EURECOM
Sophia Antipolis

SORBONNE UNIVERSITÉ

PHD THESIS

In Partial Fulfilment of the Requirements for the
Degree of Doctor of Philosophy from Sorbonne University
Specialization: Data Science

Knowledge-based Music Recommendation: Models, Algorithms and Exploratory Search

Pasquale LISENA

Defended on 11/10/2019 before a committee composed of:

Reviewer	**Michel BUFFA**, Université Côte d'Azur, INRIA, Sophia Antipolis, France
Reviewer	**Mounia LALMAS**, Spotify, University College London, United Kingdom
Examiner	**Gaël RICHARD**, TELECOM Paris, France
Examiner	**Tommaso DI NOIA**, Politecnico di Bari, Italy
Examiner	**Pietro MICHIARDI**, EURECOM, Sophia Antipolis, France
Thesis Director	**Benoit HUET**, EURECOM, Sophia Antipolis, France
Thesis Co-Director	**Raphäel TRONCY**, EURECOM, Sophia Antipolis, France

Dedicated to my family

Acknowledgements

Firstly, I would like to sincerely thank my advisor Raphäel Troncy, for having strongly wanted me to start this PhD, for having directed and guided my research, for having constantly given value and importance to my work, and for having shared with me goals and responsibilities.

I would like to thank the members of my research group for their feedback and cooperation, in particular Enrico Palumbo, which shared with me most of this journey. An acknowledgement goes to the members of the DOREMUS project, which it was a pleasure to work with. I would like to thank also the research group of VU Amsterdam which welcomed me in early 2019 in their vibrant research environment, and in particular Frank van Harmelen, Albert Meroño Peñuela and Ilaria Tiddi.

A very special gratitude goes out to all colleagues at EURECOM which contributed to create a pleasant work environment, with a special thank to those that have become genuine friends for me here in France.

Finally, I would like to thank my parents, for having always supported me and having represented a standing point for my life. I would like to thank my grandpa Pasquale, the first of my supporters, that is for me a big source of motivation. Lastly, I am extremely grateful to Valeria, for having chosen to stay next to me and for having helped, sustained, inspired, and encouraged me.

Sophia Antipolis, 11 October 2019 ***Pasquale Lisena***

Abstract

Representing information about music is a complex activity that involves different sub-tasks. This thesis mostly focuses on classical music, researching how to represent and exploit rich metadata. Our main goal is to investigate knowledge representation and discovery strategies applied to classical music, including research topics such as Knowledge-Base population, metadata prediction and recommender systems. We first propose a complete workflow for the management of music metadata using Semantic Web technologies. We introduce a specialised ontology and a set of controlled vocabularies for the different concepts specific to music. Then, we present an approach for converting data, in order to go beyond the librarian practice currently in use, relying on mapping rules and interlinking with controlled vocabularies. Finally, we show how these data can be exploited. In particular, we study approaches based on embeddings computed on structured metadata, titles, and symbolic music for ranking and recommending music. Several demo applications have been realized for testing the previous approaches and resources.

Abrégé

Représenter l'information décrivant la musique est une activité complexe, qui implique différentes sous-tâches. Cette thèse porte principalement sur la musique classique et étudie comment représenter et exploiter ces informations. L'objectif principal est l'étude de stratégies de représentation et de découverte de connaissances appliquées à la musique classique, dans des domaines tels que la production de bases de connaissances, la prédiction de métadonnées et les systèmes de recommandation. Nous proposons tout d'abord une architecture pour la gestion des métadonnées de musique à l'aide des technologies du Web Sémantique. Nous introduisons une ontologie spécialisée et un ensemble de vocabulaires contrôlés pour les différents concepts spécifiques à la musique. Ensuite, nous présentons une approche de conversion des données, afin d'aller au-delà de la pratique bibliothécaire actuellement utilisée, en s'appuyant sur des règles d'appariement et sur l'interconnexion avec des vocabulaires contrôlés. Enfin, nous montrons comment ces données peuvent être exploitées. En particulier, nous étudions des approches basées sur des plongements calculés sur des métadonnées structurées, des titres et de la musique symbolique pour classer et recommander de la musique. Plusieurs applications de démonstration ont été réalisées pour tester les approches et les ressources produites.

