The Semantic Web MIDI Tape: An Interface for Interlinking MIDI and Context Metadata

Conference or Workshop Item

How to cite:

For guidance on citations see FAQs.

© 2018 The Authors

Version: Accepted Manuscript

Link(s) to article on publisher’s website:
http://saam.semanticaudio.ac.uk/

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.
The Semantic Web MIDI Tape: An Interface for Interlinking MIDI and Context Metadata

Albert Meroño-Peñuela
Dept. of Computer Science,
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
albert.merono@vu.nl

Reinier de Valk
Jukedeck Ltd.
London, United Kingdom
reinier@jukedeck.com

Enrico Daga
Knowledge Media Institute,
The Open University
Milton Keynes, United Kingdom
enrico.daga@open.ac.uk

Marilena Daquino
Dept. of Classical Philology and
Italian Studies, University of Bologna
Bologna, Italy
marilena.daquino2@unibo.it

Anna Kent-Muller
Dept. of Music,
University of Southampton,
Southampton, United Kingdom
alkm1g12@soton.ac.uk

ABSTRACT
The Linked Data paradigm has been used to publish a large number of musical datasets and ontologies on the Semantic Web, such as MusicBrainz, Acoustichainz, and the Music Ontology. Recently, the MIDI Linked Data Cloud has been added to these datasets, representing more than 300,000 pieces in MIDI format as Linked Data, opening up the possibility for linking fine-grained symbolic music representations to existing music metadata databases. Despite the dataset making MIDI resources available in Web data standard formats such as RDF and SPARQL, the important issue of finding meaningful links between these MIDI resources and relevant contextual metadata in other datasets remains. A fundamental barrier for the provision and generation of such links is the difficulty that users have at adding new MIDI performance data and metadata to the platform. In this paper, we propose the Semantic Web MIDI Tape, a set of tools and associated interface for interacting with the MIDI Linked Data Cloud by enabling users to record, enrich, and retrieve MIDI performance data and related metadata in native Web data standards. The goal of such interactions is to find meaningful links between published MIDI resources and their relevant contextual metadata. We evaluate the Semantic Web MIDI Tape in various use cases involving user-contributed content, MIDI similarity querying, and entity recognition methods, and discuss their potential for finding links between MIDI resources and metadata.

KEYWORDS
MIDI, Linked Data, score enrichment, metadata, MIR

ACM Reference Format:

1 INTRODUCTION
Symbolic music representations express fundamental information for musicians and musicologists. Musicians, apart from using it for performing, may use it to look for similar performances of the same piece, while musicologists may seek for style similarities between artists. The MIDI format, a symbolic music representation, is widely used by musicians, amateurs and professionals alike, and music information retrieval (MIR) researchers because of its flexibility. For example, MIDI files are easy to produce by playing a MIDI instrument, and generated content can be manipulated by controlling parameters such as pitch and duration, as well as by changing instrument and rearranging or recomposing the various tracks. Moreover, MIDI files are much smaller than audio files; thus, vast collections are easier to be stored and reused. Many MIDI datasets are publicly available online for MIR tasks, such as the Lakh MIDI dataset [20]; the Essen Folksong collection, searchable with ThemeFinder; and the user-generated Reddit collection.[2]

However, music notation alone is not always sufficient to answer more sophisticated questions, e.g., Which pieces reference the same topic? Which pieces are related to a specific cultural resource, such as the soundtrack of a movie? Which pieces are from the same geographical region? In order to answer these and other questions, music notation needs to be interlinked with contextual information. Unfortunately, current datasets generally lack good quality descriptive metadata (e.g., provenance, artist, genre, topic, similar pieces, alternative notations, etc.), making retrieval challenging.

Recently, many music metadata datasets have been published on the Semantic Web, following the Linked Data principles to address meaningful relations between music and context information [4]. Nonetheless, semantically interlinking the MIDI datasets with contextual information, and between themselves, is not a trivial task. Recently, the MIDI Linked Data Cloud [17] has been proposed as a hub of semantic MIDI data, publishing an RDF representation
of the contents of 300,000 MIDI files from the Web. Due to this representation, the MIDI data is ready to be enriched with contextual information and linked to music metadata. However, the usability of the dataset is currently hampered by several issues: (1) the MIDI files collected include little metadata; (2) there is no method to identify different versions of the same piece (i.e., to represent musical similarity in the MIDI Linked Data Cloud); and (3) including user-generated content, e.g., contributed metadata, but also original MIDI performances, is difficult.

In this paper, we address these issues by proposing the Semantic Web MIDI Tape, an interface and set of tools for users to play, record, enrich, and retrieve MIDI performances in native Linked Data standards. Our goal is to use these user interactions in order to create two different kinds of meaningful links: those between published MIDI resources (such as similar song versions, common notes, motifs, etc.), and those connecting MIDI resources to their relevant contextual metadata (such as a song’s interpreter, author, year of publication, etc.). To bootstrap these links, we propose a first approach leveraging the ShapeM melody similarity algorithm [24], to generate the MIDI-to-MIDI links; and DBpedia Spotlight [3], to generate the MIDI-metadata links. Therefore, we combine the user provided data with these methods in order to provide users with relevant and extended query answers, and to enrich the MIDI Linked Data platform. The Semantic Web MIDI Tape leverages the MIDI Linked Data Cloud, enriching the RDF representation of MIDI events with links to external data sources, and underlining the importance of notation data and metadata. Therefore, we mix benefits derived from using MIDI data, and we exploit linking to the Linked Data Cloud so as to (1) enhance the expressivity of music data at scale, and (2) enable knowledge discovery in the symbolic music domain. The Semantic Web MIDI Tape allows users to interact—in a bottom-up fashion—with their MIDI instrument, convert their performance data into RDF, and upload it to the MIDI Linked Data Cloud. Then, the platform lets users listen to their performances again, and retrieve MIDI performances related to theirs, based on their MIDI similarity and contextual information.

