EXTRACTING GROUND TRUTH INFORMATION FROM MIDI FILES: A MIDIFESTO

Colin Raffel and Daniel P. W. Ellis
LabROSA
Department of Electrical Engineering
Columbia University
New York, NY

ABSTRACT

MIDI files abound and provide a bounty of information for music informatics. We enumerate the types of information available in MIDI files and describe the steps necessary for utilizing them. We also quantify the reliability of this data by comparing it to human-annotated ground truth. The results suggest that developing better methods to leverage information present in MIDI files will facilitate the creation of MIDI-derived ground truth for audio content-based MIR.

1. MIDI FILES

MIDI (Music Instrument Digital Interface) is a hardware and software standard for communicating musical events. First proposed in 1983 [1], MIDI remains a highly pervasive standard both for storing musical scores and communicating information between digital music devices. Its use is perhaps in spite of its crudeness, which has been lamented since MIDI's early days [21]; most control values are quantized as 7-bit integers and information is transmitted at the relatively slow (by today's standards) 31,250 bits per second. Nevertheless, its efficiency and well-designed specification make it a convenient way of formatting digital music information.

In the present work, we will focus on MIDI files, which in a simplistic view can be considered a compact way of storing a musical score. MIDI files are specified by an extension to the MIDI standard [2] and consist of a sequence of MIDI messages organized in a specific format. A typical MIDI file contains timing and meter information in addition to a collection of one or more "tracks", each of which contains a sequence of notes and control messages. The General MIDI standard [3] further specifies a collection of 128 instruments on which the notes can be played, which standardizes the playback of MIDI files and has therefore been widely adopted.

When paired with a General MIDI synthesizer, MIDI files have been used as a sort of *semantic* audio codec, with entire songs only requiring a few kilobytes of storage. The early availability of this "coding method", combined with the expense of digital storage in the 90s, made MIDI files a highly pervasive method of storing and playing back songs before the advent of the MP3. Even after high-quality perceptual audio codecs were developed and storage prices plummeted, MIDI files remained in use in resource-scarce settings such as karaoke machines and cell phone ringtones. As a result, there is an abundance of MIDI file transcriptions of music available today; through a large-scale web scrape, we obtained 178,561 MIDI files with unique MD5 checksums. Given their wide availability, we believe that MIDI files are underutilized in the Music Information Retrieval community.

In this paper, we start by outlining the various sources of information present in MIDI files and reference relevant works which utilize them in Section 2. In Section 3, we discuss the steps needed to leverage MIDI-derived information as ground truth for content-based MIR. We then establish a baseline for the reliability of MIDI-derived ground truth by comparing it to handmade annotations in Section 4. Finally, in Section 5, we argue that improving the process of extracting information from MIDI files is a viable path for creating large amounts of ground truth data for MIR.

2. INFORMATION AVAILABLE IN MIDI FILES

While various aspects of MIDI files have been used in MIR research, to our knowledge there has been no unified overview of the information they provide, nor a discussion of the availability and reliability of this information in MIDI transcriptions found "in the wild". We therefore present an enumeration of the different information sources in a typical MIDI file and discuss their applicability to different MIR tasks. Because not all MIDI files are created equal, we also computed statistics about the presence and quantity of each information source across our collection of 178,561 MIDI files; the results can be seen in Figure 1 and will be discussed in the following sections.

2.1 Transcription

MIDI files are specified as a collection of "tracks", where each track consists of a sequence of MIDI events on one of 16 channels. Commonly used MIDI events are note-on and note-off messages, which together specify the start

