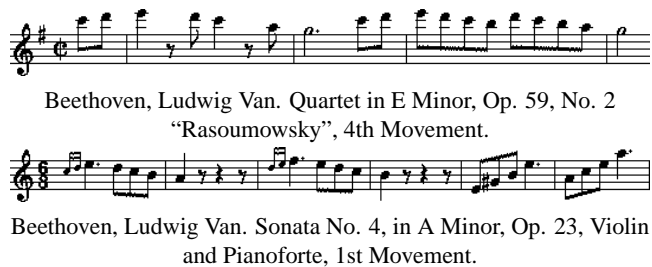


Figure 1: **These two incipits start with the identical set of pitches, [c d e d], but with different rhythmic content. With better rhythmic specificity, irrelevant results could be eliminated.** (<http://www.themefinder.org/>)

### 3.2 MELDEX

The simple text-based query strings in Themefinder are easy to learn and use by those with moderate musical training. MELDEX, however, has a more natural interface for non-musicians. The user sings a melody using a syllable with a strong attack such as “tah.” The pitches of the melody are determined using pitch-tracking, and the rhythm is quantized. The results are used as the search query. The query is approximately matched to melodies in the database using a fast-matching algorithm related to dynamic programming. While this approach is highly effective for non-musicians and simple queries, it is limiting to those wanting more fine-grained control.

## 4. CAPABILITIES

Our musical search engine supports both melodic and rhythmic searches. Search queries can also include the notion of simultaneity. That is, events can be constrained to occur at the same time as other events. The search engine, as described here, is limited to standard-practice Western music, though modifications could be made to support other musical traditions.

### 4.1 Extensibility

Other types of musical searching beyond these core capabilities require additional layers of musical knowledge to be built on top of the search engine. The general design of the search engine encourages such extensibility. Any analytical data that can be derived from the score data can be generated offline (ahead of time) and later used as search criteria.

This data can be generated by new custom tools or existing analysis tools such as the Humdrum toolkit (Huron 1999).

For example, the search engine could be extended to support harmonic searches with respect to harmonic function. Western tonal harmonic theory is ambiguous, making it difficult to objectively interpret and label harmonies. This is a largely unsolved problem that is not the subject of our present research. However, assuming an acceptable solution to these issues could be found, labeling of harmonic function could be implemented as an input filter.

Also, the core search engine does not include any notion of melodic similarity. This is an open problem strongly tied to subjective matters of human perception (Hewlett and Selfridge-Field 1998). It is possible for a specialized front-end to include notions of melodic similarity by generating specialized search queries. The search query language of the core search engine is expressive enough that these advanced features could be added without modifying the core itself.

## 4.2 Meeting diverse user requirements

We define the users of our musical search engine as anyone who wants to access the collection in a musical way. Of course, the needs of different users are greatly varied. A non-musician may want to hum into a microphone to retrieve a particular melody. A copyright lawyer may want to track the origins of a particular melody, even melodies that are merely similar. A musicologist may want to determine the frequency of particular melodic or rhythmic events. To meet these diverse needs, it is necessary to provide different interfaces for different users. The set of interfaces is arbitrary and can be extended as new types of users are identified. It may include graphical applications, web-based forms and applets, or text-based query languages. Audio interfaces, with pitch- and rhythm-tracking may also be included. The purpose of these interfaces is to translate a set of user-friendly commands or interactions into a query string accepted by the search engine. The details of that query can be hidden from the end-user and therefore can be arbitrarily complex.

At present, we have focused our attention on the core search engine itself. In the second phase of the search engine project, the user interfaces will be developed in collaboration with a usability specialist.

## 5. THE CORE SEARCH ENGINE

The core search engine in our system was originally developed for text-based retrieval of scores based on their metadata and full-text lyrics. Its overall design was inspired by recent developments in the field of natural-language searching (DiLauro et al. 2001). These features allow the user to perform search queries using the embedded context in natural languages, such as parts of speech, rhyming scheme, and scansion. While not originally intended for musical searching, it was soon discovered that the core was very well suited for searching across symbolic musical data.

