# 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, AcousticBrainz, 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:**
Albert Meroño-Peñuela, Reinier de Valk, Enrico Daga, Marilena Daquino, and Anna Kent-Muller. 2018. The Semantic Web MIDI Tape: An Interface for Interlinking MIDI and Context Metadata. In *Proceedings of Workshop on Semantic Applications for Audio and Music (SAAM 2018).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

*SAAM 2018, October 2018, Monterey, CA, USA*
© 2018 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

## 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;[1] 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].[3] 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

[1] http://essen.themefinder.org/
[2] https://tinyurl.com/y7t6l6yt/
[3] See the catalogue *musoW: Musical data on the web* for a comprehensive list of music-related resources available on the web: http://musow.kmi.open.ac.uk/.

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 ShapeH melodic 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.[4] The Music Score Portal [9] addresses music score discovery and recommendation by exploiting links to the Linked Open Data cloud.[5] 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,[6] a Java library and tool for modelling melodic similarity [24, 26]. One of these algorithms, ShapeH, has consistently obtained the best results [25] in the last

[4] https://opac.rism.info/metaopac/start.do?View=rism
[5] http://linkeddata.uni-muenster.de/musicportal/web/
[6] https://github.com/julian-urbano/MelodyShape/

| Resource | Link |
|---|---|
| MIDI Linked Data Cloud dataset | http://purl.org/midi-ld |
| Portal page | https://midi-ld.github.io/ |
| `midi2rdf`-as-a-Service | http://purl.org/midi-ld/midi2rdf |
| MIDI Vocabulary namespace | http://purl.org/midi-ld/midi#<prefix `midi`> |
| MIDI Resource namespace | http://purl.org/midi-ld/<prefix `midi-r`> |
| MIDI Notes namespace | http://purl.org/midi-ld/notes/<prefix `midi-note`> |
| MIDI Programs namespace | http://purl.org/midi-ld/programs/<prefix `midi-prog`> |
| MIDI Chords namespace | http://purl.org/midi-ld/chords/<prefix `midi-chord`> |
| MIDI Pieces namespace | http://purl.org/midi-ld/piece/<prefix `midi-p`> |
| GitHub organisation & code | https://github.com/midi-ld/ |
| Dataset generation code | https://github.com/midi-ld/midi2rdf/ |
| Documentation and tutorials | https://github.com/midi-ld/documentation/ |
| Soure MIDI collections | https://github.com/midi-ld/sources/ |
| Sample SPARQL queries | https://github.com/midi-ld/queries/ |
| VoID description | http://purl.org/midi-ld/void |
| Full dump downloads | http://midi-ld.amp.ops.labs.vu.nl/ |
| SPARQL endopint | http://virtuoso-midi.amp.ops.labs.vu.nl/sparql/ |
| API | http://grlc.io/api/midi-ld/queries/ |
| Figshare | https://figshare.com/articles/The_MIDI_Linked_Data_Cloud/4990415 |
| Zenodo | https://zenodo.org/record/579603#.WRluUXV97MU/ |
| Datahub | https://datahub.io/dataset/the-midi-linked-data-cloud/ |

**Table 1: Links to key resources of the MIDI Linked Data Cloud dataset.**

few iterations (2010-2015) of the Music Information Retrieval Evaluation eXchange (MIREX) Symbolic Melodic Similarity evaluation task,[7] 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 10,215,557,355 RDF triples. In what follows, we provide a summary of the dataset,[8] 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).

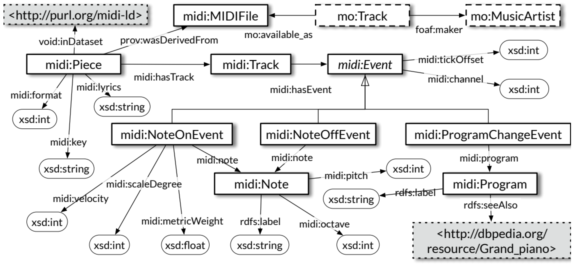

**Figure 1: Excerpt of the MIDI ontology: pieces, tracks, events, and their attributes.**

```
midi-p:cb87a5bb1a44fa72e10d519605a117c4 a midi:Piece ;
    midi:format 1 ;
    midi:key "E minor" ;
    midi:hasTrack midi-p:cb87a5b/track00,
        midi-p:cb87a5b/track01, ... .
midi-p:cb87a5b/track01 a midi:Track ;
    midi:hasEvent midi-p:cb87a5b/track01/event0000,
        midi-p:cb87a5b/track01/event0001, ... .
midi-p:cb87a5b/track01/event0006 a midi:NoteOnEvent ;
    midi:channel 9 ;
    midi:note midi-note:36 ;
    midi:scaleDegree 6 ;
    midi:tick 0 ;
    midi:velocity 115 ;
    midi:metricWeight 1.0 .
```

