A Distributed Data Space for Music-Related Information

Yves Raimond, Christopher Sutton, Mark Sandler
Centre for Digital Music
Queen Mary, University of London
{yves.raimond,chris.sutton,mark.sandler}@elec.qmul.ac.uk

ABSTRACT

In this paper, we describe how some key Semantic Web technologies can be used to gather in a single distributed knowledge environment several music-related sources of information, from digital archives to feature extractors or personal music collections. Such knowledge can then be used for a wide range of purposes, such as aggregation and information retrieval, visualisation and enriched access, or cross-repository interlinking. We also describe on-going efforts aiming at bootstrapping such a data-space, as well as preliminary results.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design

Keywords

Semantic-Web,Multimedia,Music,MIR

1. INTRODUCTION

Information management is becoming an important part of multimedia related technologies, ranging from the management of personal collections to the construction of large cultural archives or the storage of analysis results. Solutions to each of these problems have emerged, such as GreenStone [1] or Fedora [2] in the case of digital archive management, but so far all these frameworks are isolated from each other. This becomes a greater problem when we extend our range of interest to other datasets, such as personal music collections, Creative Commons music repositories or automatically extracted features which can reveal the inner musical structure of a particular piece. There is no way to make one of these datasets benefit from the knowledge another one may hold—even between instances of the same program.

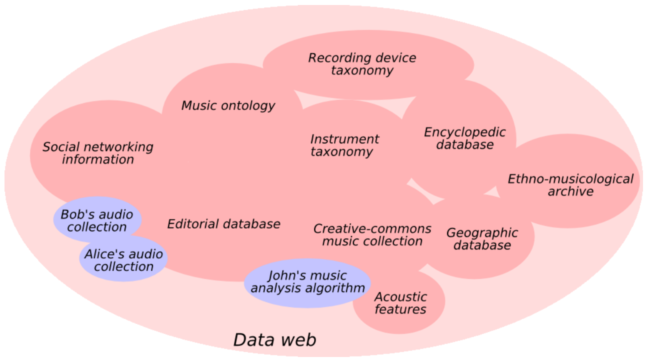

Figure 1: Linking music-related data on the Semantic Web

At the same time information access is tremendous on the current web of documents, and the need for a more structured web, accessible and understandable by software agents as well as human agents, is becoming obvious. As a result, new languages and standards for structuring the information available on the web have emerged, such as Microformats, RDF or OWL—Semantic Web technologies. These technologies, when bound with other web technologies, allow the creation of a *web of data*, which defines itself in an organic way by the inclusion of *ontologies* into it.

In this paper, we explain how a distributed and democratic knowledge environment based on the technologies described in § 2 can act as a *data hub* for many music-related applications and data sources, as depicted in fig. 1. In particular, we explain in § 3 how we can link together freely available music-related information repositories, digital archives and personal music collections. In § 4, we detail how music analysis algorithms can also take part in this web of data, in order to try bridge the gap between Music Information Retrieval researchers and actual music consumers. Finally, in § 5, we detail further applications which illustrate the considerable potential benefits of linking the data from all these sources.

2. TOWARDS A WEB OF DATA

As mentioned in the introduction, there is great incentive to make currently published information on multimedia resources available in a common, structured, interlinked format. Tim Berners-Lee's vision of the Semantic Web [3], and the vast array of technologies already established in pursuit of it, provide just the functionality required to begin build-

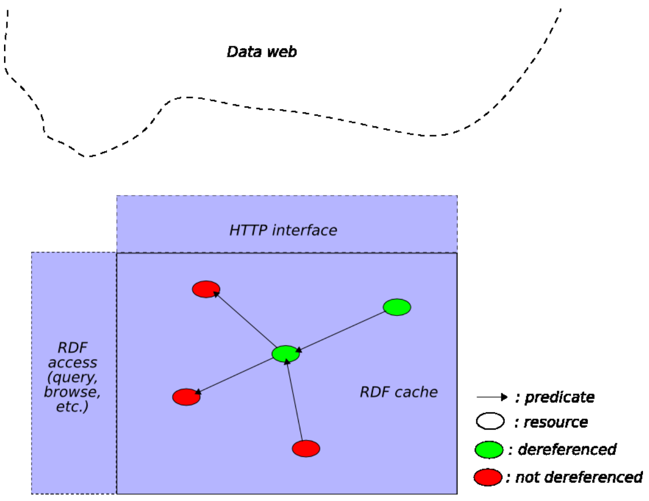

Figure 2: Overview of a Semantic Web user agent

ing such a “web of data”. We discuss below the component technologies currently being used. In § 2.1 we introduce RDF and the use of HTTP URIs to identify resources. In § 2.2 we discuss the ways such data can be usefully consumed by various types of user agents in a trustworthy way, as mentioned in § 2.3. Examples of such user agents are given in section § 2.4.

2.1 Dereferencable identifiers and machine-readable descriptions

The W3C’s Resource Description Framework (RDF) [4] allows the description of resources by expressing statements about them in the form of *triples*: *subject*, *predicate* and *object*. Each element of such a triple is specified by a URI (Unique Resource Identifier) and a set of triples may be interpreted as a graph of these resources, with arcs corresponding to the relationships between them.

RDF alone then provides us with a common, structured format for expressing data. *Interlinking* of data sets may be achieved by ensuring URIs are unique across data sets, and providing a common access mechanism for following references between data sets. In practice, the existing HTTP protocol proves ideal for this task. If each resource is identified by a HTTP URI (eg. `http://example.com/resource7341`), we obtain an established system for “ownership” of URIs, and we can traverse data sets using simple HTTP GET operations.