The core itself did not need to be modified to support music searching. Instead, specialized filters and front-ends were added to adapt it to the music domain. In the ingestion stage, the data is filtered to store it in the appropriate indices and partitions (see Section 6). When searching, special user interfaces handle the details of generating search query strings and filtering and displaying the resulting data. Figure 2 shows how the individual parts of the system fit together to ingest and query the data.

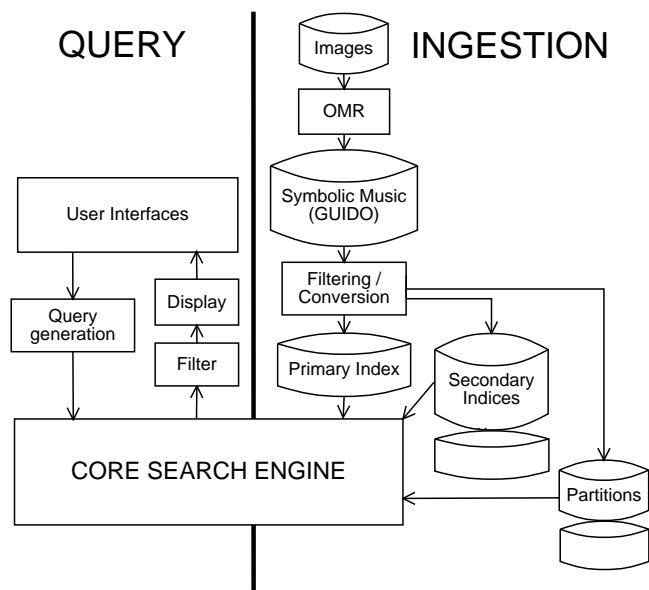


Figure 2: Workflow diagram of the musical search engine.

### 5.1 Inverted lists

Many search engines, including ours, are built on the concept of an inverted list. For a complete discussion of inverted list search engines, see Witten et al. (1999).

Sequential data, such as English prose or melodic data, is stored on disk as a sequence of atoms. In the case of English, the atom is the word and the sequence is simply the ordered words as they appear in sentences and paragraphs. Take for example the following sentence:

To be , or not to be , that is  
the question .

Note that both words and punctuation are treated as indivisible atoms. To search for a particular atom in this string, a computer program would need to examine all thirteen atoms and compare it with a query atom. To increase searching efficiency, an inverted list search engine would store this string internally as:

, → {3, 8}  
 . → {13}  
 be → {2, 7}  
 is → {10}  
 not → {5}  
 or → {4}  
 question → {12}  
 that → {9}  
 the → {11}  
 to → {1, 6}

Here, each atom in the string is stored with a list of numbers indicating the atom's ordinal location within the string. The set of words in the index is called the vocabulary of the index. To search for a particular atom using the index, the program needs only to find that word in the vocabulary and it can easily obtain a list of indices (or pointers) to where that atom is located within the string. Since the vocabulary can be sorted, the lookup can be made faster using hashing or a binary search.

Inverted lists perform extremely well when the size of the vocabulary is small relative to the size of the corpus. In the case of English, of course, the vocabulary is much smaller relative to the size of all the works written in that language. This property also allows us to improve the efficiency of musical searching as we will see below.

## 6. THE MUSICAL SEARCH ENGINE

The musical search capabilities are supported by three main features of the core search engine:

1. **Secondary indices** allow the amount of specificity to vary with each token.
2. **Partitions** allow search queries to be performed upon specific discontinuous parts of the corpus.
3. **Regular expressions** allow advanced pattern matching.



Figure 3: A measure of music, from the Levy collection, used as an example throughout this section. (Guion, D. W., arr. 1930. "Home on the range." New York: G. Schirmer.)