**Listing 1: Excerpt of Black Sabbath's *War Pigs* as MIDI Linked Data, with long hashes and track and event URIs shortened.**

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:Track`s, each containing a number of `midi:Event`s. 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 temporally 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 `mo:Track` of the Music Ontology [22] to the class `midi:MIDIFile` through the property `mo:available_as`. 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:piece/<hash>/track<tid>` and `midi-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/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,

[7] See http://www.music-ir.org/mirex/wiki/2016:Symbolic_Melodic_Similarity (task) and http://www.music-ir.org/mirex/wiki/2015:Symbolic_Melodic_Similarity_Results (latest results).

[8] A complete technical description and download links to the latest version can be found at https://midi-ld.github.io/.

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`[9] 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 using, e.g., the Krumhansl-Schmuckler algorithm [13]; 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 monophonic, 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 order to obtain MIDI files containing only monophonic tracks, we preprocessed the data by means of a script that uses `pretty_midi`, a Python module for creating, manipulating and analysing MIDI files [21].[10] Our script takes a MIDI file as input, and, for each track in the file, traverses all the notes in this track. In `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.[11]

```
mysongbook_midi/Hard,Heavy/Black_Sabbath/Black Sabbath\
- War Pigs (3).midi
mysongbook_midi/Pop,Rock/Beatles/Beatles (The) - Hey Jude (2).midi
mysongbook_midi/TV,Movie,Games/TV_And_Movie_Theme_Songs/Unknown\
(TV) - X-Files Theme.midi
```

**Listing 2: File names very often contain useful information about the context of the MIDI piece.**

`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.mid`:[12]

```
$ java -jar melodyshape-1.4.jar -q query.mid -c data/
-a 2015-shapeh -k 10
```

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.[13] 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 dataset 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 [3]. Our approach takes the file names from the MIDI Linked Data Cloud, removes any non-alphanumeric characters (such as directory separators),

[9] http://web.mit.edu/music21/

[10] https://github.com/reinierdevalk/MIDI-LD/

[11] 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.

[12] See https://github.com/julian-urbano/MelodyShape/releases/ for a detailed user manual describing the usage of the command line tool.

[13] https://github.com/marilenadaquino/midi-ld-similarity/

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.[14] 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 (61.93% of the total) to 25,667 different DBpedia entities. Table 2 shows the top 15 entity types identified.

| **#Matches** | **DBpedia entity** |
|---:|---|
| 124247 | http://dbpedia.org/resource/TopicalConcept<br>http://dbpedia.org/resource/MusicGenre<br>http://dbpedia.org/resource/Genre |
| 56488 | http://dbpedia.org/resource/Agent |
| 47901 | http://schema.org/MusicGroup |
| 40590 | http://schema.org/Organization<br>http://dbpedia.org/resource/Organisation |
| 37408 | http://dbpedia.org/resource/Band |
| 24222 | http://schema.org/CreativeWork<br>http://dbpedia.org/resource/Work |
| 15898 | http://xmlns.com/foaf/0.1/Person<br>http://schema.org/Person<br>http://dbpedia.org/resource/Person |
| 15620 | http://dbpedia.org/resource/MusicalWork |
| 10651 | http://dbpedia.org/resource/Artist |
| 10493 | http://dbpedia.org/resource/MusicalArtist |

**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.

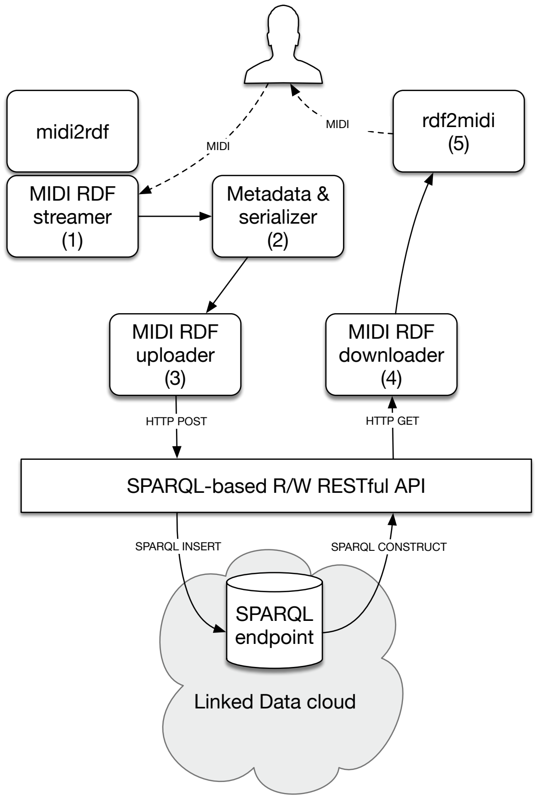

**Figure 2: Architecture of the Semantic Web MIDI Tape. Dotted lines depict data flow in native MIDI format; continuous lines represent RDF or SPARQL data transfer through HTTP or the system shell.**