More specifically, the contributions of this paper are:

- an updated description of the MIDI Linked Data Cloud dataset, with two important additions based on MIDI similarity and entities recognised in the metadata (Section 3);
- a description of the Semantic Web MIDI Tape, an interface for writing and reading MIDI performance information and associated metadata natively in RDF (Section 4);
- an experiment to evaluate the effectiveness of mixed metadata annotations and musical information in RDF for various MIR tasks. We leverage existing named entity recognition algorithms on the metadata side, and MIDI similarity algorithms on the music notation side (Section 5).

The remainder of the paper is organised as follows. In Section 2 we survey related work on integration and interlinking of musical notation and metadata, and MIDI similarity measures. In Section 3 we describe the MIDI Linked Data Cloud and two important extensions based on MIDI similarity and named entity recognition. In Section 4 we describe the Semantic Web MIDI Tape interface, and in Section 5 we provide a preliminary evaluation based on two use cases. Finally, in Section 6 we discuss our findings and present our conclusions.

2 RELATED WORK

2.1 Integration and interlinking

Up until now, a large number of music-related datasets have been published as Linked Open Data, where there is a strong emphasis on making music metadata explicit. Linked Data can be applied to describe cataloguing metadata, as exemplified by LinkedBrainz [7] and DoReMus [14]. Emerging fields such as semantic audio combine (audio) analysis techniques and Semantic Web technologies in order to associate provenance metadata with content-based analyses [1, 2].

Although music-specific datasets exist, and descriptive metadata is available to link contents to context, there is a lack of methods for the analysis and the integration of digitised symbolic notation [4]. The chord symbol service [6] provides RDF descriptions from compact chord labels, but does not include any information on the pieces or scores that can be related to such chords. In the Répertoire International des Sources Musicales (RISM) [10] portal, users can search scores by entering a melody—but the search is restricted to monophonic incipits, i.e., beginnings, of the scores. The Music Score Portal [9] addresses music score discovery and recommendation by exploiting links to the Linked Open Data cloud. However, links reference only authors and contributors of the scores, and users cannot contribute to enrich the knowledge base with new metadata. The Music Encoding and Linked Data (MELD) framework [30] applies Linked Data to express user-generated annotations on the musical structure.

2.2 MIDI similarity measures

Because of the enormous increase of music in digital form over the past decades, the computational modelling of music similarity has become an increasingly important research topic within the field of MIR. Recently, modelling music similarity has been called a “crucial need” [29] for researchers, librarians and archivists, industry, and consumers. Music similarity plays a large role in MIR tasks as divergent as content-based querying, music classification, music recommendation, or digital rights management and plagiarism detection [15, 29]. The similarity modelling task is different in the audio domain, which focuses on recorded sound, and where the input query is an audio signal [12, 15, 27], than in the symbolic domain, which deals with scores, encodings, and texts, and where the input query is some textual encoding (including MIDI) of the music [8, 23]. For this paper, we have restricted ourselves to the use of models of melodic similarity [28]. With respect to such models, three approaches have been proposed [19]: those based on the computation of index terms, those based on sequence matching techniques, and those based on geometric methods, which can cope with polyphonic scores. Examples of the latter are the algorithms that are part of MelodyShape, a Java library and tool for modelling melodic similarity [24, 26]. One of these algorithms, ShapeM, has consistently obtained the best results [25] in the last
The MIDI Linked Data Cloud is generated using midi2rdf [16]. This tool reads MIDI events from a MIDI file, and generates an equivalent representation in RDF by mapping the events onto the lightweight MIDI ontology shown in Figure 1. The top MIDI container is midi:Piece, which contains all MIDI data organised in midi:Tracks, each containing a number of midi:Events. A midi:Event is an abstract class around all possible musical events in MIDI, for example those that dictate to start playing a note (midi:NoteOnEvent), to stop playing it (midi:NoteOffEvent), or to change the instrument (midi:ProgramChangeEvent). Specific events have their own attributes (e.g., a midi:NoteOnEvent has a pitch and a velocity, i.e., loudness), but all events have a midi:tick, fixing them temporarily within the track. Instances of midi:Track are linked to the original file they were derived from (an instance of midi:MIDIFile) through prov:wasDerivedFrom. To enable interoperability and reuse with other datasets, as well as future extensions, we link the class midi:Track of the Music Ontology [22] to the class midi:MIDIFile through the property rdfs:seeAlso. An excerpt of a MIDI file, in Turtle format, is shown in Listing 1. IRIs of midi:Piece instances have the form midi-r:piece/[hash]/, where [hash] is the unique MD5 hash of the original MIDI file. Instances of midi:Track and midi:Event have IRIs that have the form midi-r:r:piece/[hash]/[track][<tid>] and midi-r:r:piece/[hash]/[track][<tid>]/[event][< eid>], where <tid> and <eid> are their respective IDs.