A user agent, wishing to know more about a resource x, *dereferences* the URI of x by performing a HTTP GET operation on the URI address, and receives RDF data containing triples related to the resource, as depicted in fig. 2. This allows dynamic exploration of linked data sets [5] which may be as distributed across geographic locations, institutions and owners as the traditional Web.

This idea of using HTTP addresses to provide machine-readable descriptions of resources may seem at odds with the current usage of the HTTP namespace. However there exist various techniques (eg. specification of content-type by user agents, content negotiation by 303 redirects, embedded (micro)formats such as RDFa) to allow the same HTTP address to provide both machine-readable RDF data and human-readable HTML data describing a resource. The Semantic Web can therefore be built alongside the current web, and a content publisher with knowledge of Semantic Web technologies can ensure his published data is useful both directly to a human reader via a traditional web browser, and to a Semantic Web user agent possibly performing reasoning and deduction on behalf of a human user.

2.2 Semantic Web user agents

We use the term “user agent” to describe any software acting directly on user requests, and a “Semantic Web user agent” is then one which accesses resources on the Semantic Web in order to satisfy a user’s demands. One example of a Semantic Web user agent would be a simple browser, analogous to a modern web browser, which allows the user to navigate data on the Semantic Web just as a web browser allows a user to navigate web sites.

Although we are beginning to see quite sophisticated use of traditional web resources (such as web “scrapers” which extract data from webpages, scripts which modify webpage content on-the-fly [1], and “mashups” which dynamically combine the functionality of multiple sites[2]), the traditional web was designed solely for the “browsing” use case. The Semantic Web on the other hand is designed to allow much more complex interaction with available data sources. So the term “Semantic Web user agent” also encompasses programs which automatically explore and dereference extra resources in order to satisfy a user’s query.

A simple example of this is the Tabulator [6] (see § 2.4) which automatically dereferences resources if they are semantically marked “the same as” or “see also” from the resource the user requested information about, and then provides the user with a conflated view of all the information found. This “automatic exploration” approach is only one way for a user agent to exploit Semantic Web resources, and for more complex queries or larger volumes of data other access mechanisms (such as SPARQL [7]) may be more efficient. Further examples of user agents which take advantage of semantic information are given in § 2.4 below.

Having information about the *type* of each resource and its *relationships* to other resources also allows a user agent to present the data to the user in an appropriate format. This can be done by using a display language such as Fresnel [8] to specify how to display data from a particular ontology, or by explicitly writing user agents to suit particular information domains—something that is not possible with the traditional web. Software written to suit a particular set of ontologies and display resources and relationships from those ontologies appropriately might also allow the inclusion of newly-encountered ontologies, and (by performing reasoning on their relationships to known ontologies) suitably display resources and relationships from those ontologies also.

2.3 Confidence handling

A potentially infinite number of agents can publish information on the Semantic Web—anyone can be a data publisher or consumer. Therefore, there is a need to implement some sort of confidence handling at the user agent level. The Named Graph approach [9] provides a formal framework which can act as a basis for a trustworthy Semantic Web. This relies on the fact that every piece of information

[1]see `http://greasemonkey.mozdev.org/`
[2]see `http://www.programmableweb.com/`

aggregated into a user agent has a provenance, which should itself be modelled within the agent's knowledge base. For example, if a user agent aggregated two pieces of information respectively stating that `John Doe is a PhD student` and `John Doe is a lecturer`, it should also keep track of the *provenance* of these statements. The first statement could come from `http://johndoeuniversity.edu/students/` and the second could come from `http://anotheruniversity.edu/staff/`. Then, the user agent has a way to trust one information source more than another (in our example, to specify that the first source is probably outdated).

2.4 Implementations

As discussed in § 2.2, we can distinguish between Semantic Web user agents based on whether they are agnostic to the domain of their data or designed for a particular domain, and whether they are restricted to exploration by browsing or can retrieve data in more sophisticated ways (such as answering queries).

The **Tabulator** and the **Disco - Hyperdata Browser**[10] are both generic browsers for Semantic Web data. The Tabulator allows interactive traversal of data (both locally stored documents, or remote HTTP resources) with some basic reasoning (see § 2.2 above) and can visualise selected properties in several ways, including a table view, timeline view, and a Google map view. The Tabulator is open source and can be extended with custom views, allowing the possibility of adaptation to particular information domains. Development is ongoing, and recent work is focused on allowing the user to edit data as they browse.

The Disco - Hyperdata Browser provides a simpler browsing interface, in which the user views properties of one resource at a time, and all resources dereferenced are cached in a graph for the current session. Disco takes a more proactive approach than the tabulator, dereferencing every URI mentioned in the requested URI's properties. Like the tabulator, it can perform simple merging of resource information based on "same as" relationships.

The **mSpace Classical Music Explorer**[11] is a good example of a Semantic Web user agent tailored to suit a particular domain. Using classical music as a test case, researchers have applied original ideas on user interface design to the creation of a very flexible and easy-to-use information browser. An important distinction between this and the other user agents presented here is that the mSpace explorer operates primarily on a closed data set stored on the server, rather than dynamically finding new sources of Semantic Web data.

Other user agents for specific domains include the FOAF Explorer and foafnaut for the Friend of a Friend ontology, and the Geonames browser which lets the user explore geographical data from several large databases through interactive maps.

The **Semantic Web Client Library**[12] and **SWIC**[13] are two libraries designed to allow more sophisticated modes of use than simple browsing, by modelling the whole Semantic Web as a single graph and allowing queries to be executed on that graph. The client library explores the graph, dereferencing resources in order to satisfy the query.

