mir_eval

First author
Affiliation1
author1@ismir.edu

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

Third author
Affiliation3
author3@ismir.edu

ABSTRACT

Central to the field of MIR research is the evaluation of algorithms used to extract information from music data. We present mir_eval, an open source software library which provides a transparent and easy-to-use implementation of the most common metrics used to measure the performance of MIR algorithms. In the present work, we enumerate the metrics implemented by mir_eval and quantitatively compare each to an existing implementation. When the score reported by mir_eval differs substantially from the reference, we detail the differences in implementation. We also provide a brief overview of mir_eval's architecture, design, and intended use.

1. EVALUATING MIR ALGORITHMS

Much of the research in Music Information Retrieval (MIR) involves the development of systems that process raw music data to produce semantic information. The goal of these systems is frequently defined as attempting to duplicate the performance of a human listener given the same task [**?**]. A natural way to determine a system's effectiveness might be for a human to study the output produced by the system and judge its correctness. However, this would yield only subjective ratings, and would also be extremely time-consuming when evaluating a system's output over a large corpus of music.

Instead, objective metrics are developed to provide a well-defined way of computing a score which indicates each system's output's correctness. These metrics typically involve a heuristically-motivated comparison of the system's output to a reference which is known to be correct. Over time, certain metrics have become standard for each task, so that the performance of systems created by different researchers can be compared when they are evaluated over the same dataset [?]. Unfortunately, this comparison can be confounded by small details of the implementations or procedures that can have disproportionate impacts on the resulting scores.

For the past 10 years, the yearly Music Information Retrieval Evaluation eXchange (MIREX) has been a forum for comparing MIR algorithms over common datasets [**?**]. By providing a standardized shared-task setting, MIREX has become critically useful to track progress in MIR research. MIREX is built upon the Networked Environment for Music Analysis (NEMA) [**?**], a large-scale system which includes exhaustive functionality for evaluating, summarizing, and displaying evaluation results. The NEMA codebase includes multiple programming languages and dependencies (some of which, e.g. Matlab, are proprietary) so compiling and running it at individual sites is nontrivial. In consequence, the NEMA system is rarely used for evaluating MIR algorithms outside of the setting of MIREX [?]. Instead, researchers typically create their own implementations of common metrics for evaluating their algorithms. These implementations are thus not standardized, and may contain differences in details, or even bugs, that confound comparisons.

These factors motivate the development of a standardized software package which implements the most common metrics used to evaluate MIR systems. Such a package should be straightforward to use and well-documented so that it presents a preferable alternative to in-house development for MIR researchers. In addition, it should be community-developed and transparently implemented so that all design decisions are easily understood and open to discussion and improvement.

Following these criteria, we present mir_eval, a software package which intends to provide an easy and standardized way to evaluate MIR systems. This paper first discusses the architecture and design of mir_eval in Section 2, then, in Section 3, describes all of the tasks covered by mir_eval and the metrics included. In order to validate our implementation decisions, we compare mir_eval to existing software in Section 4. Finally, we discuss and summarize our contributions in Section 5.

2. mir_eval'S ARCHITECTURE

mir_eval is a Python library which includes metrics for the following tasks: Beat detection, chord recognition, pattern discovery, structural segmentation, melody extraction, onset detection, and source separation. Each task is given its own submodule (e.g. mir_eval.beat), and each metric is defined as a separate function in each submodule (e.g. mir_eval.beat.f_measure). Each task submodule also includes common data pre-processing steps for the task. Every metric function includes detailed documentation, example usage, input validation, and references to

the original paper which defined the metric. `mir_eval` also includes a submodule `io` which provides convenience functions for loading in task-specific data from common file formats. For readability, all code follows the PEP8 style guide [**?**]. `mir_eval`'s only dependencies outside of the Python standard library are the free and open-source SciPy/Numpy [**?**] and scikit-learn [**?**] Python libraries.

In order to simplify the usage of `mir_eval`, it is packaged with a set of "evaluator" scripts, one for each task. These scripts include all code necessary to load in data, pre-process it, and compute all metrics for a given task. The evaluators allow for `mir_eval` to be called directly from the command line so that no knowledge of Python is necessary. They are also distributed as executables for Windows and Mac OS X, so that `mir_eval` may be used with no dependencies installed.