Contents

List of Figures

List of Tables

List of Abbreviations

AI artificial intelligence.

API Application Programming Interface.

ASCII American Standard Code for Information Interchange.

BFS Breath First Search.

BnF Bibliothèque nationale de France.

BPM Beats per Minute.

CARS context-aware recommender system.

CDPL Choral Public Domain Library.

CF Collaborative Filtering.

CIDOC-CRM CIDOC Conceptual Reference Model.

COMUS Context-based Music Recommendation Ontology.

CSV Comma-separated values.

DCAT Data Catalog Vocabulary.

DH Digital Humanities.

DL Digital Library.

EM editorial metadata.

ETL extract, transform, load.

FRBR Functional Requirements for Bibliographic Records.

FRBRoo FRBR-object oriented.

HTML HyperText Markup Language.

IAML International Association of Musical Libraries.

IFLA International Federation of Library Associations and Institutions.

IRI Internationalized Resource Identifier.

ISNI International Standard Name Identifier.

JSON JavaScript Object Notation.

JSON-LD JSON for Linking Data.

KB Knowledge Base.

KG Knowledge Graph.

kNN k-nearest neighbours.

LD Linked Data.

LD4P Linked Data for Production.

LOD Linked Open Data.

LOV Linked Open Vocabularies.

MARC MAchine-Readable Cataloging.

MDW Maximum Degree Weighted.

MEI Music Encoding Initiative.

MIDI Musical Instrument Digital Interface.

MIMO Musical Instruments Museum Online.

MIR Music Information Retrieval.

ML Machine Learning.

MO Music Ontology.

MODS Metadata Object Description Schema.

MoP Medium of Performance.

MPD Million Playlist Dataset.

MSD Million Song Dataset.

MTC MIDI Time Code.

N3 Notation3.

NLP Natural Language Processing.

NPM NPM Package Manager.

ORM Object-relational mapping.

OWL Web Ontology Language.

PCA Principal Component Analysis.

PMO Performed Music Ontology.

PoI Point of Interest.

PP Philharmonie de Paris.

PyPI Python Package Index.

RAMEAU Répertoire d'autorité-matière encyclopédique et alphabétique unifié.

RDF Resource Description Framework.

RDFa Resource Description Framework in Attributes.

ReLu rectified linear unit.

REST Representational State Transfer.

RF Radio France.

RNN Recurrent Neural Network.

RS recommender system.

SEO Search Engine Optimisation.

SERP Search Engine Result Page.

SKOS Simple Knowledge Organization System.

SLAC Symbolic Lyrical Audio Cultural.

SPP Song Position Pointer.

STTL SPARQL Template Transformation Language.

TF-IDF Term frequency-inverse document frequency.

Turtle Terse RDF Triple Language.

UI user interface.

UNIMARC Universal MARC.

URI Uniform Resource Identifier.

UUID universally Unique Identifier.

VIAF Virtual International Authority File.

VU University Vrije Universiteit of Amsterdam.

W3C World Wide Web Consortium.

XML eXtensible Markup Language.

YAML YAML Ain't a Markup Language.

Chapter 1

Introduction

Music is everywhere. Our era opens for us the possibility to access and play music anytime, anywhere, from a multitude of network connected devices. The recent technological progress has deeply changed the music listening experience: in last decade, we moved from local music archives – saved on physical media such as optical discs and MP3 players, which were defining the limits in terms of storage – to potentially endless catalogues belonging to streaming music services, free from the constraints of the supports and dematerialised in computer clouds. In this context, the role of recommender systems in item discovering may be decisive. And consequently, the importance of the data on which those systems are based grows.

Classical music is a niche in the world of streaming music services. This niche actually constitutes a super-genre that groups together a multiplicity of different genres – from Gregorian chant to symphony, from ballet to chamber music – and involves artists who play a greater variety of functions than their colleagues in modern music: composers, conductors, instrumentalists, voices, soloists, members of orchestras, etc.

This thesis manuscript mostly focuses on classical music, researching how to represent and exploit its information. The main goal is the investigation of strategies of knowledge representation and discovery applied to classical music, involving subjects such as Knowledge-Base population, metadata prediction, recommendation systems.