3. A MUSIC-RELATED WEB OF DATA

In discussing music data online, we can distinguish between data (the music itself, whether sampled (eg. MP3) or symbolic (eg. MIDI)), and metadata (information about the music (eg. ID3 tags, album reviews, track listings)). There is a vast amount of musical metadata currently online, some of it provided quite free of restrictions by an open community (eg. the MusicBrainz database, FreeDB CD listings, the MusicMoz directory, articles in Wikipedia) and some of it under copyright of commercial entities (eg. the All Music Guide, music product information on Amazon, CDDB CD listings, and the Grove dictionary of music). There are also considerable resources for music data. Generally music data is available for sale rather than free download (eg. the iTunes Music Store), but there are also free audio data resources available (eg. from the Jamendo and the Magnatune labels).

For the most part these resources all exist in isolation, despite providing similar kinds of information. In most cases, interlinking between these resources would be of benefit to all concerned, but instead each data source uses its own identifiers, data formats and APIs. This necessitates the writing of "glue code" to combine data sources (for example a mash-up which uses your Last.FM listening profile to plot your recently-heard artists on a map), and new code must be written for each desired combination. If this data were integrated into the web rather than just being accessible through particular web pages, such "glue code" would be unnecessary, and a generic user interface could allow arbitrary reuse and combination of data. Some efforts in this direction are discussed below.

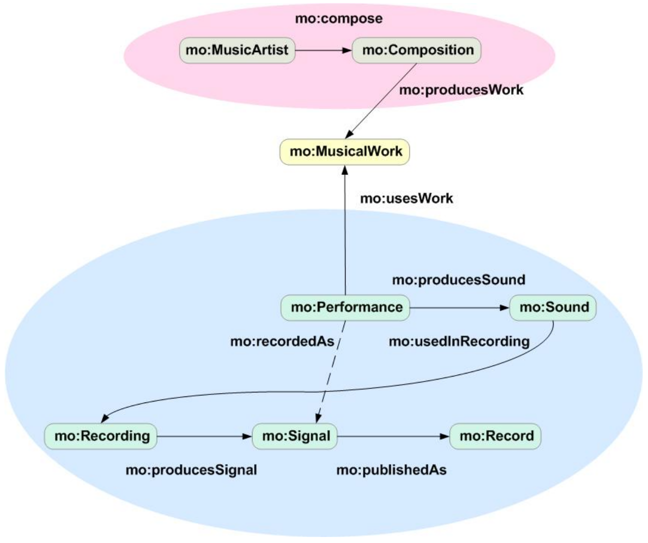

Figure 3: Describing a music production process using the level 2 of the Music Ontology

3.1 Overview of the Music Ontology

Integration and interlinking of data sources is possible even when they don't share a common ontology, but far easier when they do. The Music Ontology [14] has been created to provide a standard base ontology for describing musical information. It can currently describe a wide range of music information at three distinct levels of detail— Level 1 describes top-level editorial information such as is found in an ID3 tag; Level 2 describes the process behind the production of music, whether in the studio, on a home PC, or in concert; Level 3 allows low-level description of the structure and component *events* of the music being played, such as the notes, chords, or samples being played. A depiction of the main concepts in level 2 can be found in fig. 3.

The Music Ontology builds on existing ontologies, most notably Functional Requirements for Bibliographic Records [15], the Timeline [16] and Event [17] ontologies, and the Friend-of-a-Friend [18] ontology. No one ontology could hope to cover the requirements of all music descriptions, and so the Music Ontology is designed to allow extension with specialised ontologies. For example, the ontology itself provides only very basic instrument and genre terms, but has been extended by using the SKOS adaptation of the MusicBrainz instrument taxonomy [19], and the DBpedia [20] adaptation of Wikipedia's genre taxonomy.

The following listing shows an example description (in Turtle notation [21], which will be used throughout all the RDF examples in the paper) of a music track using the Music Ontology, which is represented graphically in fig. 4.

```
@prefix : <http://example.org/joco_info/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix event: <http://purl.org/NET/c4dm/event.owl#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix timeline: <http://purl.org/NET/c4dm/timeline.owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix yago: <http://dbpedia.org/class/yago/> .
@prefix geo: <http://www.geonames.org/ontology> .
###########################  Level 1  ####################################
:joco
        rdfs:type      mo:MusicArtist ;
        rdfs:label     "Jonathan Coulton" ;
        owl:sameAs
   <http://zitgist.com/music/artist/d8df7087-06d5-4545-9024-831bb8558ad1>.
:rit_surr_sound
        rdfs:type      mo:MusicGroup ;
        foaf:name      "RIT's Surround Sound" ;
        foaf:homepage  "http://www.ritsurroundsound.org/" .
:taw3
        rdfs:type      mo:Record ;
        rdfs:label     "Thing a Week III" ;
        mo:has_track   :c_m_album_version ;
        owl:sameAs
   <http://zitgist.com/music/record/ba027f50-90a8-4e36-9f95-871d66978ffa>.
:code_monkey
        rdfs:type      mo:MusicalWork ;
        dc:title       "Code Monkey" .
:c_m_album_track
        rdfs:type      mo:Track ;
        dc:title       "Code Monkey" ;
        dc:creator     :joco ;
        owl:sameAs
    <http://zitgist.com/music/track/ba236f3d-1f21-406b-9870-3d33422c1509>;
        mo:freeDownload
      <http://www.jonathancoulton.com/music/thingaweek/CodeMonkey.mp3>.
###########################  Level 2  ####################################
:c_m_comp
        rdfs:type          mo:Composition ;
        rdfs:label         "Composition of Code Monkey (Thing a Week 29)";
        event:hasAgent     :joco ;
        event:hasProduct   :code_monkey ;
        event:time
      <http://placetime.com/interval/gregorian/2006-04-10T00:00:00Z/P5D> .
### Album performance
:c_m_album_perf
        rdfs:type           mo:Performance ;
        mo:usesWork         :code_monkey ;
        mo:performer        :joco ;
        event:time
      <http://placetime.com/interval/gregorian/2006-04-10T00:00:00Z/P5D> ;
        mo:producesSound    :c_m_album_sound .
:c_m_album_sound
        rdfs:type           mo:Sound ;
        mo:usedInRecording  :c_m_album_recording ;
        mo:usedInRecording  :c_m_remix .
:c_m_album_recording
        rdfs:type           mo:Recording ;
        mo:producesSignal   :c_m_album_signal .
:c_m_album_signal
        rdfs:type           mo:Signal ;
        mo:publishedAs      :c_m_album_track .
### Live performance
:c_m_live_perf
        rdfs:type          mo:Performance ;
        rdfs:label         "Performance of Code Monkey (video available)";
        event:hasAgent     :joco ;
        event:usesWork     :code_monkey ;
        event:place        :TempleBar ;
        mo:availableAs     <http://www.youtube.com/watch?v=j4TnhemCEmc> .
:TempleBar
        rdfs:type          yago:venue ;
```

