metamidi: a tool for automatic metadata extraction from MIDI files

Tomás Pérez-García, José M. Iñesta, and David Rizo
Departamento de Lenguajes y Sistemas Informáticos
University of Alicante
Alicante, 03080, Spain
e-mail: {tperez,inesta,drizo}@dlsi.ua.es

***Abstract*—The increasing availability of on-line music has motivated a growing interest for organizing, commercializing, and delivering this kind of multimedia content. For it, the use of metadata is of utmost importance. Metadata permit organization, indexing, and retrieval of music contents. They are, therefore, a subject of research both from the design and automatic extraction approaches. The present work focuses on this second issue, providing an open source tool for metadata extraction from standard MIDI files. The tool is presented, the utilized metadata are explained, and some applications and experiments are described as examples of its capabilities.**

I. Introduction

Metadata permit organization, indexing, and retrieval of music contents in digital collections, e.g. digital libraries [10] or on-line music stores, to name but a few. A digital music library requires addressing complex issues related to description, representation, organization, and use of music information [9]. Another application of accurate metadata is related to copyright issues when sharing of music databases among music researches located in different countries. A suitable solution is the use of metadata instead of the source song files. The presented tool along with its helper formats can help in that direction.

Metadata are, therefore, an important subject of research focusing on both the design and automatic extraction approaches. The quality of content-based music recommendations is importantly influenced by the number and quality of available metadata attributes [3].

A number of efforts have been addressed to automatic metadata extraction from music data, both from audio and symbolic files. For example, MPEG-7 [6] deals with multimedia content description. In this context, Pachet in [11] described a project on the automatic computation of those descriptors focusing on digital audio data, while Kleedorfer et al. [4] aimed at generation of metadata attributes automatically extracted from song lyrics. Some of these metadata are utilized, through standard text classification methods, to derive genre, mood, and offensive content tags.

The present work focuses on metadata extraction from standard MIDI files, introducing `metamidi`, an open source tool for metadata extraction from them. A number of works in the literature have deal with related problems, reporting MIR platforms for symbolic file format analysis in different languages like Matlab [2] or java [8]. These works are able to analyze technical properties of music formats, ranging from Humdrum, Lilypond ([5]) or MIDI ([2], [8]). Some of these works are based on solving a music information retrieval (MIR) problem, like genre extraction from content, while others are application-independent. Our proposal falls in the latter category, focusing on metadata like text metaevents, key, tempo, and meter signatures, etc., although some statistical properties based on track contents are also provided. This range of data, permits to address organization oriented problems, like categorization and indexation of MIDI files, but also well know MIR problems like melody part or genre recognition.

A special attention has been paid to interfacing with other programs for metadata analysis, so the software has been designated to operate in command line.

The rest of the paper is structured as follows. The obtained metadata are explained in section II, the `metamidi` tool is presented in section III, and some applications and experiments are described in section IV as examples of its capabilities.

II. MIDI metadata

`metamidi` extracts descriptive, structural, and technical metadata at two different levels: global and track-local metadata. Output is provided in three different formats:

- register format, where metadata are presented in a report style useful for readability,
- line format, oriented to communication with other software for further processing, and
- XML format, providing a standard and modern format for interfacing with other applications. The use of XML-based format has proven to be useful in former works [1].

More details on these formats are provided below in section III.

A. Description of global metadata

In this section the general metadata extracted from a MIDI file, are described. These metadata are extracted from the file storage system, from its header, and from its sequence tracks.

- **Name**, the complete path where the MIDI file is stored.
- **Metaeventtext**, strings contained in text metaevents. This information includes all kind of text metaevents found

in the track 0 of a MIDI file, like author, song name, copyright notices, sequencer, etc.
- **Size**, the size in bytes of the file.
- **Format**, the format of a MIDI file. 0 for single track; 1 for multi-track files, and 2 for multi-sequence files.
- **Number of tracks**, the number of tracks that the file is composed of.
- **Resolution**, the number of ticks per beat.
- **Tempo**, initial value for tempo in beats per minute.
- **Tempo changes**, the number of tempo change events. Tempo tracking has not been implemented, due to the large number of changes that may appear if *accelerandos* or *rittardandos* are present in the sequence.
- **Meter**, the list of number of beats per bar over beat kind separated by commas. The pulse where it changes is showed between brackets.
- **Meter changes**, the number of meter changes.
- **Key**, the list of tonality metaevent in the file. The pulse where it changes is showed between brackets.
- **Key changes**, the number of tonality changes.
- **Instruments**, the numbers of the General MIDI patches utilized in the sequence.
- **Percussion**, the percussion instruments utilized in the sequence according to the pitch values utilized in channel 10, assuming the General MIDI standard percussion map. Percussion instruments have been categorized in three different groups, coded as follows:
  - 1 : instruments usually found in a drum kit,
  - 2 : latin percussion, and
  - 3 : other percussion elements.
- **Sequence duration**, the number of clock ticks where the offset of the last note happens.
- **Has sysex**, a flag showing whether sysex messages appear (value 1) in the sequence or not (value 0).