3. TASKS INCLUDED IN `mir_eval`

3.1 Beat Detection

The aim of a beat detection algorithm is to report the times at which a typical human listener might tap their foot to a piece of music. As a result, most metrics for evaluating the performance of beat tracking systems involve computing the error between the estimated beat times and some reference list of beat locations. Many metrics additionally compare the beat sequences at different metric levels in order to deal with the ambiguity of tempo [**?**].

`mir_eval` includes the following metrics for beat tracking, which are defined in detail in [**?**]: The **F-measure** of the beat sequence, where an estimated beat is considered correct if it is sufficiently close to a reference beat; **Cemgil's score**, which computes the sum of Gaussian errors for each beat; **Goto's score**, a binary score which is 1 when some specific heuristic criteria are met; **McKinney's P-score**, which computes the cross-correlation of the estimated and reference beat sequences represented as impulse trains; **continuity-based scores** which compute the proportion of the beat sequence which is continuously correct; and finally the **information gain** of the beat error histogram to a uniform distribution.

3.2 Chord Recognition

Despite being one of the oldest MIREX tasks, evaluation methodology and metrics for automatic chord recognition is an ongoing topic of discussion. Several recent articles address issues and concerns with vocabularies, comparison semantics, and other lexicographical challenges unique to chord recognition []. Ultimately, the source of this difficulty stems from the inherent subjectivity in "spelling" a chord name and the level of detail a human observer can provide in a reference annotation [**?**]. As a result, a consensus has yet to be reached regarding the single best approach to comparing two sequences of chord labels, and instead are often compared over a set of rules, e.g Major-Minor, Sevenths, with or without inversions, and so on.

Thanks to the previous efforts of Harte [], text representations of chord labels adhere to a standardized format, consisting of a root, quality, extensions, and a bass note; of these, only the root is strictly required. However, in order to efficiently compare chords in a variety of different ways, it is helpful to first translate a given chord label $\mathcal{C}$ into a numerical representation, shown in Figure **??**. In this example, a $G : 7(9)/5$ is mapped to split into 4 pieces of information: one, the root is mapped to an absolute pitch class $\mathcal{R}$, in $[0, 11]$, where $C \rightarrow 0$, $C\sharp/D\flat \rightarrow 1$, etc; two, the quality is mapped to a root-invariant 12-dimensional bit vector $\mathcal{Q}$ by setting the scale degrees of the quality; three, any extensions are applied (via addition or omission) to the quality bit vector as scale degrees in a single octave, resulting in pitch vector $\mathcal{P}$; and four, the bass interval (5) is translated to the relative scale degree in semitones $\mathcal{B}$. Note that the add-9 is rolled into a single octave as an add-2. This is a matter of convenience as extended chords (9's, 11's or 13's) are traditionally resolved to a single-octave equivalent, but the bit-vector representation could be easily expanded to represent such information.

Having gone through this bit of effort, it is now straightforward to compare chords along the five rules used in MIREX 2013:

1. Root:
   (a) $\mathcal{R}_{est} == \mathcal{R}_{ref}$
   (b) $\forall \mathcal{Q}_{ref}$
2. Major-Minor: Rule 1.a, plus
   (a) $\mathcal{Q}_{est} == \mathcal{Q}_{ref}$
   (b) $\mathcal{Q}_{ref} \in \{Maj, min\}$
3. Major-Minor w/Inversions: Rule 2, plus
   (a) $\mathcal{B}_{ref} \in \mathcal{Q}_{ref}$
4. Sevenths: Rule 1.1, plus
   (a) $\mathcal{Q}_{est} == \mathcal{Q}_{ref}$
   (b) $\mathcal{Q}_{ref} \in \{Maj, min, Maj7, min7, 7\}$
5. Sevenths w/Inversions: Rule 4, plus
   (a) $\mathcal{B}_{ref} \in \mathcal{Q}_{ref}$

Following recent trends in MIREX, an overall score is computed by weighting each comparison by the duration of its interval, over all intervals; stated another way, this is the piecewise continuous-time integral of the intersection of two chord sequences, $(\mathbf{C}_{ref}, \mathbf{C}_{est})$, expressed as follows:

$$S(\mathbf{C}_{ref}, \mathbf{C}_{est}) = \frac{1}{T} \int_{t=0}^{T} \mathcal{C}_{ref}(t) == \mathcal{C}_{est}(t) \quad (1)$$

Here, this is achieved here by forming the union of the boundaries in each sequence, and summing the time intervals of the correct ranges. Note that equivalence is subject to one of the rules defined previously.