Aside from the mapping of MIDI datasets onto RDF, the MIDI Linked Data Cloud contains three additional sets of MIDI resources (see Table 1) that provide a rich description of MIDI notes (pitches), programs (instruments), and chords (simultaneous notes)—all of which in MIDI are expressed simply as integers. MIDI Linked Data notes link to their type (midi:Note), label (e.g., C, octave (e.g., 4)), and their original MIDI pitch value (e.g., 60). MIDI Linked Data programs link to their type (midi:Program), label (e.g., Acoustic Grand Piano), and their relevant instrument resource in DBpedia (e.g., http://dbpedia.org/resource/Acoustic_Grand_Piano). The links to corresponding DBpedia instruments have been added manually by an expert. All tracks link to resources in midi-note and midi-prog. IRIs in the midi-chord namespace are linked to instances of the midi:Chord class. The chord resources (see Table 1) describe a comprehensive set of chords, each of them with a label, quality, and is therefore used for this paper (see Section 3.1 and Section 5.2.2).

3 THE MIDI LINKED DATA CLOUD

The MIDI Linked Data Cloud [17] is a linked dataset of 308,443 MIDI files gathered from the Web and converted into 10,215,577,355 RDF triples. In what follows, we provide a summary of the dataset, and we describe two important additions to it.

The MIDI Linked Data Cloud is published at http://purl.org/midi-ld, and provides access to the community, documentation, source code, and dataset. All relevant dataset links and namespaces are shown in Table 1. A GitHub organisation hosts all project repositories, including documentation and tutorials, source MIDI collections, and the dataset generation source code. The MIDI Linked Data Cloud dataset results from applying this source code to the source MIDI collections, and adding the resources described in this section, as well as the extensions described in Section 3.1. All MIDI Linked Data is accessible as a full dump download, both through a SPARQL endpoint and through an API (see Section 4.2).

The MIDI Linked Data Cloud is generated using midi2rdf [16]. This tool reads MIDI events from a MIDI file, and generates an equivalent representation in RDF by mapping the events onto the lightweight MIDI ontology shown in Figure 1. The top MIDI container is midi:Piece, which contains all MIDI data organised in midi:Tracks, each containing a number of midi:Events. A midi:Event is an abstract class around all possible musical events in MIDI, for example those that dictate to start playing a note (midi:NoteOnEvent), to stop playing it (midi:NoteOffEvent), or to change the instrument (midi:ProgramChangeEvent). Specific events have their own attributes (e.g., a midi:NoteOnEvent has a pitch and a velocity, i.e., loudness), but all events have a midi:tick, fixing them temporarily within the track. Instances of midi:Track are linked to the original file they were derived from (an instance of midi:MIDIFile) through prov:wasDerivedFrom. To enable interoperability and reuse with other datasets, as well as future extensions, we link the class midi:Track of the Music Ontology [22] to the class midi:MIDIFile through the property rdfs:seeAlso. An excerpt of a MIDI file, in Turtle format, is shown in Listing 1. IRIs of midi:Piece instances have the form midi-r:piece/[hash]/, where [hash] is the unique MD5 hash of the original MIDI file. Instances of midi:Track and midi:Event have IRIs that have the form midi-r:r:piece/[hash]/[track][<tid>] and midi-r:r:piece/[hash]/[track][<tid>]/[event][<eid>], where <tid> and <eid> are their respective IDs.

Aside from the mapping of MIDI datasets onto RDF, the MIDI Linked Data Cloud contains three additional sets of MIDI resources (see Table 1) that provide a rich description of MIDI notes (pitches), programs (instruments), and chords (simultaneous notes)—all of which in MIDI are expressed simply as integers. MIDI Linked Data notes link to their type (midi:Note), label (e.g., C, octave (e.g., 4)), and their original MIDI pitch value (e.g., 60). MIDI Linked Data programs link to their type (midi:Program), label (e.g., Acoustic Grand Piano), and their relevant instrument resource in DBpedia (e.g., http://dbpedia.org/resource/Acoustic_Grand_Piano). The links to corresponding DBpedia instruments have been added manually by an expert. All tracks link to resources in midi-note and midi-prog. IRIs in the midi-chord namespace are linked to instances of the midi:Chord class. The chord resources (see Table 1) describe a comprehensive set of chords, each of them with a label, quality.
the number of pitch classes the chord contains, and one or more intervals, measured in semitones from the chord’s root note.

The resulting Linked Data is enriched with additional features that are not contained in the original MIDI files: provenance, integrated lyrics, and key-scale-metric information. To generate provenance, the extracted midi:Pieces are linked to the files they were generated from, the conversion activity that used them, and the agent (midi2rdf) associated with such activity. 8,391 MIDI files contain lyrics split into individual syllables, to be used mainly in karaoke software. Using the midi:lyrics property, these syllables are joined into an integrated literal as to facilitate lyrics-based search. Finally, the music analysis library music21\footnote{http://web.mit.edu/music21/} is used to further enrich the data: a piece’s key is either extracted directly from the MIDI file, or, if this information is not provided, detected automatically in order to obtain MIDI files containing a Homer-Schumacker algorithm\footnote{https://github.com/reinierdevalk/MIDI-LD/}; every note event is represented as the scale degree in that key; and for every note event the metric weight (i.e., position in the bar) is extracted, or, if no metric information is provided, detected (see Listing 1).

3.1 Dataset additions

In order to improve its quality and usability, in this work we extend the MIDI Linked Data Cloud with two additional subsets of data: a MIDI similarity subset and a named entity recognition over MIDI metadata subset.