B. Description of each track metadata:

The second level of metadata are descriptions of each track content. Therefore, the metadata described below are for each track.

- **Metaeventtext**, strings contained in text metaevents included track, like track name, instrument name, lyrics, markers, etc.
- **Track duration**, the number of ticks from the onset of the first note to the offset of the last note in the track.
- **Duration rate**, = track duration / sequence duration.
- **Occupation**, the sum of the number of ticks where notes are sounding in the track.
- **Occupation rate**, = occupation / track duration.
- **Polyphony duration rate**, ticks where two or more notes are sounding / occupation.
- **max polyphony**, the number of maximum simultaneous notes in the track.
- **Avg polyphony**, number of sounding notes in average (weighted by their durations), computed as

$$= \frac{\sum_{\forall n>0} n \times (\#\text{ticks with } n \text{ notes})}{\text{occupation}} \qquad (1)$$

- **Low pitch**, lowest pitch in the track.
- **High pitch**, highest pitch in the track.
- **Modulations**, number of modulation messages.
- **Aftertouches**, number of aftertouch messages.
- **Pitch bends**, number of pitch alteration messages.
- **Program changes**, patch change messages. The pulse where it changes is showed between brackets.

III. THE `metamidi` TOOL

`metamidi` can be freely downloaded from `http://grfia.dlsi.ua.es/gen.php?id=resources`. In the downloading page the reader can find the source code of `metamidi`, the DTDs listed in Table I, and an alternative XSD format [1] that can help in the automatic developing of automatic parsers like the one programmed in Java with XMLBeans [2] that is also provided.

It has been developed in ANSI C 4.2.4 version and tested under Linux. It is designed to operate in the command line, providing the different output formats, with this syntax:

```
metamidi -{r|x|l} file [-o fileoutput]
```

where:

- `-r` : outputs metadata in register format.
- `-l` : outputs metadata in line format. A "|" separates the global metadata from track metadata and also the different track metadata parts. A ";" character is used to separate each feature. "," character separates multi-valuated features.
- `-x` : outputs metadata in XML format. A DTD is provided where tags and attributes are described (see Table I)

When a string metadata is not found, a "$" is given. For missing numerical values, a -1 is displayed.

If no output file is provided, the standard output is utilized, providing a way to pipe the result to other data processing software that uses the standard input as input. This permits to design powerful scripts using off-the-shelf text and data processing utilities.

In table II an example of a two-track MIDI file is displayed when `metamidi` operates in report format.

IV. APPLICATION EXAMPLES

The kind of extracted metadata can be used in a number of applications like MIDI file indexing, organizing, classifying, etc. In this section, two well known MIR problems have been addressed using the provided metadata. Firstly, we will report results on selecting the track containing the melody in a multi-track MIDI file, and secondly, we will present results of an experiment on genre classification using the timbral metadata provided by the patch instrument map in the file.

[1] http://www.w3.org/XML/Schema
[2] http://xmlbeans.apache.org/

TABLE I
PROPOSED DTD

```
<!ELEMENT midifile (external,global,tracks)>
<!ELEMENT external (comments?)>
<!ATTLIST external
  name CDATA #REQUIRED
  size CDATA #REQUIRED
  midiformat (0|1|2) #REQUIRED
  numtracks CDATA #REQUIRED
  resolution CDATA #REQUIRED
>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT global (comments?)>
<!ATTLIST global
  metaeventtext CDATA #REQUIRED
  tempo CDATA #REQUIRED
  tempoChanges CDATA
  meter CDATA #REQUIRED
  meterChanges CDATA
  key CDATA #REQUIRED
  keyChanges CDATA
  instruments CDATA
  percussion (-1|1|2|3)
  duration CDATA #REQUIRED
  hasSysEx (true|false) #REQUIRED
>
<!ELEMENT tracks (track+)>
<!ELEMENT track (comments?)>
<!ATTLIST track
  metaeventtext CDATA #REQUIRED
  channel CDATA #REQUIRED
  duration CDATA #REQUIRED
  durationRate CDATA #REQUIRED
  occupation CDATA #REQUIRED
  occupationRate CDATA #REQUIRED
  polyphonyDurationRate CDATA #REQUIRED
  maxPoliphony CDATA #REQUIRED
  avgPoliphony CDATA #REQUIRED
  low pitch CDATA #REQUIRED
  high pitch CDATA #REQUIRED
  modulations CDATA
  afterTouches CDATA
  pitchBends CDATA
  programChanges CDATA
>
```