## 4 THE SEMANTIC WEB MIDI TAPE

The *Semantic Web MIDI Tape*[15] 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.[16] 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 the `midi2rdf` 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;

[14] https://github.com/enridaga/midi-ld-tags/
[15] https://github.com/midi-ld/semweb-midi-tape/
[16] If no endpoint is specified, we assume it to be the MIDI Linked Data Cloud SPARQL endpoint.
[17] See https://github.com/midi-ld/semweb-midi-tape/ for source code , install instructions, and examples.

- `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 uploaded 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 performance, 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 expression,[19] 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 culture referred to as a *cover*), this would be this piece's original creator. 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 identified 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 Section 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 Shazam[22] 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* component, 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:
[0] Midi Through:Midi Through Port-0 14:0
[1] VMini:VMini MIDI 1 20:0
[2] VMini:VMini MIDI 2 20:1
1
```

Interaction menus are shown via `stderr`, making output redirects 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/eac0d507-46ed-3ed7-80d5-e4ac31719221/ or http://dbpedia.org/resource/Hey_Jude);

[18] See, e.g., http://purl.org/midi-ld/pattern/49cb8dce-31e1-4e50-a2e2-244872535531.

[19] http://musicontology.com/specification/#term-MusicalWork

[20] https://github.com/midi-ld/queries/

[21] See example at https://github.com/midi-ld/queries/.

[22] https://shazam.com/

```
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX mid: <http://purl.org/midi-ld/midi#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
SELECT (count(?pattern) as ?c) ?genre
WHERE {
  ?pattern dct:subject ?genre .
  ?genre a dbpedia:MusicGenre
} GROUP BY ?genre ORDER BY DESC(?c)
```

**Listing 3: SPARQL query for analysing the distribution of genres in the dataset.**

- the URI of the composer of the work performed (e.g., http://dbpedia.org/resource/The_Beatles);
- the URI that identifies the user as the artist that performed the work (e.g., http://example.org/foaf.rdf).

The user provides these URIs, and the system links them to the performance MIDI Linked Data (see Section 4.1). The PROV provenance model [5] is used to create a subgraph of provenance information containing activities, agents, and the start and end of creation timestamps. Finally, the user uploads the generated RDF graph to the cloud, optionally specifying in the first parameter a named graph:

```
$ ./swmiditp-upload.sh '<urn:graph:midi-ld>'
myperformance.nt
```

Depending on the graph size, within seconds after this, both performance notation and metadata become available, de-referenceable and browsable at the previously given MIDI performance URI.[23]

We follow a postprocessing strategy within our platform in which we employ the user-contributed data to add external links and improve the quality of the MIDI Linked Data Cloud. This strategy makes use of the MIDI similarity links `skos:closeMatch` (generated as described in Section 3.1) to propagate the user-contributed metadata to other MIDI files that are similar to the user-contributed MIDI performance.[24] In this way, we use both input metadata and symbolic music similarity to generate links to external music datasets, we increase the amount of context for a given musical work, and we improve the quality of the MIDI Linked Data Cloud.

If no similar MIDI files are found in the cloud, we assume that the user has made a novel contribution. In this case, we bootstrap the piece's metadata with the attached metadata, and store it for further retrieval or expansion.

## 5.2 Use Case 2: querying

To demonstrate the increase in usability of the MIDI Linked Data Cloud we consider two types of querying that were not possible before the dataset was extended with DBpedia links and similarity measures: *querying contextual information* and *querying by playing*.

*5.2.1 Querying contextual information.* Here, the objective is to retrieve musical content related to specific entities. For example, it is now possible to show the music genres most frequently represented in the dataset among the ones identified by DBpedia Spotlight (see query in Listing 3). Linking the MIDI data effectively integrates the dataset in the Linked Data Cloud, allowing to use potentially infinite 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 4).

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?pattern ?subject WHERE {
?pattern dct:subject ?subject
{{
SELECT ?subject
WHERE {
  SERVICE <http://dbpedia.org/sparql> {
        ?subject dbo:hometown dbr:Liverpool
    }}
}}}
```