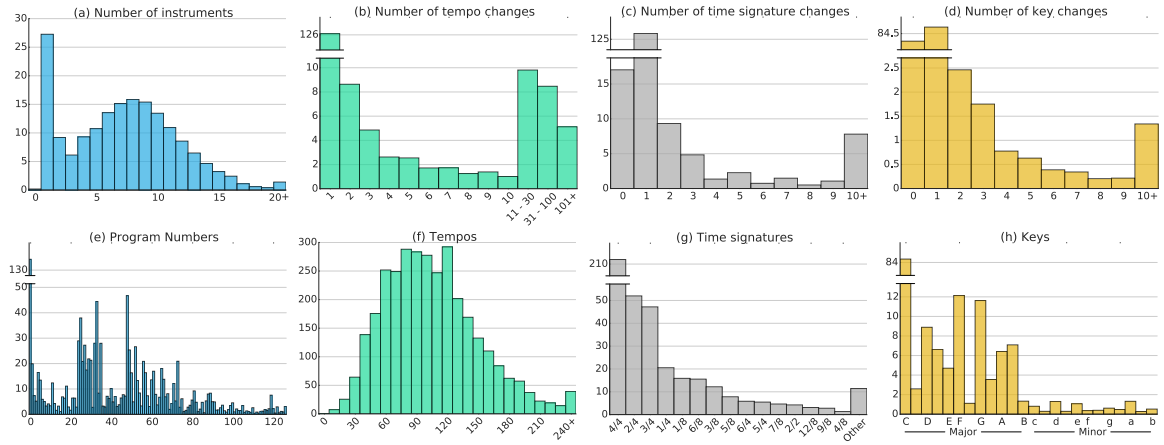

Figure 1: Statistics about sources of information in 178,561 unique MIDI files scraped from the internet. Histograms in the top row show the number of MIDI files which had a given number of events for different event types; in the bottom row, we show distributions of the different values set by these events across all MIDI files. All counts are reported in thousands. For example, about 125,000 MIDI files had a single time signature change event, and about 210,000 4/4 time signature changes were found in all of our MIDI files.

and end time of notes played at a given pitch on a given channel. Various control events also exist, such as pitch bends, which allow for finer control of the playback of the MIDI file. Program change events determine which instrument these events are sent to. The General MIDI standard defines a correspondence between program numbers and a predefined list of 128 instruments. General MIDI also specifies that all notes occurring on MIDI channel 10 play on a separate percussion instrument, which allows for drum tracks to be transcribed. The distribution of the total number of program change events (corresponding to the number of instruments) across the MIDI files in our collection and the distribution of these program numbers are shown in Figures 1(a) and 1(e) respectively. The four most common program numbers (shown as the four tallest bars in Figure 1(e)) were 0 ("Acoustic Grand Piano"), 48 ("String Ensemble 1"), 33 ("Electric Bass (finger)"), and 25 ("Acoustic Guitar (steel)").

This specification makes MIDI files naturally suited to be used as transcriptions of pieces of music, due to the fact that they can be considered a sequence of notes played at different "velocities" (intensities) on a collection of instruments. As a result, many MIDI files are transcriptions and are thus commonly used as training data for automatic transcription systems (see [32] for an early example). This type of data also benefits score-informed source separation methods, which utilize the score as a prior to improve source separation quality [15]. An additional natural use of this information is for "instrument activity detection", i.e. determining when certain instruments are being played over the course of a piece of music. Finally, the enumeration of note start times lends itself naturally to onset detection, and so MIDI data has been used for this task [4].

2.2 Music-Theoretic Features

Because many MIDI files are transcriptions of music, they can also be used to compute high-level musicological characteristics of a given piece. Towards this end, the software library `jSymbolic` [20] includes functionality to extract a wide variety of features, including instrumentation, rhythm, and pitch statistics. Similarly, `music21` [9] provides a general-purpose framework for analyzing collections of digital scores (including MIDI files). Computing these features on a collection of MIDI transcriptions is valuable for computational musicology and can enable data-driven corpus studies. For example, [10] discusses the use of `music21` and `jSymbolic` to extract features from scores and uses them to distinguish music from different composers and musical traditions.

2.3 Meter

Timing in MIDI files is determined by two factors: The MIDI file's specified "resolution" and tempo change events. Each event within the MIDI file specifies the number of "ticks" between it and the preceding event. The resolution, which is stored in the MIDI file's header, sets the number of ticks which correspond to a single beat. The amount of time spanned by each tick is then determined according to the current tempo, as set by tempo change events. For example, if a MIDI file has a resolution of 220 ticks per beat and the current tempo is 120 beats per minute,[1] each tick would correspond to $60/(120 * 220) = 0.002\overline{27}$ seconds. If a MIDI event in this file is specified to occur 330 ticks after the previous event, then it would occur $330 * 0.002\overline{27} = .75$ seconds later.