TABLE II
OUTPUT EXAMPLE OF A TWO TRACK MIDI FILE USING THE "REPORT" FORMAT.

```
name:              /home/repository/Beethoven/Fur-Elise.mid
text metaevent:          Fur Elise,Ludwig van Beethoven
size:                    9574
format:                  1
num tracks:              3
resolution:              480
tempo:                   75.00
tempo changes:           25
meter:                   4/4(0),3/8(0),3/8(5760)
meter changes:           3
key:                     CM(0)
key changes:             1
instruments:             1
percussion:              -1
duration:                90000
has sysex:               0
------ Features of track 1 ----
text metaevent:          Piano RH
channel:                 1
duration:                89968
duration rate:           1.00
occupation:              75348
occupation rate:         0.84
polyphony duration rate: 0.20
max polyphony:           4
avg polyphony:           1.30
low pitch:               57
high pitch:              100
modulations:             0
aftertouches:            0
pitch bends:             0
program changes:         1(0)
------ Features of track 2 ----
text metaevent:          Piano LH
channel:                 2
duration:                90000
duration rate:           1.00
occupation:              39587
occupation rate:         0.44
polyphony duration rate: 0.19
max polyphony:           3
avg polyphony:           1.23
low pitch:               33
high pitch:              76
modulations:             0
aftertouches:            0
pitch bends:             0
program changes:         1(960)
```

A. Melody track selection

Standard MIDI files are structured as a number of tracks. One of them usually contains the melodic line of the piece, while the other tracks contain accompaniment music. Finding that melody track is very useful for a number of applications, including music retrieval or motif extraction, among others. In [12] the authors introduced a method to identify the track that contains the melody using statistical properties of the musical content and pattern recognition techniques.

Here we are going to use metadata extracted through `metamidi` to perform the same task under a Gaussian approach. For that, we can compute the probability of a track to contain the melody of a MIDI file from different track metadata:

- the amount of music information in the track (occupation rate),
- melody are usually monophonic, so the use of polyphony is important (polyphony duration rate, max polyphony, average polyphony),
- for a melody to be sung it must be *cantabile*, so the pitch ranges are relevant (low pitch and high pitch).

In the training phase, given one of the former metadata, d, its probability distribution function is computed assuming a Gaussian distribution, both for the metadata values of the tracks labeled as melodies, obtaining $P(d|M)$, and for those extracted from non melody tracks, obtaining $P(d|\neg M)$.

During the test phase, in order to select the melody track from a midi file, the a posteriori probabilities for all the tracks are computed using the Bayes theorem:

$$P(M|d_i) = \frac{P(d_i|M)P(M)}{P(d_i|M)P(M) + P(d_i|\neg M)P(\neg M)} \qquad (2)$$

where i subindex refers to the values extracted from the i-th track. The a priori probability for a track of being a melody, $P(M)$ is computed from all the files in the training set as number melody tracks / total number of tracks, and $P(\neg M) =$

$1 - P(M)$.

The final decision is taken using a maximum likelihood decision taking into account those tracks with probabilities higher than a threshold, θ:

$$\hat{t}_M = \arg\max_i \{P(M|d_i) > \theta\} \quad (3)$$

If no track has a $P(M|d_i) > \theta$, the decision of "no melody" is given for that file.

There is also the possibility of combining two probability distributions. The same methodology is applied, but in this case, the two Gaussians involved are multiplied, so $P(d|M) = P(d_A|M) \times P(d_B|M)$, and therefore, the same has to be done with the respective decision thresholds, $\theta = \theta_A \times \theta_B$. The rest of equations remain the same.

The same 6 data sets previously utilized in [12] have been utilized for the experiments. Midi files from pop-rock ("Kar"), jazz ("Jazz"), and classical music ("Clas") are organized in the different data sets in order to test the specificities of this problem depending on the music genre. The sets with a "200" suffix are smaller sets, more uniform in their structure, while the other three are bigger ones, more heterogeneous and, therefore, more difficult for their melody track to be identified. All of them have been manually tagged and multiple melody track and no melody track situations occur.