3.1.1 MIDI similarity. We use the ShapeH melodic similarity algorithm that is part of the MelodyShape toolbox (see Section 2) to search for MIDI files that are similar to a query. This algorithm relies on a geometric model that encodes melodies as curves in the pitch-time space, and then computes the similarity between two melodies using a sequence alignment algorithm. The similarity of the melodies is determined by the similarity of the shape of the curves. The algorithm, which takes as input a query MIDI file and compares it with a corpus of MIDI files, can cope with polyphony. This means that it can process multi-track MIDI files, both at the query side and at the corpus side—but all individual tracks of these files need to be monophonc, that is, they, cannot contain note overlap. The large majority of the files in the MIDI Linked Data Cloud, however, does not meet this criterion: piano or drum tracks, for example, are almost without exception non-monophonic. Furthermore, numerous tracks are unintentionally non-monophonic, most likely due to sloppy data entering (e.g., because of keys of a MIDI keyboard having been released too late) or bad quantisation: in such cases, the offset time of the left note of a pair of adjacent notes is (slightly) larger than the onset time of the right note. In order for the algorithm to be able to process a file, both in the case of intentional and unintentional non-monophony, preprocessing is necessary. For this paper, we restricted ourselves to using only monophonic queries; see Section 5.2.2.

Thus, in files containing only monophonic tracks, we preprocess the data by means of a script that uses pretty_midi, a Python module for creating, manipulating and analysing MIDI files\footnote{https://github.com/marilenadaquino/midi-ld-similarity/}. Our script takes a MIDI file as input, and, for each track in the file, traverses all the notes in this track. In this script, every note event is represented as the metric weight in that key; and for every note event the metric weight (i.e., position in the bar) is extracted, or, if no metric information is provided, detected (see Listing 1).

pretty_midi, a track can be represented as a list of notes, ordered by onset time. The script checks for each note whether it overlaps with a note with a higher list index. If this is the case, there are two scenarios: either the overlap is considered significant, in which case it is assumed that the note simultaneity is intended, i.e., that both notes are part of a chord, or the overlap is considered insignificant, in which case it is assumed that the simultaneity is not intended. Significance is determined by the amount of note overlap and can be parameterised: if the overlap is greater than 1/n the duration of the left note, it is considered significant. We found a value of $n = 2$ to yield good results. In the case of significant note overlap, then, the track is simply removed from the MIDI file; in the case of insignificant note overlap, quantisation is applied by setting the left note’s offset to the right note’s onset.\footnote{Admittedly, this approach is somewhat crude: even if a track contains only one chord, it is removed. As a consequence, some files have all their tracks removed (see also Section 5.2.2). A more sophisticated approach is left for future work.}

MelodyShape can be run as a command line tool. With the following command, the 2015 version of the ShapeH algorithm is used to search the MIDI files in the data/ directory for the ten files (the number of files retrieved is controlled by the `-k` option) melodically most similar to the query query.midi:\footnote{See https://github.com/julian-urbano/MelodyShape/releases for a detailed user manual describing the usage of the command line tool.}

\begin{verbatim} $ java -jar melodyshape-1.4.jar -q query.mid -c data/ -W a rP i g s( 3 ) . m i d i -Y -a 2015-shapeh -k 10
\end{verbatim}

When executed, this command returns ten file names, each of them followed by the similarity score assigned by the algorithm to the file by that name (a concrete example can be seen in Table 3 below).

The output of the matching process is passed to a script that transforms this information into RDF statements.\footnote{See https://github.com/marilenadaquino/midi-ld-similarity/} In particular, MIDI files are identified as individuals of the class midi:Piece (see also Section 3), and files found to be similar are linked through skos:closeMatch. This statement is reified and identified by a hash derived from the URIs of the two MIDI files. The reified statement is further annotated with the midi:MelodyShapeScore property, which records the value of the similarity score as a xsd:float value. At the moment, only relations between MIDI files whose similarity score is greater than 0.6 are converted into RDF.

3.1.2 Named entity recognition. MIDI files included in the data set come from various collections on the Web. These files contain very limited contextual information. Nevertheless, the file name can include valuable information, as can be seen in the examples in Listing 2. We chose to exploit this information by relying on a named entity recognition tool, DBpedia Spotlight \cite{3}. Our approach takes the file names from the MIDI Linked Data Cloud, removes any non-alphanumeric characters (such as directory separators),
and considers the remaining words as a string to be annotated with DBpedia entities. The returned entities are then associated with the Linked Data URI of the MIDI piece using the `dc:subject` predicate. The generated RDF can be accessed at http://virtuoso-midi.amp.ops.labs.vu.nl/sparql/ under the named graph http://purl.org/midi-ld/spotlight/, and contains 1,894,282 new triples, of which 856,623 are `dc:subject` links from 197,126 unique MIDI pieces (63.93% of the total) to 25,667 different DBpedia entities. Table 2 shows the top 15 entity types identified.