The timing in a MIDI file can vary over time by including many tempo change events. In practice, as shown in Figure 1(b), most MIDI files only contain a single tempo change and are therefore transcribed at a fixed tempo. However, there are many MIDI files in our collection which have a large number of tempo change events (as indicated by the rightmost bars in Figure 1(b)). We have found that this is a common practice for making the timing of a MIDI transcription closely match that of an audio

[1] Actually, tempo change events specify the number of microseconds per quarter beat, but this can be readily converted to beats per minute.

recording of the same song. Despite the fact that the default tempo for a MIDI file is 120 beats per minute, Figure 1(f) demonstrates that a wide range of tempos are used. In practice, we find that this is due to the fact that even when a single tempo event is used, it is often set so that the MIDI transcription's tempo approximates that of an audio recording of the same song.

Time signature change events further augment MIDI files with the ability to specify time signatures, and are also used to indicate the start of a measure. By convention, MIDI files have a time signature change at the first tick, although this is not a requirement. Because time signature changes are relatively rare in western popular music, the vast majority of the MIDI files in our collection contain a single time signature change, as seen in Figure 1(c). Despite the fact that 4/4 is the default time signature for MIDI files and is pervasive in western popular music, a substantial portion (about half) of the time signature changes were not 4/4, as shown in Figure 1(g).

Because MIDI files are required to include tempo information in order to specify their timing, it is straightforward to extract beat locations from a MIDI file. By convention, the first (down)beat in a MIDI transcription occurs at the first tick. Determining the beat locations in a MIDI file therefore involves computing beat locations starting from the first tick and adjusting the tempo and time signature according to any tempo change or time signature change events found. Despite this capability, to our knowledge MIDI files have not been used as ground truth for beat tracking algorithms. However, [19] utilized a large dataset of MIDI files to study drum patterns using natural language processing techniques.

2.4 Key

An additional useful event in MIDI files is the key change event. Any of the 24 major or minor keys may be specified. Key changes simply give a suggestion as to the tonal content and do not affect playback, and so are a completely optional meta-event. As seen in Figure 1(d), this results in many MIDI files omitting key change events altogether. A further complication is that a disproportionate number (about half) of the key changes in the MIDI files in our collection were C major, as shown in Figure 1(h). This disagrees with corpus studies of popular music, e.g. [8] which found that only about 26% of songs from the Billboard 100 were in C major. We believe this is because many MIDI transcription software packages automatically insert a C major key change at the beginning of the file.

2.5 Lyrics

Lyrics can be added to MIDI transcriptions by the use of lyrics meta-events, which allow for timestamped text to be included over the course of the song. This capability enables the common use of MIDI files for karaoke; in fact, a separate file extension ".kar" is often used for MIDI files which include lyrics meta-events. Occasionally, the generic text meta-event is also used for lyrics, but this is not its intended use. In our collection, we found 23,801 MIDI files (about 13.3%) which had at least one lyrics meta-event.

2.6 What's Missing

Despite the wide variety of information sources available in MIDI files outlined in the previous sections, there are various types of information which are not possible (or not common) to store in MIDI files. While the General MIDI specification includes the vocal instruments "Choir Aahs", "Voice Oohs", "Synth Choir", "Lead 6 (voice)" and "Pad 4 (choir)", in practice there is no specific program number (or numbers) which is consistently used to transcribe vocals. As a result, in a given MIDI file there is no reliable way of determining which instrument is a transcription of the vocals in a song. Furthermore, because a substantial portion of MIDI files were designed for karaoke, the vocals may not be transcribed at all.

While the MIDI specification does include "track name", "program name", and "instrument name" meta-events, they are not standardized and so are not used consistently. It follows that there is no simple way to retrieve the "melody" from a MIDI transcription, although the fact that all instruments are transcribed separately can make its estimation more straightforward than for audio files. For example, [31] explores the use of simple features such as the average velocity and note range within a track to predict whether it is a melody, and also finds that in a small dataset the track name reliably indicates a melody track 44.3% of the time. Similarly, [23] uses heuristic features and a random forest classifier to predict with high accuracy whether a track is a melody.