1.1 Motivation

In an interview of 2011 [102], Nolan Gasser from Pandora Radio stated that the challenges of classical music deeply differs from the ones of the *popular music*[1]. Table 1.1 summarises those differences. First, classical music embraces a huge material of centuries, which spans from

[1] With *popular music*, we refers here to all those genres that do not falls under the definitions of classical music, jazz or world music, e.g. pop, rock, hip-hop, funk, rap, electronica, dance.

the Gregorian chant to works written last Tuesday. A first consequence consists of a higher number of musical works among which a recommender system can select relevant items, when compared with the roughly 70 years of popular music history. The physical support (before) and the radio schedule (today) contributed to define the form of the music we listen daily, mostly consisting in songs lasting in average 2 or 3 minutes, in contrast with the long duration of some classic forms, often articulated in parts (movements, acts, scenes). Even the harmonic construction shows differences, with much more complexity and heterogeneity in classical compositions.

	Popular Music	**Classical Music**
Main element	Track (recording)	Work (composition)
Main artist	Performer	Composer
Involved period	70 years	Thousand years
Music forms	Single songs	Multi-movement works
Duration	Few minutes	Up to hours
Modes and keys	Major, minor	Polyphonic, homophonic, monophonic

Table 1.1 – Summary of the differences between popular and classical music

Other studies have as subject the data which represent the music content, which can be expressed with an audio signal (recording) or with any symbolic representation of music, in format of notation (music score) or digital encoding format like Musical Instrument Digital Interface (MIDI) or Music Encoding Initiative (MEI). In this thesis, the focus is put on the data about the music, which follow in the denomination of **metadata**. This group includes both factual information – such as title, composer, composition date – and cultural definitions – genre, emotion, style. Metadata are the driver of Music Information Retrieval (MIR) [30], being often among the inputs or the output of MIR systems. In the same time, metadata are the most used way humans have to access the information, for example searching for a specific artist or song.

The music information can be very complex. Taking as example a well-known masterpiece such as Beethoven's *Moonlight Sonata*, it is possible to describe the music work as composed by the German composer, its score in the handmade original version, in a later transcription, or in the different printed editions, the multiple interpretations by pianists and – in case of arrangement – by other instruments and orchestras. Related to these interpretations, the performances, concerts, recordings, music albums edited on CDs and other media can also be described. A substantial divergence between the world of classical music and the one of popular music consists in the element (composition or recording) that people identify as representative for a specific piece.

The experience of popular music is commonly driven by the **recording**. Our idea *Bohemian Rhapsody* overlaps perfectly the recording of 1975 which brought the song to success. Like-

wise, the Queen are addressed as the artist of the song, assigning to the performer a bigger importance than the composer Freddy Mercury. On classical side, the centre of the experience is the **composition**, while none of the performances can be considered representative. In facts, for classical music the word *interpretation* is widely used, in order to emphasise the performer's artistic choices, which make each performance unique and distinct from the others. Nevertheless, the paternity of the music is always assigned to the composer, and no one would ever claim that *Moonlight Sonata*'s authorship belongs to no one else except Beethoven.

1.1.1 Track-based vs work-based approach

These differences are reflected in the way the music information is managed by stakeholders.

On one side, music steaming services follow a **track-based approach**. This approach see the track as atomic unit, the artist as unique carrier of the authorship, and the presence in the same album as unique possible relationship between tracks. In Spotify[2] or Deezer[3], Beethoven is often not even specified as the "artist" of Moonlight Sonata, while sometimes his name may be displayed near the performer's one, without any distinction between their roles (Figure 1.1). Similarly, the title string contains frequently other kind of information – like opus statement, catalogue number, order number, key or instrument – without following any regularity. In facts, classical pieces do not always have a proper title; the difficulty in naming classical music works is a well-known problem, and often it produces a variety of different titles for the same composition. This leads to a difficulty in searching for precise musical works and identifying tracks which refers to the same composition. In addition, for a classical music lover is generally difficult, if not impossible, searching for works of a given composer, or specific performances of a defined conductor [156].

Music archives and libraries are used, in contrast, to have much more structured information (Figure 1.2). We can talk about a **work-based approach**, which put the work as aggregation unit and entry point, not only for those metadata defined at the composition-level (genre, composition date, key), but also for related publications, performances, recordings, books, etc. Libraries are indeed considered among the most complex and advanced forms of information systems [113].