<table>
<thead>
<tr>
<th>#Matches</th>
<th>DBpedia entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>10493</td>
<td><a href="http://dbpedia.org/resource/MusicalArtist">http://dbpedia.org/resource/MusicalArtist</a></td>
</tr>
<tr>
<td>10651</td>
<td><a href="http://dbpedia.org/resource/Artist">http://dbpedia.org/resource/Artist</a></td>
</tr>
<tr>
<td>15620</td>
<td><a href="http://dbpedia.org/resource/MusicalWork">http://dbpedia.org/resource/MusicalWork</a></td>
</tr>
<tr>
<td>15898</td>
<td><a href="http://xmlns.com/foaf/0.1/Person">http://xmlns.com/foaf/0.1/Person</a></td>
</tr>
<tr>
<td>24222</td>
<td><a href="http://schema.org/CreativeWork">http://schema.org/CreativeWork</a></td>
</tr>
<tr>
<td>24590</td>
<td><a href="http://schema.org/Organization">http://schema.org/Organization</a></td>
</tr>
<tr>
<td>37408</td>
<td><a href="http://dbpedia.org/resource/Band">http://dbpedia.org/resource/Band</a></td>
</tr>
<tr>
<td>40590</td>
<td><a href="http://schema.org/MusicGroup">http://schema.org/MusicGroup</a></td>
</tr>
<tr>
<td>55488</td>
<td><a href="http://dbpedia.org/resource/Agent">http://dbpedia.org/resource/Agent</a></td>
</tr>
<tr>
<td>56488</td>
<td><a href="http://dbpedia.org/resource/Agent">http://dbpedia.org/resource/Agent</a></td>
</tr>
<tr>
<td>105851</td>
<td><a href="http://dbpedia.org/resource/SheetMusic">http://dbpedia.org/resource/SheetMusic</a></td>
</tr>
<tr>
<td>10653</td>
<td><a href="http://dbpedia.org/resource/Artist">http://dbpedia.org/resource/Artist</a></td>
</tr>
<tr>
<td>10655</td>
<td><a href="http://dbpedia.org/resource/Person">http://dbpedia.org/resource/Person</a></td>
</tr>
</tbody>
</table>

Table 2: Top 15 entity types identified.

The process is entirely automatic, and although a large quantity of entities have been correctly identified, we are aware of inaccuracies in the data (for example, many files have been associated to the entity Life_Model_Decoy, or to Electronic_Dance_Music). Overall, the quality of data could be improved by filtering out entities that are not of specific safe types (Genre, Band, etc.), or by employing human supervision.

4 THE SEMANTIC WEB MIDI TAPE

The Semantic Web MIDI Tape\(^4\) is a set of tools and associated API that offer a read/write interface to the MIDI Linked Data Cloud, allowing users to play their MIDI instruments and stream their performance in native RDF form, record their performance in the Linked Open Data cloud, and then retrieve this recording.\(^5\) Concretely, with the Semantic Web MIDI Tape, users can:

1. broadcast a performance as a stream of RDF triples using a MIDI instrument;
2. record a performance as a MIDI Linked Data RDF graph, add associated metadata to this performance, and add metadata and curate annotations of existing MIDI Linked Data entities;
3. integrate a MIDI Linked Data RDF graph into the existing MIDI Linked Data cloud dataset;
4. retrieve the RDF graph of a performance;
5. play a retrieved RDF graph of a performance through any standard MIDI synthesizer.

Figure 2 shows how these activities fit in the architecture of the system. (1) and (2) are provided by the Semantic Web MIDI Tape tools (Section 4.1), (3) and (4) are provided as small clients that interact with the MIDI Linked Data API (Section 4.2), and (5) is provided by themidi2rdf suite of converters and algorithms\(^16\)—more concretely, rdf2midi, which converts Linked Data representations of MIDI data back to synthesizer-ready MIDI files.

4.1 MIDI Tape tools

We provide a set of open source tools to add and retrieve MIDI Linked Data and metadata to the MIDI Linked Data Cloud. These are intended to cover the workflow shown in Figure 2.\(^17\) Although there definitely is a role for software agents to use these tools, especially the API, this goes beyond the scope of this paper. The set consists of the following tools:

- `swmiditp-stream`. Produces a stream of RDF triples that represent MIDI data as it is played by the user through a MIDI input device (physical or virtual). When the user finishes their MIDI RDF performance, they can choose to attach relevant metadata to it, and serialise the corresponding RDF graph (Figure 2, steps (1) and (2)). The midi2rdf package is used to map MIDI events to a lightweight MIDI ontology.

\(^4\)See https://github.com/midi-ld/semweb-midi-tape/ for source code, install instructions, and examples.

\(^5\)https://github.com/enridaga/midi-ld-tags/

\(^6\)https://github.com/midi-ld/semweb-midi-tape/
The MIDI Linked Data API is the default entry point to access any
19
18
20
In order to enable their functioning,
18
21

22
22

SAAM 2018, October 2018, Monterey, CA, USA

• swmiditp-upload. Uploads MIDI RDF N-triples files to the
MIDI Linked Data Cloud triplestore (Figure 2, step (3)). The
user can browse the Linked Data representation of the up-
loaded MIDI performance and its associated metadata.18
• swmiditp-download. Downloads MIDI RDF N-triples that
represent a MIDI performance identified by its URI (Figure 2,
step (4)).
• rdf2midi. We use the rdf2midi algorithm to convert the
downloaded MIDI Linked Data into a standard MIDI file that
can be played by most synthesizers (Figure 2, step (5)).

The metadata collection in step (2) consists of asking the user
for a number of URIs that identify entities that are relevant to the
generated MIDI RDF performance. Concretely, we gather URIs to
implement relevant subsets of the Music Ontology. The MIDI per-
formance, identified by its URI, is an instance of mo:Performance.
More precisely, this performance is a mo:performance_of some
mo:Composition, which in turn is an individual realisation, or ex-
pression,1 of some mo:MusicalWork—which may or may not be
original. Importantly, such a musical work has a mo:composer of
type mo:MusicArtist that is also provided by the user; if the user
entered a non-original, pre-existing piece (in popular music cul-
ture referred to as a cover), this would be this piece’s original cre-
ator. Finally, we add a statement that the MIDI performance has
a mo:performer that is of type mo:MusicArtist, i.e., a URI that
identifies the user as such.