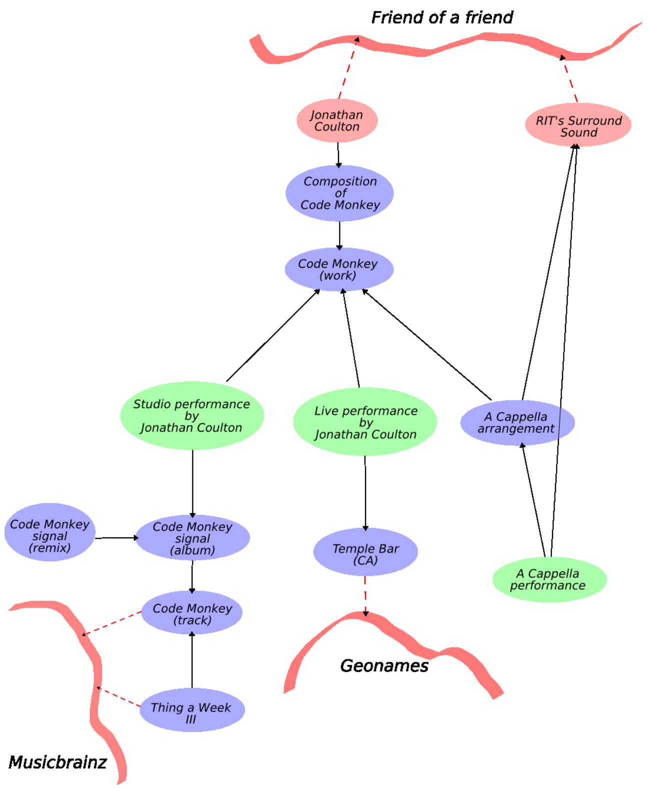

Figure 4: Depiction of a description using the Music Ontology

```
        foaf:name          "Temple Bar, Santa Monica, CA" ;
        geo:locatedIn      <http://sws.geonames.org/5393212/> .
### A Cappella arrangement & live performance
:c_m_acappella_arr
     rdfs:type        mo:Arrangement ;
     rdfs:label       "A Cappella arrangement of Code Monkey" ;
     event:hasFactor  :code_monkey ;
     foaf:Agent       :rit_surr_sound .
:c_m_acappella_perf
     rdfs:type        mo:Performance ;
     rdfs:label       "A Cappella Code Monkey Performance (video avail.)";
     foaf:Agent       :rit_surr_sound ;
     event:hasFactor  :c_m_acappella_arr ;
     mo:availableAs   <http://www.youtube.com/watch?v=vsKBQLz8yrI> .
### "Speed Monkey" remix
:c_m_remix_signal
     rdfs:type        mo:Signal ;
     mo:remix_of      :c_m_album_signal ;
     mo:availableAs   :c_m_remix .
:c_m_remix
     rdfs:type        mo:Track ;
     dc:title         "Speed Monkey" ;
     mo:releaseType   mo:remix ;
     foaf:homepage  <http://art.twobrotherssoftware.com/speedmonkey.html>;
     mo:freeDownload
            <http://www.art.twobrotherssoftware.com/music/speedmonkey.mp3>.
```

3.2 Linking music-related open-data

The *open data* movement aims at making data freely available to everyone. Such data sources cover a wide range of topics: from music (Musicbrainz, Magnatune or Jamendo) to encyclopedic information (Wikipedia) or bibliographic information (Wikibooks, DBLP bibliography). The "Linking Open Data on the Semantic Web" community project [22] aims at interlinking such sources of information, using the technologies described in § 2.1. For example, if we provide a description of a particular artist, we may want to, instead of providing a complete geographic information by ourselves, just link the artist resource to a location in the Geonames dataset, which provides additional knowledge about this location, such as hierarchical relationships with other geographical entities, latitude, longitude, etc. Then an agent crawling the Semantic Web can jump from our knowledge base to the Geonames one by following this link.

This interlinking process comes in two steps. First, the data sources must be wrapped in order to provide identifiers for the resources they hold, and be able to provide RDF descriptions of these resources on demand. Then, links between datasets can be created.

The Music Ontology described in § 3.1 helps this process for music-related information. It provides a framework for publishing heterogeneous music-related content in RDF—from MusicBrainz to the Royal Scottish Academy for Music and Drama (RSAMD) HotBed database. Moreover, it provides links to other ontologies, covering other domains, such as reviews [23], social networking information [18], geographic information [24] or *acoustic* information [25] related to the music data itself.