### 6.1 Secondary indices

In the case of music, the searchable atom is not the word, but the musical event. Events include anything that occurs in the

score, such as notes, rests, clefs, and barlines. Each of these events, of course, can have many properties associated with it. For instance, the note  $b^b$  at the beginning of the fragment in Figure 3 has the following properties:

- **Pitch name:**  $b$
- **Accidental:**  $b$
- **Octave:** -1 (first octave below middle- $c$ )
- **Twelve-tone pitch:** 10 (10<sup>th</sup> semitone above  $c$ )
- **Base-40 pitch:** -3 (see Hewlett 1992)
- **Duration:** eighth note
- **Interval to next note:** perfect 4<sup>th</sup>
- **Contour (direction) to next note:** up
- **Scale degree:**  $so$  (5<sup>th</sup> scale degree in  $E^b$  major)
- **Lyric syllable:** “sel-”
- **Metric position:** Beat 1 in a  $\frac{6}{8}$  measure

All of these properties are self-evident, with the exception of base-40 pitch, which is a numeric pitch representation where the intervals are invariant under transposition while maintaining enharmonic spelling (Hewlett 1992). Note also, we use GUIDO-style octave numbers, where the octave containing middle- $c$  is zero, as opposed to ISO-standard octave numbers.

The concept of secondary indices allows the individual properties of each atom to be indexed independently of any other properties. This allows search queries to have arbitrary levels of specificity in each event. The set of properties can be extended to include any kinds of data that can be extracted from the musical source. For example, if the harmonic function of chords could be determined unambiguously, a secondary index containing chord names in Roman numeral notation could be added. In our design, we use secondary indices to handle the properties of events that change from note to note. Continuous properties of events, that are carried from one event to the next, such as clefs, time signatures, and key signatures, are handled using partitions, explained below (see Section 6.2).

### 6.1.1 Ingestion of secondary indices

During the ingestion phase, the source GUIDO data is first converted to an interim format where all of each event’s properties are fully specified. For example, the GUIDO representation of Figure 3 is as follows:

```
[ \clef<"treble"> \key<-3> \time<6/8>
  \lyric<"sel-> b&-1*/8.
  \lyric<"dom"> e&*0
  \lyric<"is"> f
  \lyric<"heard"> g/4
  \lyric<"A"> e&/16
  \lyric<"dis-> d
]
```

Each event is then extended so it is fully specified. In this format, each note event is a tuple of properties:

*pitch-name, accidental, octave, twelve-tone-pitch, base-40-pitch, duration, interval, contour, scale-degree, lyric-syllable, metric-position*

Figure 4 shows the example in fully-specified symbolic representation.

Each one of these fields is used to index the database in a particular secondary corpus. For example, if the notes in the example were labeled 1 through 6, the data in the secondary indices may look something like:

- **Pitch name**
  - a  $\rightarrow \emptyset$
  - b  $\rightarrow \{1\}$
  - c  $\rightarrow \emptyset$
  - d  $\rightarrow \{6\}$
  - e  $\rightarrow \{2, 5\}$
  - f  $\rightarrow \{3\}$
  - g  $\rightarrow \{4\}$
- **Accidentals**
  - n ( $\natural$ )  $\rightarrow \{3, 4, 6\}$
  - & ( $b$ )  $\rightarrow \{1, 2, 5\}$
- **Octave**
  - 1 (octave below middle- $c$ )  $\rightarrow \{1\}$
  - 0 (octave above middle- $c$ )  $\rightarrow \{2, 3, 4, 5, 6\}$
- **Duration**
  - 1/4 (quarter note)  $\rightarrow \{4\}$
  - 1/8 (eighth note)  $\rightarrow \{1, 2, 3\}$
  - 1/16 (sixteenth note)  $\rightarrow \{5, 6\}$

### 6.1.2 Searching using secondary indices

The search query itself is simply a series of events. Each event can be indicated as specifically or as generally as the end user (as represented by a user interface) desires. For example, the following query would match any melodic fragment that begins on a  $b^b$  eighth note, has a sequence of 3 ascending notes, ending on a  $g$ :

$b, \&, 1/8 / / / g$

To execute a search query using secondary indices, the search engine looks up each “parameter” in their corresponding secondary indices, and retrieves tokens in the secondary index. These tokens are then looked up in the primary index, returning a list of positions. These lists are intersected to find the common elements. This list of locations is then filtered to include only those events that are sequenced according to the search query.