4.2 MIDI Linked Data API
The MIDI Linked Data API is the default entry point to access any
MIDI resource in the MIDI Linked Data Cloud. It is implemented
as a grlc [18] Linked Data API, and powered by publicly shared,
community maintained SPARQL queries.20 The full documentation
and call names of the MIDI Linked Data API are available at http://
grlc.io/api/midi-ld/queries/.

For the Semantic Web MIDI Tape, this API has been extended
with the two following routes:

• POST :insert_pattern?g=uri2&data=lit1. Inserts the MIDI
RDF graph contained in lit1 under the named graph uri2.
This operation is implemented with a SPARQL INSERT DATA
query.
• GET :pattern_graph?pattern=uri1. Returns the complete
graph of all RDF statements associated with the MIDI identi-
fied by the URI uri1. This operation is implemented with a
SPARQL CONSTRUCT query.

These operations are used by the tools swmiditp-upload and
swmiditp-download in steps (3) and (4), respectively (see Sec-
tion 4.1 and Figure 2). The SPARQL endpoint against which they
are executed can be customised in the underlying SPARQL queries.21
In order to enable their functioning, grlc has been extended with
support for CONSTRUCT and INSERT queries.

5 USE CASES
We perform a use case-based evaluation of the Semantic Web MIDI
Tape. The goal of this evaluation is to show that the joint notation
and metadata capabilities of the Semantic Web MIDI Tape enable a
rich interaction between users and the Web in at least two scenarios:
a data cleaning, annotation, and enrichment scenario; and a MIR
scenario. The first scenario shows how to contribute to the MIDI
LD Cloud by means of the Semantic Web MIDI tape and discover
similar music contents and information, in a Shazam22 fashion.
The second scenario addresses typical scholars’ needs and more
sophisticated musicians’ needs. For example, a scholar can retrieve
and group performances by topic (e.g., all the performances related
to Liverpool) and see the distribution of the latter; musicians and
DJs can group performances by both topic and music characteristics
(e.g., songs about Romeo and Juliet having the same tempo) for
reusing music samples or remixing purposes.

In this preliminary evaluation we do not yet tackle the problem
of scalability derived from similarity matching, and we apply the
aforementioned method only to a subset of the MIDI Linked Data
Cloud. Hence, the retrieval of performances and related metadata
is currently limited to that subset.

5.1 Use Case 1: contributing
This use case showcases the basic functionality of the Semantic Web
MIDI Tape by enabling the user to add new MIDI RDF performance
data, accompanied by rich metadata descriptions, to the cloud. In
this use case, data submitted by the user has two components: a
notation component, which describes musical events of the user’s
performance as MIDI Linked Data triples; and a metadata compo-
nent, which annotates the notation component with relevant links
to external music metadata datasets (see Section 4.1).

The use case starts with a user ready to play a performance
on a MIDI instrument. The user executes swmiditp-stream (see
Section 4.1), is prompted a list of detected input MIDI devices, and
chooses the one to be played:

$ python swmiditp-stream.py > myperformance.nt

Detected MIDI input devices:
[8] Midi Through:Midi Through Port=0 14:0
[1] VMini:VMini MIDI 1 20:0

Interaction menus are shown via stderr, making output redi-
 rects become valid RDF N-Triples files. At this point, the user plays
the performance, and the stream of triples is stored in a file (or,
alternatively, shown on screen if no output redirect is used). To
end the performance, the user presses Ctrl+C.

The system subsequently provides http://purl.org/midi-ld/pattern/
604115f5-45ad-4135-be35-0281193103ed as the URI to the generated
MIDI RDF mo:Performance that identifies it in the http://purl.org/
midi-ld/dataspace. The system then prompts for a set of additional
URIs, pointing to essential metadata, to describe the performance:

• the URIs of the musical work and composition performed
(e.g., https://musicbrainz.org/work/cac6d507-46ed-3ed7-80d5-e4ac31719221 or http://dbpedia.org/resource/Hey_Jude);

18See, e.g., http://purl.org/midi-ld/id-pattern/40e4bldeo-31e1-4e50-a2e2-2446872535531.
19http://musicontology.com/specification#term-MusicalWork
20https://github.com/midi-ld/queries/
21See example at https://github.com/midi-ld/queries/.
22https://shazam.com/
MIDI metadata exploiting the content of remote SPARQL endpoints. For example, we can search for the MIDI files related to entities whose hometown is Liverpool (Listing 5). Finally, we can integrate musical content and metadata in the same query. For example, the SPARQL query in Listing 5 looks up all MIDI files that reference the topic Romeo and Juliet in common time (i.e., a \( \frac{4}{4} \) time signature), effectively enabling querying that combines notation data and metadata. The query retrieves two results: the soundtrack from a popular movie, and the Dire Straits song.\(^{25}\)

5.2.2 Querying by playing. In the second proposed type of querying, the objective is to retrieve MIDI files that are similar to a given MIDI query—either a file created ad hoc using a MIDI device, or a pre-existing file. No context information is provided by the user. The query returns (1) all MIDI files satisfying a certain similarity threshold, and (2), if available, for each file the contextual information it is annotated with, and, by extension, the files related—linked—to it. This type of querying again demonstrates the benefits of representing symbolic music formats as Linked Data.