While the simplified version of the metadata of the track-based approach is sometimes enough for commercial purpose, expressing the whole complexity of the music information opens up new possibilities for advanced search, visualisation of music influences, and for developing new recommendation strategies for musical applications.

[2] https://open.spotify.com/search/songs/moonlight+sonata
[3] https://www.deezer.com/search/moonlight+sonata

Figure 1.1 – Beethoven's Moonlight Sonata in a screenshot from Spotify

1.1.2 Status of Classical music metadata online

Fans of classical music are underrepresented on social media and music streaming platforms [176]. On Spotify, Justin Bieber's monthly listeners outnumber Mozart's ones more than 10 to one[4]. Therefore, those platforms hardly have classical music among their priorities and this produces several consequences, one of the most evident being the correctness and completeness of data.

At the beginning of my PhD, in 2016, we collected few screenshot of Google's Search Engine Result Page (SERP) (Figure 1.3). In that period, the Mountain View company was spreading the presence of informative cards – today called Knowledge Panels[5] – which covered also music. We searched for two totally different music compositions. Contemporary music like Queen's *Bohemian Rhapsody* (a) was enriched with different metadata (artist, album, awards) and with other version of the same song realised by different artists. Instead, a classic masterpieces such as Beethoven's *Moonlight Sonata* (b) displays much poorer information. Moreover, no Knowledge Panel at all was displayed when searching with the original title *"Sonata Quasi una Fantasia"* (c).

[4] Justin Bieber's monthly listeners: 40 023 282 (https://open.spotify.com/artist/1uNFoZAHBGtllmzznpCI3s). Mozart's monthly listeners: 3 873 058 (https://open.spotify.com/artist/4NJhFmfw43RLBLjQvxDuRS). Data updated on 27/05/2019.