There is also no explicit way for MIDI files to include chord labels or structural segmentation boundaries (e.g. "verse", "chorus", "solo"). While this would in principle be possible thanks to the generic MIDI "text" meta-event, we have yet to find any MIDI files which store this information. Nevertheless, estimating chords in particular is greatly facilitated by the presence of a ground truth transcription. Both `music21` [9] and `melisma` [30] include functionality for estimating chord sequences from symbolic data. Rhodes et al. [29] also proposed a symbolic chord estimation method using Bayesian Model Selection, which was shown to outperform `melisma` on a dataset of Beatles MIDI files in [14].

While text meta-events could also be used to store song-level metadata (song title, artist name, etc.) in a MIDI file, we seldom encountered this. There is no standardized way to store this metadata in a MIDI file, although we found that a minority of the filenames in our collection indicated the song title and occasionally the artist name. The lack of a metadata specification also inhibits the attribution of MIDI transcriptions to the person who transcribed them.

3. UTILIZING MIDI FILES AS GROUND TRUTH

Utilizing MIDI files as ground truth information for audio content-based MIR tasks requires the following: First, the compact low-level binary format used by MIDI files must be parsed so that the information can be readily extracted. Second, the artist and song of a MIDI file must be determined so it can be paired with a corresponding audio recording. Finally, for many uses, the MIDI file must be aligned in time with its matching audio.

3.1 Extracting Information

The information sources enumerated in Section 2 are not readily available from MIDI files due to fact that they follow a low-level binary protocol. For example, in order to extract the time (in seconds) of all onsets from a given instrument in a MIDI file, note events which occur on the same track and channel as program change events for the instrument must be collected and their timing must be computed from their relative ticks using the global tempo change events. Fortunately, various software libraries have been created to facilitate this process. `pretty_midi` [24] simplifies the extraction of useful information from MIDI transcriptions by taking care of most of the low-level parsing needed to convert the information to a more human-friendly format. It contains functions for retrieving beats, onsets, and note lists from specific instruments, and the times and values of key, tempo, and time signature changes. It also can be used to modify MIDI files, as well as to convert them to synthesized audio or a spectrogram-like piano roll representation. The aforementioned `jSymbolic` contains an extensive collection of routines for computing musicological features from MIDI files. Finally, both `music21` and `melisma` are capable of inferring high-level music information from symbolic data of various types, including MIDI.

3.2 Matching

Apart from metadata-agnostic corpus studies such as [19], determining the song a given MIDI file represents is usually required. Matching a given MIDI file to, for example, a corresponding entry in the Million Song Dataset [5] can be beneficial even in experiments solely involving symbolic data analysis because it can provide additional metadata for the track including its year, genre, and user-applied tags. Utilizing information in a MIDI file for ground truth in audio content-based MIR tasks further requires that it be matched to an audio recording of the song, but this is made difficult by the lack of a standardized method for storing song-level metadata in MIDI files (as discussed in Section 2.6). Content-based matching offers a solution; for example, early work by Hu et al. [17] assigned matches by finding the smallest dynamic time warp (DTW) distance between spectrograms of MIDI syntheses and audio files across a corpus. This approach is prohibitively slow for very large collections of MIDI and/or audio files, so [25] explored learning a mapping from spectrograms to downsampled sequences of binary vectors, which greatly accelerates DTW. [27] provided further speed-up by mapping entire spectrograms to fixed-length vectors in a Euclidean space where similar songs are mapped close together. These methods make it feasible to match a MIDI file against an extremely large corpus of music audio.

3.3 Aligning

There is no guarantee that a MIDI transcription for a given song was transcribed so that its timing matches an audio recording of a performance of the song. For the many types of ground truth data that depend on timing (e.g. beats, note transcription, or lyrics), the MIDI file must therefore have its timing adjusted so that it matches that of the performance. Fortunately, score-to-audio alignment, of which MIDI-to-audio alignment is a special "offline" case, has received substantial research attention. A common method is to use DTW or another edit-distance measure to find the best alignment between spectrograms of the synthesized MIDI and the audio recording; see [26] or [14] for surveys.

In practice, audio-to-MIDI alignment systems can fail when there are overwhelming differences in timing or deficiencies in the transcription, e.g. missing or incorrect notes or instruments. Ideally, the alignment and matching processes would automatically report the success of the alignment and the quality of the MIDI transcription. [26] explores the ability of DTW-based alignment systems to report a "confidence" score indicating the success of the alignment. We do not know of any research into automatically determining the quality of a MIDI transcription.