**Listing 4: SPARQL query to search for content related to entities whose hometown is Liverpool.**

```
PREFIX midi: <http://purl.org/midi-ld/midi#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?pattern WHERE {
  ?pattern a midi:Pattern .
  ?pattern dc:subject dbr:Romeo_and_Juliet .
  ?pattern midi:hasTrack ?track .
  ?track midi:hasEvent ?event .
  ?event midi:numerator 4 .
  ?event midi:denominator 4 .
}
```

**Listing 5: SPARQL query for MIDI files that reference *Romeo and Juliet* in common time.**

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 `ShapeH` 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

[23] http://purl.org/midi-ld/pattern/604115f5-45ad-4135-be35-0281193103ed

[24] To avoid scalability issues we do not materialise this propagation, and only reuse a MIDI file's metadata by following MIDI similarity links to it.

[25] See http://purl.org/midi-ld/pattern/13fa17dc74232f7cb710a4d8d9f796b2 and http://purl.org/midi-ld/pattern/7a08a4b1efd5ff7afd6c1066b4a8dd94.

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).[26] 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.

| Query | Files retrieved | Similarity scores |
|---|---|---|
| here_comes_the_sun.mid (1) | **Here_Comes_The_Sun.mid** | 0.99005878 |
| | Dont_Tell_Me_2.mid | 0.22917332 |
| | Taking_It_All_To_Hard.mid | 0.15244147 |
| hey_jude.mid (1) | **Hey_Jude.mid** | 0.68150619 |
| | Its_Only_Love.mid | 0.12370080 |
| | Little_Horn.mid | 0.12370080 |
| let_it_be.mid (2) | **Let_It_Be_2.mid** | 0.94157451 |
| | **Let_It_Be.mid** | 0.76329338 |
| | Crash_Course_In_Brain_Surgery.mid | 0.57983380 |
| norwegian_wood.mid (1) | All_My_Lovin_2.mid | 0.98719078 |
| | New_Languages.mid | 0.98719078 |
| | **Norwegian_Wood.mid** | 0.95650927 |
| yesterday.mid (1) | **Yesterday.mid** | 0.94005626 |
| | Time_Is_Time.mid | 0.15554734 |
| | Beans.mid | 0.06157292 |

**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 yields 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 `dct: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 text 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 require access to (digital) musical data, and that 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.

[26] The scores, all for voice, guitar, and piano (and with Musicnotes.com IDs MN0104281, MN0053744, MN0053244, MN0053764, and MN0053784) were retrieved from https://www.musicnotes.com/sheetmusic/index/artists/Beatles-The/default.asp.

# REFERENCES

[1] Sean Bechhofer, Kevin Page, David M. Weigl, György Fazekas, and Thomas Wilmering. 2017. Linked Data publication of live music archives and analyses. In *The Semantic Web—ISWC 2017: 16th International Semantic Web Conference*, Claudia d'Amato et al. (Eds.). Vol. 2. Springer, Cham, 29–37.

[2] Tim Crawford, Ben Fields, David Lewis, and Kevin Page. 2014. Explorations in Linked Data practice for early music corpora. In *Proceedings of the IEEE/ACM Joint Conference on Digital Libraries, London, UK*. 309–312.

[3] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems, Graz, Austria*. 121–124.

[4] Marilena Daquino, Enrico Daga, Mathieu d'Aquin, Aldo Gangemi, Simon Holland, Robin Laney, Albert Mero no Peñuela, and Paul Mulholland. 2017. Characterizing the landscape of musical data on the Web: State of the art and challenges. In *Proceedings of the 2nd Workshop on Humanities in the Semantic Web, Vienna, Austria*.

[5] Paul Groth and Luc Moreau. 2013. *PROV-overview: An overview of the PROV family of documents*. Technical Report. World Wide Web Consortium.

[6] Christopher Harte, Mark Sandler, Samer Abdallah, and Emilia Gómez. 2005. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, London, UK*. 66–71.

[7] Kurt Jacobson, Simon Dixon, and Mark Sandler. 2010. LinkedBrainz: Providing the MusicBrainz next generation schema as Linked Data. In *Proc. of the 11th International Society for Music Information Retrieval Conference, Utrecht, Netherlands,* Late-Breaking/Demo session.

[8] Berit Janssen, Peter van Kranenburg, and Anja Volk. 2017. Finding occurrences of melodic segments in folk songs employing symbolic similarity measures. *Journal of New Music Research* 46, 2 (2017), 118–134.