Several music-related datasets have already been made available. The Jamendo and the Magnatune Creative Commons music collections have been made available as linked data [26], using **P2R** [27] and **UriSpace** [28]. **P2R** provides SPARQL access to SWI-Prolog [29] knowledge bases, according to a declarative mapping[3]. **UriSpace** provides content negotiation on identifiers whose description lies in a SPARQL end-point. Therefore, it is able to provide a RDF representation to Semantic Web user agents, and a HTML one to web browsers. The Musicbrainz dataset, holding detailed editorial information about more than 300 000 artists, has also been made available [30] through the use of the **OpenLink Virtuoso** [31] SQL to RDF mapper.

These datasets also hold links to other datasets. For example, many resources representing artists, records or tracks in the Jamendo and the Magnatune datasets are declared as being the same as corresponding resources in the Musicbrainz dataset (through the `owl:sameAs` property). These links were created automatically using a variant of the Similarity Flooding algorithm by Melnik et al. [32], working in a linked data environment. Moreover, when a description of an artist in the Jamendo dataset is requested, a call to the Geonames web service is done in order to find if there is a matching geographic location corresponding to the simple literal (such as “Paris, France”) available in the Jamendo database. If we succeed in finding a matching location, we add a statement to our artist description making the link from the Jamendo dataset to the Geonames one (using the `foaf:based_ near` predicate), thereby allowing user agents to access detailed information about the location of the artist. The RSAMD HotBed database was also published [33], and is currently being linked to the Geonames dataset.

3.3 Management of personal music collections

Personal music collections can also be a part of such a web of data. The Music Ontology makes the same distinction as FRBR between manifestations (all physical objects that bear the same characteristics—a particular record, for example) and items (a concrete entity, such as a particular CD or a particular audio file). A manifestation and a corresponding item are linked through a predicate `mo:availableAs`. Therefore, given a set of audio files in a personal music collection, it is possible to keep track of the set of statements linking this collection to identifiers denoting the corresponding manifestations available elsewhere in the Semantic Web. These statements provide a set of entry points to the Semantic Web, allowing one to access information such as the birth date of the artists responsible for some of the items in the collection, geographical locations of the recordings, etc. **GNAT**[4] is an automatic linking implementation, from a personal audio collection to the Musicbrainz dataset—it uses available ID3 tags (which can themselves be set correctly using an automatic tagging application using acoustic fingerprint information, such as **Picard**[5]) to find corresponding dereferencable identifiers, and then outputs RDF statements making the links between local audio files and the remote manifestation identifiers. This could be used to build small applications to enhance the user experience: for example, placing the audio collection on a timeline and allowing the generation of playlists holding only songs composed during a particular decade, placing the collection on a map and allowing the user to create playlists going from one place to another, etc.

3.4 Digital archives interlinking

As mentioned in the introduction, instances of digital archive management software (such as Greenstone [1]) are often isolated from each other, even though they could really benefit from interlinking the knowledge they hold. For example, a composer described in one archive could be linked to some musical works described in another archive. The BRICKS [34] and EASAIER [35] projects both try to overcome these limitations, and make several archives aware of each other and of external datasets.

The range of links that it is possible to create in a Semantic Web environment is not bound to range over a particular type of application: an open dataset, a personal music collection and a digital archive can be linked together— the Semantic Web therefore acts as a *data hub*.

Taking benefit from the Semantic Web can be done in two steps. The archive management software should first implement a component that is able to provide access to the information it holds using dereferencable identifiers and RDF. Then, links from one archive to another dataset (which may be another archive exposing data in the same way, open datasets, etc.) can be created either automatically (as mentioned in § 3.2) or manually.

For example, an ethnomusicological archive may take benefit from being linked to a geographical dataset such as Geonames. In this way, the archive can be tightly focused on its primary topic, and leave the burden of extra descriptions to other focused datasets.

4. DYNAMIC RESOURCES

So far we have discussed only *static* resources—data is stored somewhere, and looked up when requested. However, the Semantic Web interface can also be used to access *dynamic* resources which are computed only when requested (and possibly then cached for future requests). In the music realm, this means that current research algorithms (for

[3]such a mapping can be found at `http://moustaki.org/resources/jamendo_match.pl`

[4]the code is available at `http://moustaki.org/resources/gnat.tar.gz`

[5]see `http://musicbrainz.org/doc/PicardTagger`

example the ones mentioned in [36]) for tempo and rhythm estimation, harmonic analysis (such as chord transcription), partial transcription, or source separation could be exposed as Semantic Web resources. This could be of great benefit both to researchers, allowing them to more easily compare their results with others' algorithms, and to the general public by letting them use research algorithms without researchers needing to design end-user applications. We detail in this section some steps in this direction.

In order to export the results of our music analysis algorithms dynamically to this distributed data space, we need to create placeholder dereferencable resources which, when accessed, trigger a computation to generate a RDF description or look for cached results. Moreover, we want every Semantic Web user agent to be able to access these results without any knowledge of the underlying framework: these placeholder resources must be part of the ones the user agent would have dereferenced if the actual results were already available in the web. We now detail an encapsulation strategy which meets this requirement.

4.1 Modelling music analysis algorithms

First, we consider every analysis algorithm as a predicate, associated to a particular mode and determinism. For example, we would express a trained instrument classifier as in (1) or an algorithm computing a feature-based similarity between two audio signals as in (2)[6].

```
instrument_recognition(+Signal,-Instrument)
        is semi-deterministic. (1)
similarity(+Signal1,+Signal2,-Distance)
        is deterministic. (2)
```

When queried in the right mode (with all the input arguments bound), a computation will be launched, and the output arguments will be associated with the corresponding results, otherwise, the predicate will be false. Input arguments can be bound using either local content databases, associated to remote identifiers through a process similar as the one described in § 3.3, or access to aggregated Semantic Web data (through the Jamendo or the Magnatune SPARQL end-points, for example).