4. MEASURING A BASELINE OF RELIABILITY FOR MIDI-DERIVED INFORMATION

Given the potential availability of ground truth information in MIDI transcriptions, we wish to measure the reliability of MIDI transcriptions found "in the wild". A straightforward way to evaluate the quality of MIDI-derived annotations is to compare them with hand-made annotations for the same songs. Given a MIDI transcription and human-generated ground truth data, we can extract corresponding information from the MIDI file and compare using the evaluation metrics employed in the Music Information Retrieval Evaluation eXchange (MIREX) [12]. We therefore leveraged the Isophonics Beatles annotations [18] as a source of ground truth to compare against MIDI-derived information. MIDI transcriptions of these songs are readily available due to The Beatles' popularity.

Our choice in tasks depends on the overlap in sources of information in the Isophonics annotations and MIDI files. Isophonics includes beat times, song-level key information, chord changes, and structural segmentation. As noted in Section 2, beat times and key changes may be included in MIDI files but there is no standard way to include chord change or structural segmentation information. We therefore performed experiments to evaluate the quality of key labels and beat times available in MIDI files. Fortuitously, these two experiments give us an insight into both song-level timing-agnostic information (key) and alignment-dependent timing-critical information (beats). To carry out these experiments, we first manually identified 545 MIDI files from our collection which had filenames indicating that they were transcriptions of one of the 179 songs in the Isophonics Beatles collection; we found MIDI transcriptions for all but 11. The median number of MIDI transcriptions per song was 2; the song "Eleanor Rigby" had the most, with 14 unique transcriptions.

4.1 Key Experiment

In our first experiment, we evaluated the reliability of key change events in MIDI files. We followed the MIREX methodology for comparing keys [13], which proceeds as follows: Each song may only have a single key. All keys

Source	Score	Comparisons
MIDI, all keys	0.400	223
MIDI, C major only	0.167	146
MIDI, non-C major	0.842	77
QM Key Detector	0.687	151
`whatkeyisitin.com`	0.857	145

Table 1: Mean scores achieved, and the number of comparisons made, by different datasets compared to Isophonics Beatles key annotations.

must be either major or minor, e.g. "C# Major" and "E minor" are allowed but "D Mixolydian" is not. An estimated key is given a score of 1.0 when it exactly matches a ground truth key, 0.5 when it is a perfect fifth above the ground truth key, 0.3 when it is a relative major or minor, 0.2 when it is a parallel major or minor, and 0.0 otherwise. We utilized the evaluation library `mir_eval` [28] to compute this score.

The Isophonics annotations mostly follow this format, except that 21 songs contained multiple key annotations and 7 others contained non-major/minor keys. To simplify evaluation, we discarded these songs, leaving 151 ground truth key annotations. Of our 545 Beatles MIDIs, 221 had no key change event and 5 had more than one, which we also omitted from evaluation. This left 223 MIDI files for which we extracted key annotations and compared them to valid Isophonics annotations. Because of the preponderance of C major key change events noted in Section 2.4, we also evaluated MIDI-derived C Major and non-C major instances separately to see whether they were less reliable.

As a baseline, we also extracted keys using the QM Vamp Key Detector plugin [7] whose underlying algorithm is based on [22] which finds the key profile best correlated with the chromagram of a given song. This plugin achieved the highest score in MIREX 2013, and has been the only key detection algorithm submitted in 2014 and 2015. This gives us a reasonable expectation for a good audio content-based key estimator. To determine the extent to which human annotators agree on key labels, we also collected key annotations for Beatles' songs from `whatkeyisitin.com`. As with the Isophonics key annotations, some songs had multiple and/or modal key labels; we discarded these and ended up with 145 labels for songs in the Isophonics dataset.

The mean scores resulting from comparing each dataset to the Isophonics annotations can be seen in Table 1. At first glance, the mean score of 0.4 achieved by MIDI key change messages is discouraging. However, by omitting all MIDI files with C major key events (which achieved a mean score of 0.167), the mean score jumps to 0.842. This is comparable to the human baseline, and is substantially higher than the algorithmically estimated score. We therefore propose that *non-C major* MIDI key change events are as reliable as hand-annotated labels, but that C major key annotations in MIDI files are effectively useless.