The system is evaluated as follows. Defining TP as the number of true positive decisions, FP as the number of false positive decisions, TN the number of true negative decisions, and FN as the number of false negative decisions, the evaluation parameters were:

$$\text{Success}: \quad S = 100\frac{TP + TN}{TP + FP + TN + FN} \quad (4)$$

$$\text{Precision}: \quad P = \frac{TP}{TP + FP} \quad (5)$$

$$\text{Recall}: \quad R = \frac{TP}{TP + FN} \quad (6)$$

$$\text{F} - \text{measure}: \quad F = \frac{2RP}{R + P} \quad (7)$$

The figures presented in Table III are the results of a 10-fold cross-validation, where 10 sub-experiments were performed, using 9/10 of the set for training, keeping 1/10 for testing. The results presented are the best obtained for each data set, including the metadata they were obtained with and the utilized threshold.

TABLE III
RESULTS OF MELODY TRACK IDENTIFICATION FOR THE DIFFERENT DATA SETS.

Corpus	Descr.	θ	S%	P	R	F
Clas200	High p.	0.05	98.5	0.99	1.00	0.99
Clas	Avg.Poly	0.20	80.8	0.81	1.00	0.89
Jazz200	Max.Poly & Low p.	0.1^2	86.5	0.88	0.98	0.93
Jazz	Max.Poly & Low p.	0.05^2	82.0	0.83	0.99	0.90
Kar200	Avg.Poly	0.10	80.9	0.81	1.00	0.89
Kar	Avg.Poly	0.10	88.6	0.87	1.00	0.94

B. Timbre-based genre classification

The timbral information provided by `metamidi` can be utilized to infer the genre of the music the MIDI file contains. In fact, there are examples in the literature of using the set of instrument patch numbers (together with other descriptors) to classify MIDI files into music genres [7]. In that work, instrument patch numbers are coded in a vector and a distance-based method to instrument class vectors is used to classify. Nevertheless, the approach proposed here is a probabilistic one, in which a MIDI file has a probability of being of a given genre depending on the probabilities of its instruments to be used in that genre.

We have a set of classes $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ and a labeled training set of songs, $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{X}|}\}$. A MIDI file is represented as a vector $\mathbf{x}$, where each component $x_i \in \{0, 1\}$ codes the absence or presence of the patch t in the *instruments* and *percussion* metadata. The dimensionality of the vectors is $D = 131$, corresponding to the 128 General MIDI patches plus the 3 percussion categories described in section II. A given MIDI file is assigned to the class $c_j \in \mathcal{C}$ with maximum a posteriori probability:

$$P(c_j|\mathbf{x}) = \frac{P(c_j)P(\mathbf{x}|c_j)}{P(\mathbf{x})} \ , \quad (8)$$

where $P(c_j)$ is the a priori probability of class c_j computed in this case as $P(c_j) = 1/|\mathcal{C}|$, assuming that all genres are equally probable, $P(\mathbf{x}) = \sum_{j=1}^{|\mathcal{C}|} P(c_j)P(\mathbf{x}|c_j)$ is a normalization factor, and $P(\mathbf{x}|c_j)$ is the probability of $\mathbf{x}$ being generated by class c_j given by a multivariate Bernoulli distribution of instruments in class c_j, learned from the training set:

$$P(\mathbf{x}|c_j) = \prod_{i=1}^{D} x_i P(t_i|c_j) + (1 - x_i)(1 - P(t_i|c_j)) \quad (9)$$

where $P(t_i|c_j)$ are the class-conditional probabilities of each patch, t_i, in the instrument map, that can be easily calculated by counting the number of occurrences of each instrument in the corresponding class:

$$P(t_i|c_j) = \frac{1 + M_{ij}}{2 + M_j} \quad (10)$$

where M_{ij} is the number of files in class c_j containing the instrument t_i, and M_j is the total number of songs in class c_j. This equation permits to avoid zero probabilities when a previously unseen instrument is found in the test file.

A data base of 856 midi files from popular, jazz, and academic music has been utilized. Popular music data have been separated into three sub-genres: *pop*, *blues*, and *celtic* (mainly Irish jigs and reels). For jazz, three styles have been established: a *pre-bop* class grouping swing, early, and Broadway tunes, *bop* standards, and *bossanovas* as a representative of latin jazz. Finally, academic music has been categorized according to historic periods: *baroque*, *classicism*,

and *romanticism*. This data base is available upon request to the authors.

The input metadata for this application have been easily obtained through the following command (syntax needs to be adapted to the used console language):

```
metamidi -l *.mid | cut -d ";" -f 13,14
```

positions 13 and 14 correspond to *instruments* and *percussion* metadata, respectively, in the output line.

Tables IV and V show the confusion matrices for the two experiments performed, respectively, grouping the files in the three broad music categories (popular, jazz, and academic) and in the nine classes described above. Rows are the ground-truth classes and columns are the system predictions. Both experiments were designed following a 10-fold cross validation scheme.