4.2 RDF view of music analysis algorithms

Then, we consider the following axioms[7], dealing with the RDF representation of the algorithms mentioned earlier. The predicate *signal* looks for an available signal URI (whose sample values will be parsed as an array of numbers inside the analysis predicates if the actual signal is available), either in a local database, or in aggregated Semantic Web data. The predicate *threshold* is true if its argument is superior to a fixed value. The predicate *rdf* is true if the three arguments constitute a RDF statement.

$$\begin{aligned} &\mathit{rdf}(Signal, mo: instrument, Instrument) \Leftarrow \\ &\quad signal(Signal) \\ &\quad \wedge\, instrument\_recognition(Signal, Instrument) \end{aligned}$$

[6]`+` denotes an input argument, `-` an output one, the `is` operator here associates a mode declaration to a particular determinism class, and we use the Prolog lexical convention for distinguishing variables from constants—variables start with an uppercase character

[7]we just use the following operators: ∧ (and) and ⇐ (if)

$$\begin{aligned} &\mathit{rdf}(Signal1, mo: similar\_to, Signal2) \Leftarrow \\ &\quad signal(Signal1) \wedge signal(Signal2) \\ &\quad \wedge\, similarity(Signal1, Signal2, Distance) \\ &\quad \wedge\, threshold(Distance) \end{aligned}$$

Now, if we put a SPARQL end-point on top of such entailment rules, a DESCRIBE query on any accessible signal will dynamically derive statements holding information about the musical instrument used and about similar signals. This behaviour in fact raises a problem: computations may be resource-intensive, so we do not want to compute things that the user agent is not interested in. We therefore need a more sophisticated dynamic resource handling mechanism.

4.3 Advertising dynamic resources

We consider just deriving, at first, *advertisement* statements from such entailment rules, which will provide only enough data for the user agent to reach a resource that, when dereferenced, will trigger a unique class of computations. For example, a SPARQL end-point providing such a mechanism on top of the two rules defined earlier will issue just two statements, when a DESCRIBE query is done on an accessible signal `ex:signal`:

```
@prefix ex: <http://example.org/>.
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix : <>.

ex:signal mo:instrument :fake1.
ex:signal mo:similar_to :fake2.
```

Then, when a DESCRIBE query is issued on `:fake1`, the first entailment rule is derived in order to find the actual relationship between the signal and an instrument, and we append the corresponding statement to the returned description. In order to make it totally transparent for the client, we also state that `:fake1` is the same as the matching output. Many user agents know how to interpret a `owl:sameAs` statement, and so in practice that statement alone would suffice to relate the signal to the instrument. However, we try here to accommodate a large range of possible user agents, including those with no reasoning support. This leads us to the following RDF document, accessed when issuing a DESCRIBE query on `:fake1`:

```
@prefix ex: <http://example.org/>.
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix mit: <http://purl.org/ontology/mo/mit#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix : <>.

:fake1 owl:sameAs mit:Contrabassoon.
ex:signal mo:instrument mit:Contrabassoon.
```

When a DESCRIBE query on `:fake2` is issued, the following query is considered:

$$\begin{aligned} &signal(ex: signal) \wedge signal(Signal2) \\ &\wedge\, similarity(ex: signal, Signal2, Distance) \\ &\wedge\, threshold(Distance) \end{aligned}$$

This query is clearly multi-solution (zero or multiple matching bindings): we derive a distance measure from ex:signal to each other accessible signal, and threshold them against a fixed value. Therefore, we must derive all the possible outputs (ie. consider all possible similar audio signals) and send back the following document:

```
@prefix ex: <http://example.org/>.
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix : <>.

:fake1 owl:sameAs ex:signal2.
:signal mo:similar_to ex:signal2.
:signal mo:similar_to ex:signal3.
[...]
```

The :fake1 resource is stated as being the same as one of the similar signals.

Using such a mechanism, only the computation that the user agent is interested in will be triggered. If the user agent looks for an instrument associated to a signal, it will just trigger a call to instrument_recognition. If it looks for similar signals, it will trigger a call to similarity.

Then, we consider *tabling* of predicates such as similarity or instrument_recognition: we cache every computed instantiation of them, in order to save us from performing the same computation twice through two similar queries. For example, when an evaluation of $similarity(signal_1, Signal, Distance)$ leads to a set of facts $similarity(signal_1, signal_n, distance_n)$, we store these facts in order to directly retrieve them when similar queries are evaluated.

4.4 Publishing results in the Semantic Web

After embedding the described mechanism in a SPARQL end-point, we just need to map the identifiers we use to DESCRIBE queries onto the end-point, therefore making them dereferencable and part of the Semantic Web. A user agent interacting with such identifiers will only launch the computations that exactly fit his request.

Now, we can annotate the RDF documents dynamically generated in such a framework. This can hold various information, such as the computation time taken, a link to the DOAP [37] URI of the program which generated the actual results, an associated confidence, etc. For example, the RDF document retrieved when dereferencing :fake1 could look like:

```
@prefix ex: <http://example.org/>.
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix : <>.

ex:signal mo:instrument :fake1.
ex:signal mo:similar_to :fake2.

<> foaf:maker ex:my_instrument_classifier_project;
        ex:computationTime "PT3.12S"^^xsd:duration;
        ex:cachedResult "true"^^xsd:boolean;
        ex:confidence "0.7"^^xsd:float;
        .
```

4.5 Stacked interpretations

This mechanism allows us to provide dynamic access to a variety of features, from low-level (eg. onsets occuring on a signal timeline) to high-level (eg. overall tempo, or recognised instruments). Since only *advertisement* statements are provided in the description of the resource of interest, this approach avoids flooding the user agent with feature data which is of a different nature, or different level of abstraction than the user agent is interested in, while still providing access to an arbitrarly large, dynamic, distributed dataset.