Finally, the total score over a set of N items is given by a discrete summation, where the importance of each score, S_n, is weighted by the duration, T_n, of each annotation:

$$S_{total} = \frac{\sum_{n=0}^{N} T_n * S_n}{\sum_{n=0}^{N} T_n} \quad (2)$$

3.3 Pattern Discovery

This task aims to evaluate algorithms that identify musical patterns (i.e. short fragments or melodic ideas that repeat at least twice) both from audio and symbolic representations. Given the novelty of this task [1], the evaluation metrics are likely to be modified in further editions. Nonetheless, Collins put together all the previously existent metrics and a few new ones for this task in its first appearance in MIREX [**?**], which resulted in 19 different scores, each one (except for the two that evaluate the algorithm execution time) implemented in MIR-eval and described below:

- **Standard F-measure, Precision, and Recall ($\mathbf{F}_1$, P, R)**: This metric, composed of three scores, checks if the prototype patterns of the reference match possible key-transposed patterns in the prototype patterns of the estimations. Since the sizes of these prototypes must be equal, this metric is quite restrictive and it tends to be 0 most of the time (see 2013 MIREX results).
- **Establishment F-measure, Precision, and Recall ($\mathbf{F1}_{est}$, $\mathbf{P}_{est}$, $\mathbf{R}_{est}$)**: These scores evaluate the amount of patterns that were successfully identified by the estimated results, no matter how many occurrences they found. In other words, this metric captures the ability of the algorithm to *establish* that the estimated patterns are actually contained in the reference annotation.
- **Occurrence F-measure, Precision, and Recall ($\mathbf{F1}_{occ}$, $\mathbf{P}_{occ}$, $\mathbf{R}_{occ}$)**: Independently of how many patterns were correctly established, we may also want to know how well the algorithm finds all their respective occurrences throughout the piece. This metric aims to quantize this *occurrence* retrieval quality of the algorithm. This metric has a tolerance parameter c to allow differences when evaluating the similarity between occurrences, and in MIREX they use $c = .75$ and $c = .5$.
- **Three-layer F-measure, Precision, and Recall ($\mathbf{TLF}_1$, $\mathbf{P}_3$, $\mathbf{R}_3$)**: These scores can be seen as an improved version of the standard metrics. They capture both the establishment of the patterns and the occurrence retrieval in a single set of scores, being more permissive than the standard evaluation.
- **First N patterns metrics ($\mathbf{FFTP}_{est}$, FFP)**: This includes the first N patterns target proportion establishment recall, and the first N patterns three-layer precision. By analyzing the first N patterns only, we evaluate the ability of the algorithm of sorting the identified patterns based on their relevance. In MIREX, $N = 5$.

[1] The first year to appear on MIREX was 2013.

3.4 Segmentation

Evaluation criteria for segmentation fall into two categories: boundary annotation, and structural annotation. *Boundary annotation* is the task of predicting the times at which structural changes occur, such as when a *verse* transitions to a *refrain*. *Structural annotation* is the task of assigning labels to detected segments. The estimated labels may be arbitrary strings — such as A, B, C, *etc.* — and they need not describe functional concepts.

In both tasks, we assume that annotations express a partitioning of the track into intervals $(s_i, t_i)_{i=1}^{m}$, where $s_0 = 0$ denotes the beginning of track, t_m denotes the end of the track, and $t_i = s_{i+1}$. Structural annotation additionally requires that each interval be assigned a label y_i.