### 6.1.3 Supported user interfaces

This design supports a broad range of user interfaces. A text-based user interface may allow a user to be very specific in

```
[ \clef<"treble"> \key<-3> \time<6/8>
  b, &, -1, 10, -3, 1/8, P4, /, so, "sel-", 0
  e, &, 0, 3, 14, 1/8, M2, /, do, "dom", 1/8
  f, n, 0, 5, 20, 1/8, M2, /, re, "is", 1/4
  g, n, 0, 7, 25, 1/4, M3, \, mi, "heard", 3/8
  e, &, 0, 3, 15, 1/16, m2, \, do, "A", 5/8
  d, n, 0, 2, 9, 1/16, M2, \, ti, "dis-", 11/16
]
```

Figure 4: Fully specified symbolic representation of the example in Figure 3.

the query, and then incrementally remove layers of specificity until the desired match is retrieved. An audio-based user interface could be more or less specific depending on the pitch tracker’s confidence in each event.

#### 6.1.4 Efficiency of secondary indices

One of the efficiency problems with this approach is that the vocabularies of the individual secondary indices tend to be quite small, and thus the index lists for each atom are very large. For instance, the “pitch name” secondary index has only seven atoms in its vocabulary ( $a - g$ ). “Accidentals” is even smaller:  $\{bb, b, \natural, \sharp, \times\}$ . Therefore, a search for a  $b^b$  must intersect two very large lists: the list of all  $b$ ’s and the list of all flats. However, the search engine can combine these secondary indices in any desired combination off-line. For example, given the “pitch name” and “accidental” indices, the search engine can automatically generate a hybrid index in which the vocabulary is all possible combinations of pitch names and accidentals. The secondary indices can be automatically combined in all possible combinations, to an arbitrary order.

## 6.2 Partitions

Partitioning can be used to restrict a search query to a particular part of the corpus. Each partition is a description of how to divide the corpus into discontinuous, non-overlapping regions. More specifically, each partition is a file containing a list of regions. Each region within a partition is named and has a list of its start and stop positions.

In our music search engine, the metadata is used to partition the corpus into regions. For example, all works by a given composer would make up a discontinuous region in the “composer” partition. Partitions exist for all types of metadata in the collection, including date, publisher, geographical location, etc.

In addition, we have extended partitioning to include musical elements derived directly from the GUIDO data. Regions are generated from key signatures, clefs, time signatures, measures, movements, repeats, etc. This allows for searching for a particular melody in a particular key and clef, for example.

#### 6.2.1 Ingestion of partition data

When a new work is added to the corpus, the data is partitioned automatically. First, the metadata regions, such as

title, composer, and date, are set to include the entire piece. As the piece is scanned, continuous musical elements, such as clef, key signature, and time signature, are regioned on the fly. Therefore, when the ingestion filter sees a “treble clef” token, all further events are added to the “treble clef” region until another clef token is encountered. Lastly, events are added to the moment regions on an event-by-event basis.

For the example in Figure 3, again assuming the notes are numbered 1 through 6, the partitions may look something like:

- **Title partition**  
“Home on the range”  $\rightarrow$  [1, 6]
- **Clef partition**  
Treble clef  $\rightarrow$  [1, 6]
- **Time signature partition**  
 $\frac{6}{8}$   $\rightarrow$  [1, 6]

#### 6.2.2 Searching using partitions

Extending the example in Section 6.1.2, the user may wish to limit the search to the key signature of  $E^b$ -major:

```
( b,&,1/8 / / / g ) @ key:"E& major"
```

Here the non-partitioned search query is performed as described above, and then the results are intersected with the results of the partition lookup. Since in our case, the entire range of notes [1, 6] is in the key signature of  $E^b$ -major, the query will retrieve the example in Figure 3.