TABLE IV
CONFUSION MATRIX FOR THE THREE GENRE RECOGNITION TASK.

	Academic	Jazz	Popular
Academic	228	6	1
Jazz	3	295	40
Popular	5	16	262

The overall performance for the three-classes classification was 93 ± 2, showing the good performance of timbral metadata. For the nine-classes problem, a much harder one, the overall classification success was 68 ± 5. Note that a by-chance classifier is expected to achieve 33 and 11%, respectively. Also it is important to see that most of the errors occur now among genres of the same broad category, like classical and romantic music, or pre-bop and bop songs.

TABLE V
CONFUSION MATRIX FOR THE NINE GENRE RECOGNITION TASK.

	bar	clas	rom	pre	bop	bos	cel	blu	pop
baroque	33	4	13	0	0	0	0	0	0
classic	7	0	37	0	0	0	0	0	1
romantic	6	5	102	3	0	0	0	0	0
prebop	0	0	1	136	19	4	0	0	0
bop	0	0	0	70	8	6	0	0	0
bossa	0	0	0	2	1	49	1	0	5
celtic	0	0	1	0	1	2	85	0	0
blues	0	0	0	7	0	1	3	55	9
pop	0	1	0	1	1	13	3	7	64

V. CONCLUSION

Metadata are achieving a growing interest in music information retrieval applications thanks to the high level information they provide. The development of systems for automatically extract them from digital files is one of the main concerns. In this paper, `metamidi` has been presented as an open source software for descriptive, structural, and technical metadata extraction from standard MIDI files. These metadata are extracted from both file properties and track properties.

An illustration of its performance has been presented addressing two MIR problems through the use of the provided metadata: melody track identification and music genre recognition. The good performances obtained show the power of metadata automatic extraction tools for organizing, indexing, and accessing music information in the context of multimedia digital libraries.

ACKNOWLEDGMENT

This work is supported by the Spanish Ministry projects: TIN2006–14932–C02 and Consolider Ingenio 2010 (MIPRCV, CSD2007-00018), both partially supported by EU ERDF.

REFERENCES

[1] A. Baratè, G. Haus, and L. A. Ludovico, *Music representation of score, sound, MIDI, structure and metadata all integrated in a single multilayer environment based on XML*. Hershey, PA: Idea Group Reference, 2007.

[2] T. Eerola and P. Toiviainen, "Mir in matlab: The midi toolbox," in *ISMIR*, 2004, pp. 22–27.

[3] F. Kleedorfer, U. Harr, and B. Krenn, "Making large music collections accessible using enhanced metadata and lightweight visualizations," in *Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS '07)*, Barcelona, Spain, November 2007, pp. 138–144.

[4] F. Kleedorfer, P. Knees, and T. Pohle, "Oh oh oh whoah! towards automatic topic detection in song lyrics," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, September 2008, pp. 287–292.

[5] I. Knopke, "The perlhumdrum and perllilypond toolkits for symbolic music information retrieval," in *Proceedings of the 2008 International Conference on Music Information Retrieval*, 2008, pp. 147–152.

[6] B. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG7: Multimedia Content Description Interface*. West Sussex, England: John Wiley & Sons, 2002.

[7] C. McKay and I. Fujinaga, "Automatic genre classification using large high-level musical feature sets," in *Proc. of the 5th International Conference on Music Information Retrieval, ISMIR 2004*, 2004, pp. 525–530.

[8] C. Mckay and I. Fujinaga, "jsymbolic: A feature extractor for midi files," in *In Int. Computer Music Conf*, 2006, pp. 302–305.

[9] N. Minibayeva and J. W. Dunn, "A digital library data model for music," in *Proc. of the Second ACM/IEEE-CS Joint Conference on Digital Libraries*, Portland, Oregon, 2002, pp. 154–155.

[10] M. Notess and J. Dunn, "Variations2: improving music findability in a digital library through work-centric metadata," in *JCDL'04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*. Tucson, AZ: ACM Press, 2004, p. 422.

[11] F. Pachet, "Metadata for music and sounds: The Cuidado project," in *Proceedings of the CBMI Workshop*, University of Brescia, september 2001, pp. 411–415.

[12] D. Rizo, P. J. P. de León, C. Pérez-Sancho, A. Pertusa, and J. M. Iñesta, "A pattern recognition approach for melody track selection in midi files," in *Proc. of the 7th Int. Symp. on Music Information Retrieval ISMIR 2006*, T. A. Dannenberg R., Lemström K., Ed., Victoria, Canada, 2006, pp. 61–66.