`mir_eval` implements the following MIREX-compatible boundary detection metrics:

Boundary detection: precision, recall, and F-measure for detecting boundary events within a tolerance window $w \in \{0.5, 3\}$ [**?**];

Boundary deviation: median absolute time difference from a reference boundary to its nearest estimated boundary, and vice versa [?],

and the following structure annotation metrics

Pairwise classification: precision, recall, and F-measure for classifying pairs of sampled time instants as belonging to the same structural component [**?**];

Rand index: [2] reference and estimated annotations are sampled uniformly throughout the track, and the induced clusterings are compared by the Rand index [**?**];

Normalized conditional entropy: reference and estimated labels are sampled uniformly, and interpreted as samples of random variables Y_R, Y_E, which are computed by estimating the conditional entropy of Y_R given Y_E (*under-segmentation*) and Y_E given Y_R (*over-segmentation*) [**?**].

3.5 Melody Extraction

3.5.1 Task definition

Melody extraction algorithms aim to produce a sequence of frequency values corresponding to the pitch of the dominant melody from a musical recording [**?**]. The estimated pitch is represented as a time series of fundamental frequency (f_0) values in Hz sampled on a fixed time grid (e.g. every 10 ms). To evaluate the estimated sequence, a reference sequence is generated by running a pitch tracker on the monophonic melody track (requiring access to the multi-track recording session) and manually correcting any mistakes made by the pitch tracker. The estimate is then evaluated against the reference by comparing the two frequency sequences on a frame-by-frame basis, and computing five global measures, first used in the MIREX 2005 AME evaluation [**?**]. The goal of these measures, defined below, is to assess the algorithm's performance on two subtasks of melody extraction:

(1) pitch estimation, i.e. how well the algorithm estimates the pitch of the melody, and (2) voicing detection, i.e. how well the algorithm determines when the melody is present in a frame (a *voiced* frame) or absent (an *unvoiced* frame). To allow evaluation of these two subtasks independently, a melody extraction algorithm can provide a frequency estimate even for frames it has determined to be unvoiced.

3.5.2 Evaluation measures

The following definitions are taken from [?] with permission from the authors. Let the estimated melody pitch frequency vector be $\mathbf{f}$, and the true sequence (reference) be $\mathbf{f}^*$. Let us also define a voicing indicator vector $\mathbf{v}$, whose τ^{th} element $v_\tau = 1$ when a melody pitch is detected, with corresponding ground truth $\mathbf{v}^*$. We also define an "unvoicing" indicator $\bar{v}_\tau = 1 - v_\tau$. Recall that an algorithm may report an estimated melody pitch ($f_\tau > 0$) even for times where it reports no voicing ($v_\tau = 0$). Then the measures are:

- **Voicing Recall Rate**: The proportion of frames labeled as melody frames in the reference that are estimated as melody frames by the algorithm.

$$\text{Rec}_{\text{vx}} = \frac{\sum_\tau v_\tau v_\tau^*}{\sum_\tau v_\tau^*} \tag{3}$$

- **Voicing False Alarm Rate**: The proportion of frames labeled as non-melody in the reference that are mistakenly estimated as melody frames by the algorithm.

$$\text{FA}_{\text{vx}} = \frac{\sum_\tau v_\tau \bar{v}_\tau^*}{\sum_\tau \bar{v}_\tau^*} \tag{4}$$

- **Raw Pitch Accuracy**: The proportion of melody frames in the reference for which f_τ is considered correct (i.e. within half a semitone of the ground truth f_τ^*).

$$\text{Acc}_{\text{pitch}} = \frac{\sum_\tau v_\tau^* \mathcal{T}\left[\mathcal{M}(f_\tau) - \mathcal{M}(f_\tau^*)\right]}{\sum_\tau v_\tau^*} \tag{5}$$

  where $\mathcal{T}$ is a threshold function defined by:

$$\mathcal{T}[a] = \begin{cases} 1 & \text{if } |a| < 0.5 \\ 0 & \text{if } |a| \geq 0.5 \end{cases} \tag{6}$$

  and $\mathcal{M}$ maps a frequency in Hertz to a melodic axis as a real-valued number of semitones above an arbitrary reference frequency f_{ref}:

$$\mathcal{M}(f) = 12 \log_2 \left(\frac{f}{f_{\text{ref}}}\right) \tag{7}$$

- **Raw Chroma Accuracy**: As raw pitch accuracy, except that both the estimated and reference f_0 sequences are mapped onto a single octave. This gives a measure of pitch accuracy which ignores octave errors, a common error made by melody extraction systems:

$$\text{Acc}_{\text{chroma}} = \frac{\sum_\tau v_\tau^* \mathcal{T}\left[\langle \mathcal{M}(f_\tau) - \mathcal{M}(f_\tau^*) \rangle_{12}\right]}{\sum_\tau v_\tau^*} \tag{8}$$