Serving as a proof-of-concept, we set up a simple experiment in which we used the ShapeIt melodic similarity algorithm in the way described in Section 3.1 to query a small subset of the MIDI Linked Data Cloud. The subset was randomly selected and originally contained 1531 MIDI renditions of rock songs by 83 different artists. From this initial set, 68 MIDI files had to be omitted because they were found to be corrupt, i.e., could not be parsed by pretty_midi, and 152 further files were removed during the data preprocessing process (see Section 3.1) as each of these files contains, for one reason or another, significant note overlap in all of its individual tracks. This pruning resulted in a test set of 1311 MIDI files, each of them containing exclusively monophonic tracks. Using transcriptions of five randomly selected Beatles songs, all of which we know to be contained in the test set, we then created five query

\( \text{Listing 3: SPARQL query for analysing the distribution of genres in the dataset.} \)

```sql
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX mid: <http://purl.org/midi-ld/midi#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX prov: <http://www.w3.org/ns/prov#>

GROUP BY ?genre ORDER BY DESC(?c)
```

\( \text{Listing 4: SPARQL query to search for content related to entities whose hometown is Liverpool.} \)

```sql
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?pattern ?subject WHERE {
  ?pattern midi:hasTrack ?track .
  ?track midi:hasEvent ?event .
  ?subject dbo:hometown dbr:Liverpool .
}
```

\( \text{Listing 5: SPARQL query for MIDI files that reference Romeo and Juliet in common time.} \)

\( \text{\footnotesize example, we can search for the MIDI files related to entities whose hometown is Liverpool (Listing 5).} \)

\( \text{\footnotesize Finally, we can integrate musical content and metadata in the same query. For example, the SPARQL query in Listing 5 looks up all MIDI files that reference the topic Romeo and Juliet in common time (i.e., a \( \frac{4}{4} \) time signature), effectively enabling querying that combines notation data and metadata. The query retrieves two results: the soundtrack from a popular movie, and the Dire Straits song.} \)

\( \text{\footnotesize Serves as a proof-of-concept, we set up a simple experiment in which we used the ShapeIt melodic similarity algorithm in the way described in Section 3.1 to query a small subset of the MIDI Linked Data Cloud. The subset was randomly selected and originally contained 1531 MIDI renditions of rock songs by 83 different artists. From this initial set, 68 MIDI files had to be omitted because they were found to be corrupt, i.e., could not be parsed by pretty_midi, and 152 further files were removed during the data preprocessing process (see Section 3.1) as each of these files contains, for one reason or another, significant note overlap in all of its individual tracks. This pruning resulted in a test set of 1311 MIDI files, each of them containing exclusively monophonic tracks. Using transcriptions of five randomly selected Beatles songs, all of which we know to be contained in the test set, we then created five query} \)

\( \text{\footnotesize example, we can search for the MIDI files related to entities whose hometown is Liverpool (Listing 5).} \)

\( \text{\footnotesize Finally, we can integrate musical content and metadata in the same query. For example, the SPARQL query in Listing 5 looks up all MIDI files that reference the topic Romeo and Juliet in common time (i.e., a \( \frac{4}{4} \) time signature), effectively enabling querying that combines notation data and metadata. The query retrieves two results: the soundtrack from a popular movie, and the Dire Straits song.} \)

\( \text{\footnotesize Serves as a proof-of-concept, we set up a simple experiment in which we used the ShapeIt melodic similarity algorithm in the way described in Section 3.1 to query a small subset of the MIDI Linked Data Cloud. The subset was randomly selected and originally contained 1531 MIDI renditions of rock songs by 83 different artists. From this initial set, 68 MIDI files had to be omitted because they were found to be corrupt, i.e., could not be parsed by pretty_midi, and 152 further files were removed during the data preprocessing process (see Section 3.1) as each of these files contains, for one reason or another, significant note overlap in all of its individual tracks. This pruning resulted in a test set of 1311 MIDI files, each of them containing exclusively monophonic tracks. Using transcriptions of five randomly selected Beatles songs, all of which we know to be contained in the test set, we then created five query} \)
files, each consisting of the first four bars of the vocal melody of a song (note that the query can be any melodic line in a piece; from a user’s perspective, however, the vocal melody seems a logical choice). To account for tonal variability during data entry, each query file was transposed two, four and six semitones up as well as down—resulting in a total of 35 query files. Table 3, shows, for each query, the three files found to be most similar by the algorithm, as well as the similarity score per retrieved file. Note that only the untransposed queries are listed; transposition was found to have no effect whatsoever, always yielding the exact same results as when using the untransposed file.