#### 6.2.3 Searching with simultaneity using partitions

Scores are also partitioned at the most atomic level by “moments.” A moment is defined as a point in time when any event begins or ends in any part. Moments almost always contain multiple events, and events can belong to multiple moments (e.g. when a half note is carried over two quarter notes in another part). Each moment within a score is given a unique numeric identifier, and all events active at a given point are included in a moment region. In this way, one can search for simultaneous polyphonic events very efficiently.

To explain this further, Figure 5 shows the example measure with its assigned moment numbers. Each event is assigned to one or more moments so that it can be determined which, if any, of the events are active at the same time. These moment

numbers are used to create regions. For example, the dotted half note in the left hand of the piano part would be assigned to all seven moment regions.

To perform searches involving simultaneity, the query for each part is performed separately, and then the results are intersected based on their moments. Only the query results that occur at the same time (existing in the same moment regions) will be presented to the user.



Figure 5: The example measure of music showing moment numbers.

### 6.3 Regular expressions

The core search engine supports a full complement of POSIX-compliant regular expressions. Regular expressions, a large topic beyond the scope of this paper, are primarily used for pattern-matching within a search string (Friedl 1997).

Many users find regular expressions difficult and cumbersome ways to express searches. However, it is our intent that most of these details will be hidden from the user by appropriate interfaces. For example, regular expressions would be very useful for an interface that allowed searching by melodic similarity. What is important to our present research is that regular expressions are supported in the core search engine, leaving such possibilities open.

## 7. CONCLUSION

Based on existing advanced natural-language search techniques, we have developed an expressive and efficient musical search engine. Its special capabilities include: secondary indices for graduated specificity, partitions for selec-

tive scope and simultaneity, and regular expressions for expressive pattern matching. This allows users with different search needs to access the database in powerful and efficient ways.

## 8. ACKNOWLEDGMENTS

The second phase of the Levy Project is funded through the NSF's DLI-2 initiative (Award #9817430), an IMLS National Leadership Grant, and support from the Levy Family. We would like to thank David Yarowsky of the Department of Computer Science for motivation and discussion.

## 9. REFERENCES

Choudhury, S., T. DiLauro, M. Droettboom, I. Fujinaga, B. Harrington, and K. MacMillan. 2000. Optical music recognition within a large-scale digitization project. *ISMIR 2000 Conference*.

Choudhury, G. S., T. DiLauro, M. Droettboom, I. Fujinaga, and K. MacMillan. 2001. Strike up the score: Deriving searchable and playable digital formats from sheet music. *D-Lib Magazine*: 7(2).

DiLauro, T., G. S. Choudhury, M. Patton, J. W. Warner, and E. W. Brown. 2001. Automated name authority control and enhanced searching in the Levy collection. *D-Lib Magazine*: 7(4).

Friedl, J. E. F. 1997. *Mastering regular expressions*. Sebastopol, CA: O'Reilly.

Hewlett, W. B. 1992. A base-40 number-line representation of musical pitch notation. *Musikometrika* 4: 1-14.

Hewlett, W. B., and E. Selfridge-Field. 1998. *Melodic similarity: Concepts, procedures and applications*. Cambridge, MA: MIT Press.

Hoos, H. H., and K. Hamel. 1997. GUIDO music notation: Specification Part I, Basic GUIDO. Technical Report TI 20/97, Technische Universität Darmstadt.

Huron, D., W. Hewlett, E. Selfridge-Field, et al. 2001. How Themefinder works. <http://www.themefinder.org>

Huron, D. 1999. *Music research using Humdrum: A user's guide*. Menlo Park, CA: Center for Computer Assisted Research in the Humanities.

McNab, R. J., L. A. Smith, D. Bainbridge, and I. H. Witten. 1997. The New Zealand Digital Library MELody inDEX. *D-Lib Magazine*: 3(5).

McNab, R. J., L. A. Smith, I. H. Witten, and C. L. Henson. 2000. Tune retrieval in the multimedia library. *Multimedia Tools and Applications* 10(2/3): 113-32.

Witten, I., A. Moffat, and T. Bell. 1999. *Managing gigabytes*. 2nd Ed. San Francisco: Morgan Kaufmann.