  Octave equivalence is achieved by taking the difference between the semitone-scale pitch values modulo 12 (one octave), where

$$\langle a \rangle_{12} = a - 12 \lfloor \frac{a}{12} + 0.5 \rfloor. \tag{9}$$

- **Overall Accuracy**: this measure combines the performance of the pitch estimation and voicing detection tasks to give an overall performance score for the system. It is defined as the proportion of all frames correctly estimated by the algorithm, where for non-melody frames this means the algorithm labeled them as non-melody, and for melody frames the algorithm both labeled them as melody frames and provided a correct f_0 estimate for the melody (i.e. within half a semitone of the reference):

$$\text{Acc}_{\text{ov}} = \frac{1}{L} \sum_\tau v_\tau^* \mathcal{T}\left[\mathcal{M}(f_\tau) - \mathcal{M}(f_\tau^*)\right] + \bar{v}_\tau^* \bar{v}_\tau \tag{10}$$

  where L is the total number of frames.

The performance of an algorithm on an entire music collection for a given measure is obtained by averaging the per-song scores for that measure over all songs in the collection.

3.5.3 Discussion

In the measure definitions provided above, it is assumed that both the estimate and reference sequences are sampled using the same time grid (hop size). In practice, however, this is not always the case, since the time grid of the reference depends on the hop size used by the pitch tracker that produced it, and similarly the time grid of the estimate depends on the specific melody extraction algorithm that produced it. This means that the sequences must be resampled onto a common time-grid prior to the computation of the measures. For the MIREX AME task, any sequence (be it reference or estimate) that is not sampled using a 10 ms hop size is resampled using 0^{th}-order interpolation, i.e. each frequency value in the target sequence is set to its nearest neighbour (in time) from the source sequence. This kind of interpolation can potentially have a detrimental effect on the evaluation, depending on the difference between the source and target time grids. In particular, it can result in artificially low scores for sequences with rapidly changing pitch values, such as opera singing with deep vibrato.

For this reason, the melody evaluator in `mir_eval` uses 1^{st}-order (linear) interpolation by default in order to map the reference and estimate sequences onto a common time-grid. Assuming the original timestamps of both sequences correspond to the *center* of each analysis frame (as they should), using 1^{st}-order rather than 0^{th}-order interpolation means the results returned by `mir_eval` are lower-bounded by the MIREX results and are, in our view, more accurate. By default, the estimate is resampled using the reference's time grid so that the reference can be used

without any modification. Alternatively, the user can specify a hop size, in which case both reference and estimate are resampled onto the new time-grid defined by the hop size.

3.6 Onset Detection

The goal of an onset detection algorithm is to automatically determine when notes are played in a piece of music. As is also done in beat tracking and segment boundary detection, the primary method used to evaluate onset detectors is to first determine which estimated onsets are "correct", where correctness is defined as being within a small window of a reference onset [**?**]. From this, **precision**, **recall**, and **F-measure** scores are computed.

3.7 Source Separation

4. COMPARISON TO EXISTING IMPLEMENTATIONS

In order to validate the design choices made in `mir_eval`, it is useful to compare the scores it reports to those reported by an existing evaluation system. Beyond pinpointing intentional differences in implementation, this process can also help find and fix bugs in either `mir_eval` or the system it is being compared to.

For each task covered by `mir_eval`, we obtained a collection of reference and estimated annotations and computed a score for each metric using `mir_eval` and the evaluation system being compared to. In order to facilitate comparison, we ensured that all parameters and preprocessing used by `mir_eval` were equivalent to the reference system unless otherwise explicitly noted. Then, for each reported score, we computed the relative change between the scores as their absolute difference divided by their mean, or

$$\frac{|s_m - s_c|}{(s_m + s_c)/2}$$

where s_m is the score reported by `mir_eval` and s_c is the score being compared to. We then computed the average relative change across all examples in the obtained dataset for each score.