[5] source: Google Website https://support.google.com/business/answer/6331288

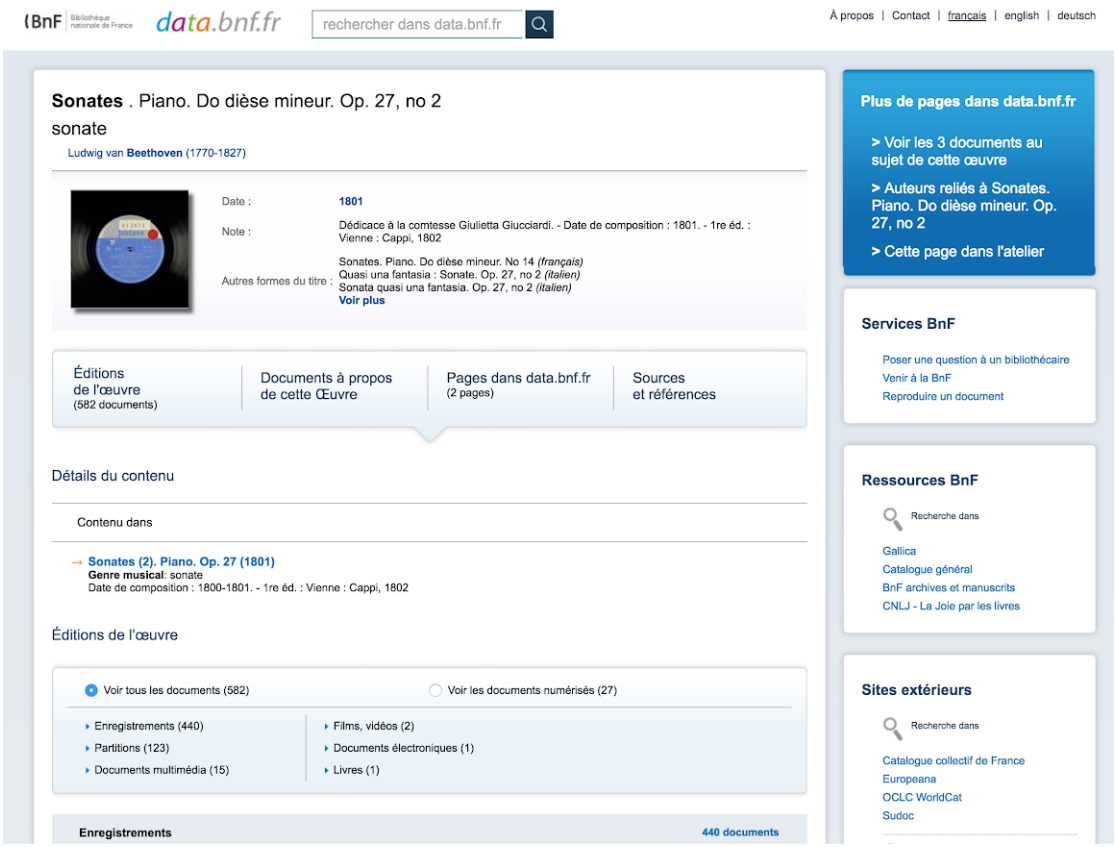


Figure 1.2 – Beethoven's Moonlight Sonata in a screenshot from data.bnf.fr

While this specific example has been fixed in the years, incorrect or poor metadata about classical music are still generally present on search engines. Thus, it can happen to google for the music genre of classic composers and discover that Mozart is a Folk and Pop star (Figure 1.4), Chopin a author of New Age pieces, and Haydn a virtuoso of Dance and Electronic music[6]. Even services like Spotify may present some oddness and assign the genre "german jazz" to Johann Sebastian Bach (Figure 1.5).

Underrepresentation means also less data for the algorithms. The related artists to Johann Sebastian Bach in Spotify[7] and Deezer may be highly influenced by popularity bias, given the presence of some of the most known composer like Beethoven, Mozart, Chopin and Vivaldi. This results do not bode well for any possibility of reaching the composers in the "long tail" [32].

Speaking about research, classical music is popular for topic like automatic music generation [73, 76, 115] and optical music recognition [45, 145]. Instead, research in recommender

[6] Those results are reproducible by searching "beethoven genre" and navigating to the artists "people also search for". https://tinyurl.com/yyrtxo7h

[7] Recently renamed "Fans Also Like".