For example, in the case of an onset detection algorithm, the signal resource, when fetched, will provide the following representation:

```
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix af: <http://purl.org/ontology/af/>.
@prefix event: <http://purl.org/NET/c4dm/event.owl#>.
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#>.
@prefix ex: <http://example.org/>.
@prefix : <>.

<> mo:signalTime [
   tl:onTimeLine ex:timeline;
   ].

:fake_onset a af:Onset;
   event:time [
      tl:onTimeLine ex:timeline;
      ];
   .
```

Then, a user agent which is interested in such low level features can access this :fake_onset resource, receiving a large set of onset events occuring on the timeline of the resource we are interested in. This data set can also be distributed, using links such as rdfs:seeAlso.

A user agent consuming such a resource might simply be a browser or visualiser of some kind, but could instead be a computation resource itself, accessing such features in order to derive higher-order knowledge and link it with available resources. In this particular example, the information output by the onset detector could be used by a tempo detection algorithm. Such an agent would then provide an additional advertisement statement indicating that it can provide a tempo event for this particular timeline:

```
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix af: <http://purl.org/ontology/af/>.
@prefix event: <http://purl.org/NET/c4dm/event.owl#>.
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#>.
@prefix ex: <http://example.org/>.
@prefix : <>.

:fake_tempo a af:TempoEvent;
   event:time [
      tl:onTimeLine ex:timeline;
      ];
   .
```

When :fake_tempo is dereferenced, the tempo detection agent will dereference :fake_onset in order to analyse the available onsets, and then send back a tempo value with time stamps delimiting an interval on which this tempo value was computed.

In this way the proposed framework provides a way to collaboratively stack interpretations.

4.6 Implementation

Audio features extracted by the playlist creation tool **SoundBite**[8] were used to implement a service augmenting Musicbrainz track information with links to similar-sounding tracks, using the similar_to relation from the

[8]see http://isophonics.net/SoundBite

Music Ontology. When a signal resource is retrieved from this service, it is marked as being the `sameAs` the corresponding Musicbrainz resource (which provides the standard textual metadata for the track), and a single `similar_to` link is provided. When this advertisement resource is dereferenced, the actual calculation is performed, and the set of links to similar tracks returned. Continuing with the example in § 3.1, dereferencing the URI `http://isophonics.net/music/signal/ba236f3d-1f21-406b-9870-3d33422c1509` retrieves an advertisement for tracks similar to Code Monkey and when this advertisement resource `http://isophonics.net/SBSimilarity/ba236f3d-1f21-406b-9870-3d33422c1509` is dereferenced, triples describing tracks similar to Code Monkey are obtained.

The proposed mechanism has also been implemented within the **Knowledge Machine** framework ([38] gives an overview of this framework, but does not explain the mechanism used for dynamic computation and interlinking of results), through the use of a SWI-Prolog meta-interpreter handling mode and determinism declaration, as well as function tabling. Access to remotely accessible data is done through **SWIC**, a small Semantic Web client. On top of it sits a component implementing the above detailed mechanism—it provides a linked data access to fake resources, which actually reflect possible computations. We plan to integrate this functionality into **P2R**, in order to provide a simple access to such functionalities.

As discussed in § 2.4, development of domain-specific semantic web user agents has yet to flourish, and so there are no existing user applications with which to consume this data in a more complex fashion than simply browsing. It is difficult therefore to directly evaluate the proposed approach to making data and computation available online, as the main benefit is expected to be in interoperability and ease of client development. However the very fact that a generic browser can make use of the resources published in this fashion is a significant departure from the traditional web service and Service Oriented Architecture approaches [39], and some directions for future work in developing more sophisticated music-specific clients is discussed in § 5 below.

5. CONCLUSION AND FURTHER WORK

The current music-related web of data, as detailed in § 3.2, mainly holds knowledge coming from open-data repositories, such as Musicbrainz, Creative Commons labels, and Wikipedia. A wide range of tools allow us to wrap existing information and to provide Semantic Web access to it. Among them, we can cite **D2R** or **OpenLink Virtuoso** for mapping relational databases to RDF published in a linked data way, and **P2R** for wrapping a Prolog knowledge-base (which itself may wrap computational tools, calls to web services or queries to relational databases). Moreover, several *linking* tools are also available, such as **GNAT**, linking personal music collections to corresponding identifiers in the Musicbrainz dataset, or **ldmapper**, finding similar resources in two datasets available as linked data. Moreover, we detailed in this paper how the Semantic Web can also act as a data hub for dynamically computed acoustic features and interpretations of them, potentially allowing us to bridge the Semantic Gap in a collaborative fashion.

Further work includes the creation of concept-specific data visualisers, allowing us to map particular concepts available on the Semantic Web to a relevant visualisation. For example, we might create a user interface similar to the Sonic Visualiser [40] which could display data available on the Semantic Web—this could allow the visualisation of features created according to the mechanism described in § 4. Moreover, end users (digital music consumers) could really benefit from having digital jukebox software which accesses this web of data—not only to provide more information on the available audio items, but also to interpret computable features, enabling new modes of interaction with music. For example, such data could be used for song browsing at the structural level (chorus, verse, etc. or key changes) or playlist generation according to some criteria (such as key, rhythm or melody, but also geographic or temporal information). Digital jukeboxes could also contribute new information to this web of data, through embedded analysis algorithms which post their results to a Semantic Web *proxy*.