For the beat detection, chord recognition, structural segmentation, and onset detection tasks, MIREX releases the the output of submitted algorithms, the ground truth annotations, and the reported score for each example in each data set. We therefore can directly compare `mir_eval` to MIREX for these tasks by collecting all reference and estimated annotations, computing each metric for each example, and comparing the result to the score reported by MIREX. We chose to compare against the results reported in MIREX 2013 for all tasks.

In contrast to the tasks above, MIREX does not release ground truth annotations or algorithm output for the melody extraction and pattern discovery tasks. As a result, we compared `mir_eval`'s output on smaller development datasets for these tasks. For melody extraction, the ADC2004 dataset used by MIREX is publicly available. We performed melody extraction using the "SG2" algorithm evaluated in 2011 [**?**] and compared `mir_eval`'s reported scores to those of MIREX. For pattern discovery, we used the development dataset released by Collins [?] and used the algorithms submitted by Nieto and Farbood [**?**] for MIREX 2013 to produce estimated patterns. We evaluated the estimated patterns using the MATLAB code released by Collins [?]. The number of algorithms, examples, and total number of scores for all tasks are summarized in Table **??**.

The resulting average relative change for each metric is presented in Table **??**. For may of the metrics, the average relative change is less than a few tenths of a percent, indicating that `mir_eval` is equivalent up to rounding/precision errors. In the following sections, we enumerate the known implementation differences which account for the larger average relative changes.

4.1 Non-greedy matching of events

Brian writes this section

4.2 McKinney's P-score

When computing McKinney's P-score [?], the beat sequences are first converted to impulse trains sampled at a 10 millisecond resolution. Because this sampling involves quantizing the beat times to a 10ms grid, shifting both beat sequences by a constant offset can result in substantial changes in the P-score. As a result, In `mir_eval`, we normalize the beat sequences by subtracting from each reference and estimated beat location the minimum beat location in either series. In this way, the smallest beat in the estimated and reference beat sequences is always 0 and the metric remains the same even when both beat sequences have a constant offset applied. This is not done in MIREX (which uses the Beat Evaluation Toolbox [?]), and as a result, we observe a considerable average relative change for the P-score metric.

4.3 Information Gain

The Information Gain metric [?] involves the computation a histogram of the per-beat errors. The Beat Evaluation Toolbox uses a non-uniform histogram binning where the first and last bins are 1/2 of the width of the rest of the bins. It also uses one more bin than is specified by the user. `mir_eval` uses a standard uniformly-binned histogram with as many bins as are specified by the user. As a result, the Information Gain score reported by `mir_eval` differs substantially from that reported by MIREX.

5. DISCUSSION

Chord					Melody				
Root	M/m	M/m + Inv	7ths	7ths + Inv	Overall	Raw Pitch	Raw Chroma	Voicing R	Voicing
0.007	0.1634	1.005	0.483	0.899	0.070%	0.087%	0.114%	0.000%	10.09
Segmentation									
NCE-Over	NCE-under	Pairwise F	Pairwise P	Pairwise R	Rand	F@.5	P@.5	R@.5	F@
3.182%	11.082%	0.937%	0.942%	0.785%	0.291%	0.429%	0.088%	1.021%	0.393
Segmentation				Beat					
P@3	R@3	Ref-est dev.	Est-ref dev.	F-Measure	Cemgil	Goto	P-score	CMLc	CML
0.094%	0.954%	0.935%	0.000%	0.703%	0.035%	0.054%	0.877%	0.161%	0.143
Beat			Pattern						
AMLc	AMLt	In. Gain	Pat	Pat	Pat	Pat	Pat	Pat	Pat
0.137%	0.139%	9.174%	???	???	???	???	???	???	???
Pattern									
Pat	Pat	Pat	Pat	Pat	Pat	Pat	Pat	Pat	Pat
???	???	???	???	???	???	???	???	???	???
Pattern				Onset			Source Separation		
Pat	Pat	Pat	Pat	F-measure	Precision	Recall	SDR	SIR	SAR
???	???	???	???	0.165%	0.165%	0.165%	???	???	???