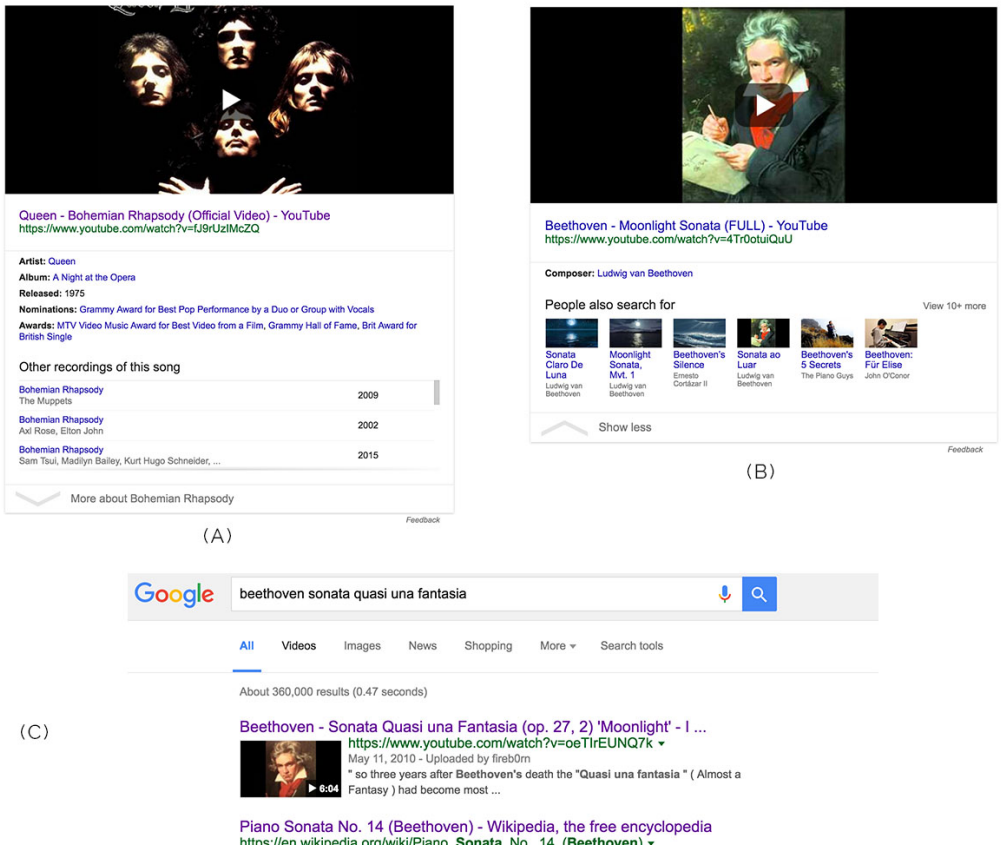


Figure 1.3 – Google Knowledge Panels comparison: (a) result for “Queen Bohemian Rhapsody”, (b) result for “Beethoven Moonlight Sonata”, (c) result for “Beethoven Sonata Quasi una Fantasia”. Screenshots captured in mid-2016.

systems (RS) for classical music is currently at an early stage, paying the lack of dedicated datasets and the incompleteness of metadata in generalist ones. In 15% of tracks involving Bach’s compositions contained in the Million Playlist Dataset (MPD) [33], Bach does not figure as an artist. In Million Song Dataset (MSD), the percentage arise up to 98%. Undoubtedly, Knowledge-based RS would take benefit of "large amounts" of precise ("unambiguous and non-noisy") data [93].

1.1.3 Why recommend classical music

The consumption of music overcome the one of any other media, including TV, books and media [165]. It is a consequence that research on music RS can have a direct impact on people’s everyday life.

The goal of RS is the improvement of users’ listening experience. A good RS help the user in selecting relevant music following his/her taste, balancing previously listened tracks and new

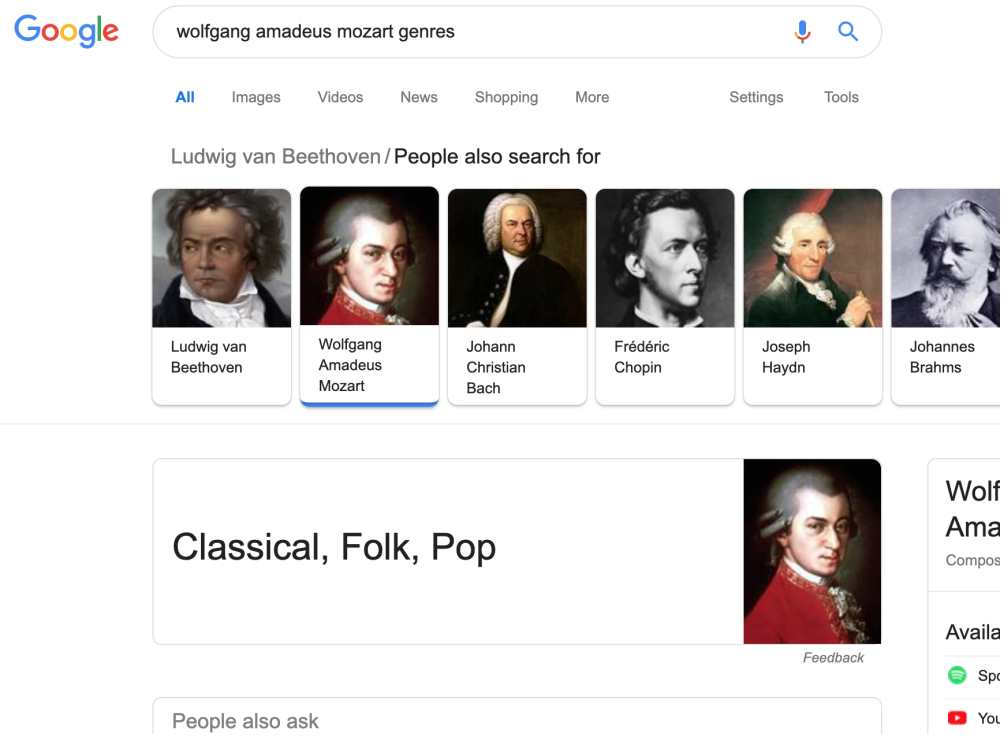


Figure 1.4 – Google Knowledge Panel about music composers and genres. Screenshot captured in May 2019.

proposals to discover [183]. This is also valid for classical music.

Collaborating with different kind of music institutions, we identified at least other three use cases which could take benefit of RS, and in particular applied to classical music: concert programming, radio broadcasting services, editorial playlist producing. The ones currently in charge of producing lists of music for those targets can be helped by automatic selection of tracks made by RS. This human-machine collaboration may ease the task and improve results, for example inducing the inclusion of unknown compositions.