<table>
<thead>
<tr>
<th>Query</th>
<th>Files retrieved</th>
<th>Similarity scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>here_courses_the_sum.mid (1)</td>
<td>Here_Courses_The_Sum.mid</td>
<td>0.98087875</td>
</tr>
<tr>
<td>Dear_Tell_Me_2.mid</td>
<td>0.22971322</td>
<td></td>
</tr>
<tr>
<td>taking_its_all_toハード.mid</td>
<td>0.15244184</td>
<td></td>
</tr>
<tr>
<td>key_idle.mid (1)</td>
<td>Hey_Hide.mid</td>
<td>0.51530613</td>
</tr>
<tr>
<td>Its_Only_Love.mid</td>
<td>0.12579889</td>
<td></td>
</tr>
<tr>
<td>Little_Horn.mid</td>
<td>0.12579889</td>
<td></td>
</tr>
<tr>
<td>last_it_heits.mid (2)</td>
<td>Let_It_Be.mid</td>
<td>0.92157645</td>
</tr>
<tr>
<td>Let_It_Bean.mid</td>
<td>0.76319985</td>
<td></td>
</tr>
<tr>
<td>Crash_Course_In_Brain_Surgery.mid</td>
<td>0.73958380</td>
<td></td>
</tr>
<tr>
<td>norwegian_wood.mid (1)</td>
<td>All_Its_Love_2.mid</td>
<td>0.90710577</td>
</tr>
<tr>
<td>New_Languages.mid</td>
<td>0.90710577</td>
<td></td>
</tr>
<tr>
<td>Norwegian_Wood.mid</td>
<td>0.90710577</td>
<td></td>
</tr>
<tr>
<td>yesterday.mid (1)</td>
<td>Yesterday.mid</td>
<td>0.90710577</td>
</tr>
<tr>
<td>Time_is_Time.mid</td>
<td>0.15554734</td>
<td></td>
</tr>
<tr>
<td>Beatle.mid</td>
<td>0.06137392</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Querying by playing: MIDI queries, top three files retrieved, and corresponding similarity scores. Numbers in parentheses indicate a query’s number of target file(s) (printed in bold) in the test set.

As the table shows, for queries 1-3 and 5, the target file (or files) receive the highest (or two highest) similarity scores. Only in the case of query 4, the target file receives the third highest similarity score (which, at 0.96, is still quite high). A possible reason for this mismatch is the fact that the preprocessing at times results in very scarce and fragmented MIDI files (in the case of query 4, for example, the file that receives the highest score contains no more than three notes)—which may throw the similarity algorithm off. The triples generated from the matching process (see Section 3.1.1) are sent to the MIDI Linked Data Cloud. Similar MIDI files and related similarity scores can be retrieved by querying for skos:closeMatch values. Moreover, similar MIDI files’ related metadata generated by the named entity recognition tool (see Section 3.1.2) can be retrieved as well by looking for (optional) dc:subject values.

The experiment shows that querying by playing by promising results—but this type of querying will have to be tested more systematically in order to properly assess its accuracy and usability. One of the issues to be addressed is the determination of an appropriate threshold value, below which similarity is deemed to end, for the similarity score. For this paper, this value was experimentally set to 0.6 (see Section 3.1.1).

6 DISCUSSION AND CONCLUSION

In this paper, we address difficulties at generating missing links from a large linked dataset representing symbolic music notation, the MIDI Linked Data Cloud, to related entities in other linked music metadata datasets. Finding these links is hard due to three fundamental issues: (1) the lack of explicit statements about MIDI music similarity; (2) the absence of named entities referred in MIDI metadata; and (3) the difficulty for users to contribute user-generated content to the platform, as well as to query it. To address these issues, we propose, first, two extensions to the MIDI Linked Data Cloud—using MIDI similarity measures, and using state-of-the-art named entity recognition algorithms—, and second, the Semantic Web MIDI Tape, an interface for streaming, writing and reading MIDI content and related metadata in the MIDI Linked Data Cloud in native RDF. To evaluate the system, we describe two use cases in which the proposed solutions are applied: (1) to contribute performance data and metadata generated through a user’s MIDI input device to the MIDI Linked Data Cloud, which we use to enrich the dataset itself; and (2) to query the dataset based on symbolic notation and metadata. In these use cases we gather evidence that overcoming the identified difficulties on usability, linkage, and contributed content is to a large extent possible. Rather than solving the MIDI Linked Data-metadata interlinking problem, we propose a modular infrastructure to address it, focusing on the user. By representing MIDI information as Linked Data, user annotations can point to globally and uniquely identifiable MIDI events, making a combined retrieval with contextual metadata trivial in SPARQL.

Many aspects remain open for improvement in future work. First, the automatic approach for named entity recognition is error-prone, and should be combined with human supervision. We currently generate links to DBpedia entities using dc:subject. More sophisticated approaches might include heuristics trying to identify the roles of such entities, for example exploiting their types (e.g., a MusicGroup must be the author). The combination of musical data and its semantics in the same knowledge base opens novel possibilities for researching the relation between musical content and associated context at scale. We plan to leverage event metadata events within MIDI files to further enrich MIDI named entity recognition. Second, we plan to address the scalability of generating MIDI similarity and named entity recognition links. Third, we plan to use platform-independent Web-enabled clients, adding to the described command line tools, and investigating issues around distributed content generation and user disagreement in metadata.

In the longer run, we aim to set a precedent for interacting and connecting with a variety of Linked Data, and eventually across music notation formats, such as MIDI, *Kern, MusicXML, and MEI. In the musicology domain there is a shared interest in relying on higher-level notations, yet there is currently no single standard for the encoding of musical data. This project states that all areas of musicology should collaboratively aim to achieve cross-format interactions, enabling an analysis across symbolic and audio data. We envisage this as a Big Musicology project, a concept derived from that of Big Science [11], in which the ethos of collaboration is embraced. Hence, we aim for a real, Web-enabled linkage of music notation across formats, offering the user to apply them as appropriate, and to find almost their least common denominator. The aims are to overcome interoperability issues among formats, to avoid loss of information in data conversion, and to enable the user to discover new and unexpected information.
REFERENCES