[9] Jim Jones, Diego de Siqueira Braga, Kleber Tertuliano, and Tomi Kauppinen. 2017. MusicOWL: The music score ontology. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Leipzig, Germany*. 1222–1229.

[10] Klaus Keil. 2014. Répertoire International des Sources Musicales (RISM). *Fontes Artis Musicae* 61, 4 (2014), 373–376.

[11] Anna Kent-Muller. 2017. Big Musicology: A framework for transformation. In *Proceedings of the 4th International Digital Libraries for Musicology Workshop*.

[12] Peter Knees and Markus Schedl. 2016. *Music similarity and retrieval: An introduction to audio- and web-based strategies*. Springer, Berlin.

[13] Carol L. Krumhansl. 1990. *Cognitive foundations of musical pitch*. Oxford University Press, New York.

[14] Pasquale Lisena, Raphaël Troncy, Konstantin Todorov, and Manel Achichi. 2017. Modeling the complexity of music metadata in semantic graphs for exploration and discovery. In *Proceedings of the 4th International Digital Libraries for Musicology Workshop*. 17–24.

[15] Brian McFee, Luke Barrington, and Gert Lanckriet. 2012. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 8 (2012), 2207–2218.

[16] Albert Meroño-Peñuela and Rinke Hoekstra. 2016. The Song Remains the Same: Lossless conversion and streaming of MIDI to RDF and Back. In *The Semantic Web: ESWC 2016 satellite events*, Harald Sack et al. (Eds.). Springer, Cham, 194–199.

[17] Albert Meroño-Peńuela, Rinke Hoekstra, Aldo Gangemi, Peter Bloem, Reinier de Valk, Bas Stringer, Berit Janssen, Victor de Boer, Alo Allik, Stefan Schlobach, and Kevin Page. 2017. The MIDI Linked Data Cloud. In *The Semantic Web—ISWC 2017: 16th International Semantic Web Conference*, Claudia d'Amato et al. (Eds.). Vol. 2. Springer, Cham, 156–164.

[18] Albert Meroño-Peñuela and Rinke Hoekstra. 2017. Automatic query-centric API for routine access to Linked Data. In *The Semantic Web—ISWC 2017: 16th International Semantic Web Conference*, Claudia d'Amato et al. (Eds.). Vol. 2. Springer, Cham, 334–339.

[19] Nicola Orio. 2006. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval* 1, 1 (2006), 1–90.

[20] Colin Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*. Ph.D. Dissertation. Columbia University.

[21] Colin Raffel and Daniel P. W. Ellis. 2014. Intuitive analysis, creation and manipulation of MIDI data with `pretty_midi`. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, Taipei, Taiwan,* Late-Breaking/Demo session.

[22] Yves Raimond, Samer Abdallah, Mark Sandler, and Frederick Giasson. 2007. The Music Ontology. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, Vienna, Austria*. 417–422.

[23] Alan Smaill, Geraint Wiggins, and Mitch Harris. 1993. Hierarchical music representation for composition and analysis. *Computers and the Humanities* 27, 1 (1993), 7–17.

[24] Julián Urbano. 2014. MelodyShape at MIREX 2014 Symbolic Melodic Similarity. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, Taipei, Taiwan,* Music Information Retrieval Evaluation eXchange.

[25] Julián Urbano. 2015. MelodyShape at MIREX 2015 Symbolic Melodic Similarity. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, Málaga, Spain,* Music Information Retrieval Evaluation eXchange.

[26] Julián Urbano, Juan Lloréns, Jorge Morato, and Sonia Sánchez-Cuadrado. 2011. Melodic similarity through shape similarity. In *Exploring music contents: 7th international symposium, CMMR 2010,* S. Ystad, M. Aramaki, R. Kronland-Martinet, and K. Jensen (Eds.). Springer, Berlin, 338–355.

[27] Julián Urbano, Markus Schedl, and Xavier Serra. 2013. Evaluation in music information retrieval. *Journal of Intelligent Information Systems* 41, 3 (2013), 345–369.

[28] Valerio Velardo, Mauro Vallati, and Steven Jan. 2016. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal* 40, 2 (2016), 70–83.

[29] Anja Volk, Elaine Chew, Elizabeth Hellmuth Margulis, and Christina Anagnostopolou. 2016. Music similarity: Concepts, cognition and computation. *Journal of New Music Research* 45, 3 (2016), 207–209.

[30] David M. Weigl and Kevin R. Page. 2017. A framework for distributed semantic annotation of musical score: "Take it to the bridge!". In *Proceedings of the 17th International Society for Music Information Retrieval Conference, Suzhou, China*. 221–228.