4.2 Beat Experiment

Utilizing many of the sources of information in MIDI files depends on the precise alignment of a given MIDI file to an audio recording of a performance of the same song. We therefore performed an additional experiment to evaluate the quality of MIDI-derived beat annotations, which are evaluated on the scale of tens of milliseconds. Producing valid beat annotations from a MIDI file requires not only that the file's meter information is correct, but also that it has been aligned with high precision.

To align our Beatles MIDI files to corresponding audio recordings, we used the scheme proposed in [26], which was found by a large-scale search over common DTW-based audio-to-MIDI alignment systems. We give an outline of this method below; for a full description, see [26]. First, the MIDI file is synthesized using the `fluidsynth` program. Log-magnitude, constant-Q spectrograms of the synthesized MIDI and audio recording are extracted and their pairwise cosine distance matrix is computed. The lowest-cost path through the distance matrix is then found using DTW, with the constraint that the path must span at least 96% of the shorter of the two spectrograms. In addition, all paths are penalized by adding the median value of the distance matrix each time a frame in one spectrogram is mapped to multiple frames in the other. Finally, a "confidence score" is computed as the mean pairwise distance along the lowest-cost path, normalized by mean of the submatrix spanned by the path.

We followed [26] exactly, except for the following changes: First, instead of computing spectrograms with a hop size of 46 ms, we used 23 ms. This finer timescale is more appropriate for the beat evaluation metrics we will use, which have tolerances measured in tens of milliseconds. Second, the confidence scores computed using the method of [26] lie in the range $[0.5, 1.0]$ where 0.5 corresponds to "highly confident" and 1.0 corresponds to "likely wrong"; we mapped this linearly to a more easily-interpretable range of $[0.0, 1.0]$ where higher scores mean higher confidence.

We used `pretty_midi`'s `get_beats` method to extract beat times from our 545 Beatles MIDI files, and adjusted each beat's timing according to the MIDI file's alignment to corresponding audio recordings. For evaluation, we used the *F-measure*, *Any Metric Level Total*, and *Information Gain* metrics described in [11], as implemented in `mir_eval`. As a baseline, we also computed beat locations using the `DBNBeatTracker` from the `madmom` software library,[2] which is based on the algorithm from [6]. This represents a state-of-the-art general-purpose beat tracker which, on the Beatles data, can reliably produce high-quality annotations. If MIDI-derived beat annotations are to be taken as ground truth, they must achieve scores similar to or higher than the `DBNBeatTracker`.

We visualize the resulting scores in Figure 2. Because we don't expect beats to be extracted accurately from MIDI files that are poor transcriptions or when alignment failed, we plotted each MIDI file as a single point whose x coor-

[2] `https://github.com/CPJKU/madmom`

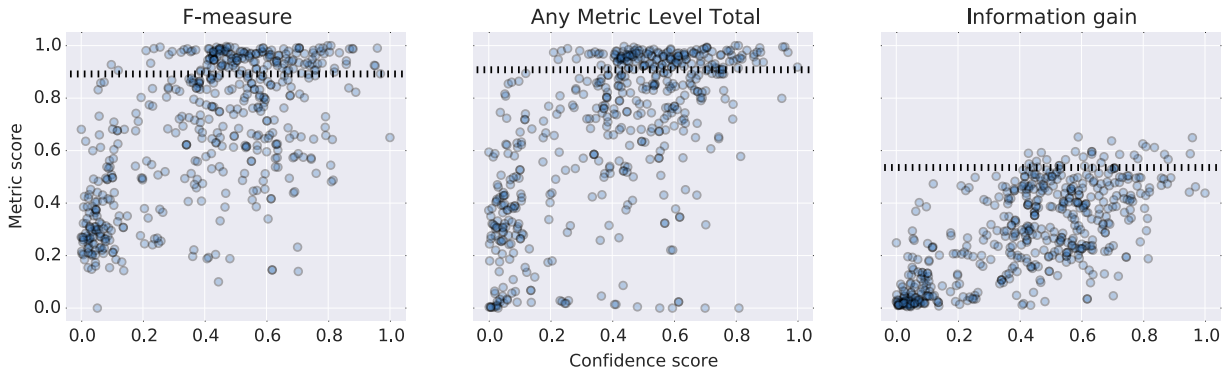

Figure 2: Beat evaluation metric scores (compared to Isophonics beat annotations) and alignment confidence scores achieved by different audio-to-MIDI alignments of Beatles MIDI files, with each shown as a blue dot. Mean scores for each metric achieved by the `DBNBeatTracker` [6] are shown as dashed lines.