6. ACKNOWLEDGEMENTS

The authors acknowledge the support of both the Centre For Digital Music and the Department of Computer Science at Queen Mary University of London for the studentship for Yves Raimond. This work has been partially supported by the EPSRC-funded ICT project OMRAS-2 (EP/E017614/1), and by the European Community 6th Framework Programme project EASAIER (IST-033902).

7. REFERENCES

[1] “Greenstone digital library software.” [Online]. Available: http://www.greenstone.org/

[2] “Fedora digital object repository system.” [Online]. Available: http://www.fedora.info/

[3] T. Berners-Lee, J. Handler, and O. Lassila, “The semantic web,” *Scientific American*, 2001.

[4] O. Lassila and R. Swick, “Resource description framework (RDF) model and syntax specification,” 1998. [Online]. Available: citeseer.ist.psu.edu/article/lassila98resource.html

[5] T. Berners-Lee, “Linked data,” World wide web design issues, July 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html

[6] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, “Tabulator: Exploring and analyzing linked data on the semantic web,” in *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006. [Online]. Available: http://swui.semanticweb.org/swui06/papers/Berners-Lee/Berners-Lee.pdf

[7] E. Prud’hommeaux and A. Seaborne, “SPARQL query language for RDF,” 2005. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[8] “Fresnel.” [Online]. Available: http://simile.mit.edu/wiki/Fresnel

[9] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler, “Named graphs, provenance and trust,” in *Proceedings of the 14th international conference on World Wide Web*, vol. Semantic Web foundations. Chiba, Japan: ACM Press, 2005, pp. 613–622.

[10] “Disco.” [Online]. Available: http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/

[11] "mspace." [Online]. Available: http://www.mspace.fm/

[12] "Semantic web client library." [Online]. Available: http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/semwebclient/

[13] "Swi-prolog semantic web client." [Online]. Available: http://moustaki.org/swic/

[14] Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson, "The music ontology," in *To appear in the proceedings of the International Conference on Music Information Retrieval*, 2007.

[15] I. Davis and R. Newman, "Expression of core frbr concepts in rdf," Working draft, 2005. [Online]. Available: http://vocab.org/frbr/core

[16] Y. Raimond and S. A. Abdallah, "The timeline ontology," OWL-DL ontology, 2006. [Online]. Available: http://purl.org/NET/c4dm/timeline.owl

[17] ——, "The event ontology," OWL-DL ontology, 2006. [Online]. Available: http://purl.org/NET/c4dm/event.owl

[18] D. Brickley and L. Miller, "Foaf vocabulary specification," Working draft, 2005. [Online]. Available: http://xmlns.com/foaf/0.1/

[19] I. Herman, "Musicbrainz instrument taxonomy in skos," Working draft, 2007. [Online]. Available: http://purl.org/ontology/mo/mit/

[20] "Dbpedia." [Online]. Available: http://dbpedia.org/

[21] D. Beckett, "Turtle - terse rdf triple language," Working draft, 2006. [Online]. Available: http://www.dajobe.org/2004/01/turtle/

[22] "Linking open data on the semantic web." [Online]. Available: http://linkeddata.org/

[23] D. Ayers, "Review vocabulary," Working draft. [Online]. Available: http://purl.org/stuff/rev#

[24] "Geonames." [Online]. Available: http://geonames.org/

[25] Y. Raimond, "The musical features ontology," Working draft, February 2007. [Online]. Available: http://purl.org/ontology/mo/mf/

[26] "Dbtune." [Online]. Available: http://dbtune.org/doc/

[27] "Prolog-to-rdf." [Online]. Available: http://moustaki.org/p2r/

[28] "Urispace." [Online]. Available: http://moustaki.org/urispace/

[29] J. Wielemaker, "An overview of the swi-prolog programming environment," 2003.

[30] "Zitgist." [Online]. Available: http://www.zitgist.com

[31] "Openlink virtuoso." [Online]. Available: http://www.openlinksw.com/virtuoso/

[32] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm," Stanford University, University of Leipzig, Tech. Rep., 2001.

[33] "Hotbed in rdf." [Online]. Available: http://dbtune.org/hotbed/

[34] "The bricks community." [Online]. Available: http://www.brickscommunity.org/

[35] "Enabling access to sound archives through enrichment and retrieval." [Online]. Available: http://www.easaier.org/

[36] P. Herrera, J. Bello, G. Widmer, M. Sandler, O. Celma, F. Vignoli, E. Pampalk, P. Cano, S. Pauws, and X. Serra, "Simac: Semantic interaction with music audio contents," in *Proceedings of the 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies*, 2005.

[37] "Description of a project." [Online]. Available: http://usefulinc.com/doap/

[38] Y. Raimond, S. Abdallah, M. Sandler, and M. Lalmas, "A scalable framework for multimedia knowledge management," in *Proceedings of the 1st conference on Semantic and Digital Media Technologies*, ser. Lecture Notes in Computer Science, Springer, Ed., December 2006.

[39] D. McEnnis, C. McKay, and I. Fujinaga, "Overview of omen," in *Proceedings of the International Conference on Music Information Retrieval*, 2006.

[40] C. Cannam, C. Landone, M. Sandler, and J. P. Bello, "The sonic visualiser: A visualisation platform for semantic descriptors from musical signals," in *Proceedings of the International Conference on Music Information Retrieval*, 2006.