dinate corresponds to the alignment confidence score and whose y coordinate is the resulting evaluation metric score achieved. Ideally, all points in these plots would be clustered in the bottom left (corresponding to failed alignments with low confidence scores) or top right (corresponding to a successful alignment and beat annotation extraction with a high confidence score). For reference, we plot the mean score achieved by the `DBNBeatTracker` as dotted lines for each metric. From these plots, we can see that in many cases, MIDI-derived annotations achieve near-perfect scores, particularly for the *F-Measure* and *Any Metric Level Total* metrics. However, there is no reliable correspondence between high confidence scores and high evaluation metric scores. For example, while it appears that a prerequisite for an accurate MIDI-derived beat annotation is a confidence score above .5, there are many MIDI files which had high confidence scores but low metric scores (appearing in the bottom-right corner of the plots in Figure 2).

We found that this undesirable behavior was primarily caused by a few issues: First, it is common that the alignment system would produce alignments which were slightly "sloppy", i.e. were off by one or two frames (corresponding to 23 milliseconds each) in places. This had less of an effect on the *F-measure* and *Any Metric Level Total* metrics, which are invariant to small temporal errors up to a certain threshold, but deflated the *Information Gain* scores because this metric rewards consistency even for fine-timing errors. Second, many MIDI files had tempos which were at a different metric level than the annotations (e.g. double, half, or a third of the tempo). This affected the *Any Metric Level Total* scores the least because it is invariant to these issues, except for the handful of files which were transcribed at a third of the tempo. Finally, we found that the confidence score produced by the alignment system is most reliable at producing a low score in the event of a total failure (indicated by points in the bottom left of the plots in Figure 2), but was otherwise insensitive to the more minor issues that can cause beat evaluation metrics to produce low scores.

5. DISCUSSION

Our results suggest that while MIDI files have the potential to be valuable sources of ground truth information, their usage may come with a variety of caveats. However, due to the enormous number of MIDI transcriptions available, we believe that developing better methods to leverage information present in MIDI files is a tantalizing avenue for obtaining more ground truth data for music information retrieval. For example, while C major key annotations cannot be trusted, developing a highly reliable C major vs. non-C major classification algorithm for symbolic data (which would ostensibly be much more tractable than creating a perfect general-purpose audio content-based key estimation algorithm) would enable the reliable usage of all key change messages in MIDI files. Further work into robust audio-to-MIDI alignment is also warranted in order to leverage timing-critical information, as is the neglected problem of alignment confidence score reporting. Novel questions such as determining whether all instruments have been transcribed in a given MIDI file would also facilitate their use as ground truth transcriptions. Fortunately, all of these tasks are made easier by the fact that MIDI files are specified in a format from which it is straightforward to extract pitch information. Any techniques developed towards this end could also be applied to other ubiquitous symbolic digital music formats such as MusicXML [16].

To facilitate further investigation, all 178,561 of the MIDI files we obtained in our web scrape (including our collection of 545 Beatles MIDIs) are available online, [3] as well as all of the code used in the experiments in this paper. [4] We hope this data and discussion facilitates a groundswell of MIDI utilization in the MIR community.

6. ACKNOWLEDGMENTS

We thank Eric J. Humphrey and Hunter McCurry for discussion about key evaluation, Rafael Valle for investigations into MIDI beat tracking, and our anonymous reviewers for their suggestions.

[3] `http://colinraffel.com/projects/lmd`
[4] `https://github.com/craffel/midi-ground-truth`

7. REFERENCES

[1] International MIDI Association. MIDI musical instrument digital interface specification 1.0. 1983.

[2] International MIDI Association. Standard MIDI files. 1988.

[3] International MIDI Association. General MIDI level 1 specification. 1991.

[4] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[5] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596, 2011.

[6] Sebastian Böck, Florian Krebs, and Gerhard Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 603–608, 2014.

[7] Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell. MIREX 2015 entry: Vamp plugins from the centre for digital music. In *11th Music Information Retrieval Evaluation eXchange*, 2015.

[8] Dave Carlton. I analyzed the chords of 1300 popular songs for patterns. This is what I found. `http://www.hooktheory.com/blog/`, June 2012.

[9] Michael Scott Cuthbert and Christopher Ariza. `music21`: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 637–642, 2010.

[10] Michael Scott Cuthbert, Christopher Ariza, and Lisa Friedland. Feature extraction and machine learning on symbolic music using the `music21` toolkit. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 387–392, 2011.

[11] Matthew E. P. Davies, Norberto Degara, and Mark D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Queen Mary University of London, 2009.

[12] J. Stephen Downie. The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[13] Andreas Ehmann, Kris West, Mert Bay, Kahyun Choi, and Yun Hao. MIREX task: Audio key detection. `http://music-ir.org/mirex/wiki/2016:Audio_Key_Detection`, 2016.

[14] Sebastian Ewert, Meinard Müller, Verena Konz, Daniel Müllensiefen, and Geraint A. Wiggins. Towards cross-version harmonic analysis of music. *IEEE Transactions on Multimedia*, 14(3):770–782, 2012.

[15] Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark Plumbley. Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3):116–124, 2014.

[16] Michael Good. MusicXML for notation and analysis. In Walter B. Hewelett and Eleanor Selfridge-Field, editors, *The virtual score: representation, retrieval, restoration*, volume 12, pages 113–124. MIT Press, 2001.

[17] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 185–188, 2003.

[18] Matthias Mauch, Chris Cannam, Matthew Davies, Simon Dixon, Christopher Harte, Sefki Kolozali, Dan Tidhar, and Mark Sandler. OMRAS2 metadata project 2009. In *10th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2009.

[19] Matthias Mauch and Simon Dixon. A corpus-based study of rhythm patterns. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, pages 163–168, 2012.

[20] Cory McKay and Ichiro Fujinaga. `jSymbolic`: A feature extractor for MIDI files. In *Proceedings of the International Computer Music Conference*, pages 302–5, 2006.

[21] F. Richard Moore. The dysfunctions of MIDI. *Computer music journal*, 12(1):19–28, 1988.

[22] Katy Noland and Mark Sandler. Signal processing parameters for tonality estimation. In *Audio Engineering Society Convention 122*, 2007.

[23] Pedro José Ponce de León Amador, José Manuel Iñesta Quereda, and David Rizo Valero. Mining digital music score collections: melody extraction and genre recognition. In Peng-Yeng Yin, editor, *Pattern Recognition Techniques, Technology and Applications*, chapter 25, pages 559–590. InTech, 2008.

[24] Colin Raffel and Daniel P. W. Ellis. Intuitive analysis, creation and manipulation of MIDI data with `pretty_midi`. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2014.

[25] Colin Raffel and Daniel P. W. Ellis. Large-scale content-based matching of MIDI and audio files. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 234–240, 2015.

[26] Colin Raffel and Daniel P. W. Ellis. Optimizing DTW-based audio-to-MIDI alignment and matching. In *41st IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 81–85, 2016.

[27] Colin Raffel and Daniel P. W. Ellis. Pruning subsequence search with attention-based embedding. In *41st IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 554–558, 2016.

[28] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. `mir_eval`: A transparent implementation of common MIR metrics. In *Proeedings of the 15th International Society for Music Information Retrieval Conference*, pages 376–372, 2014.

[29] Christophe Rhodes, David Lewis, and Daniel Müllensiefen. Bayesian model selection for harmonic labelling. In *Mathematics and Computation in Music*, pages 107–116. Springer, 2007.

[30] Daniel Sleator and David Temperley. The `melisma` music analyzer. `http://www.link.cs.cmu.edu/melisma`, 2001.

[31] Michael Tang, Yip Chi Lap, and Ben Kao. Selection of melody lines for music databases. In *Proceedings of the 24th International Computer Software and Applications Conference*, pages 243–248, 2000.

[32] Robert J. Turetsky and Daniel P. W. Ellis. Ground-truth transcriptions of real music from force-aligned MIDI syntheses. In *Proceedings of the 4th International Society for Music Information Retrieval Conference*, pages 135–141, 2003.