1.2 Research context: the DOREMUS project

This research work has been carried in the motivating and inspiring context of the DOREMUS project [2]. DOREMUS is a French research project with three main declared purposes[8]:

- improve music description to foster music exchange and reuse;
- travel to the heart of the musical archives in France's greatest institutions;
- connect sources, multiply usage, enrich user experience.

[8] source: http://www.doremus.org/

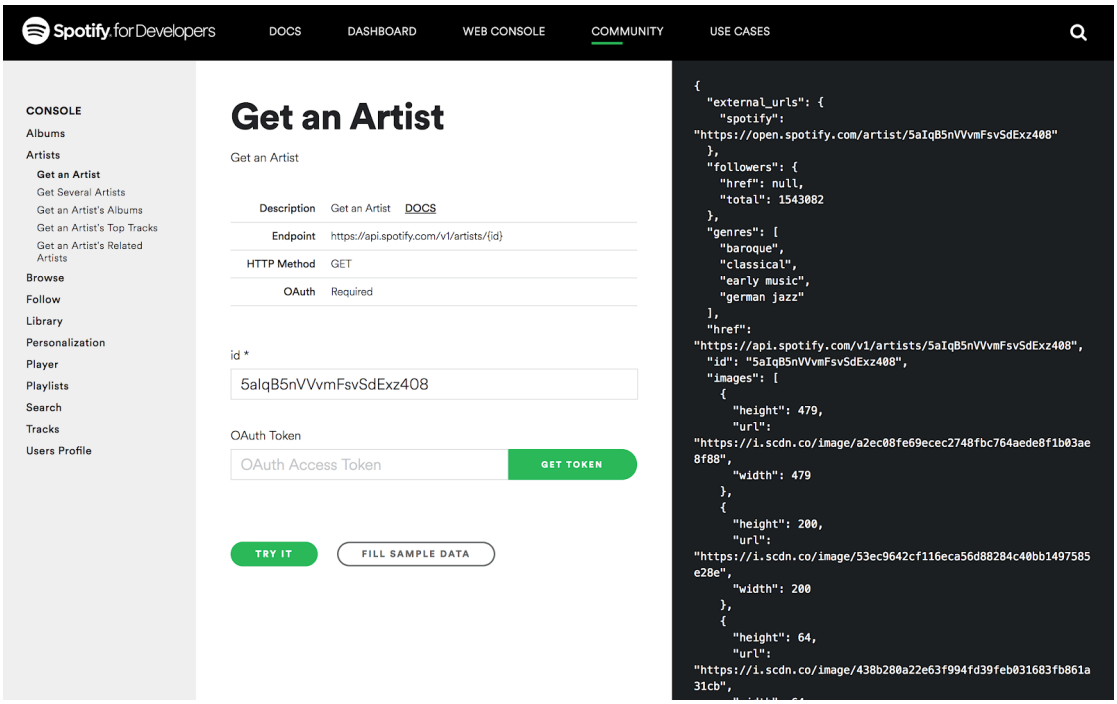


Figure 1.5 – The Spotify Developer tool page about Johann Sebastian Bach. Among genres, a “german jazz" value stands out. Screenshot captured in January 2019. The data have been corrected in the meanwhile.

The projects involved three main French cultural institution, the Bibliothèque nationale de France (BnF)[9], the concert hall Philharmonie de Paris (PP), and the public broadcaster Radio France (RF). These institution provided access to their music archives – with the aim of publishing them in the Web of Data – and made their expertise available for the project goals. The consortium was that completed by two industrial members – Ourouk and Meaning Engines – and three academic members, namely the laboratory GERiiCO of the Université de Lille – in charge of social research about music data usage –, the laboratory LIRMM of the Université de Montpellier – in charge of data interlinking –, and EURECOM, which contribution in the fields of data generation, data access and artificial intelligence is described in this thesis.

The project started in late 2014. When my research work started in April 2016, the research directions of the projects were settled and some work was already started. In particular, the foundation of the DOREMUS ontology was already defined, so that it could be published in [37, 38, 39, 51]. Some investigation in the social usages of music had been conducted [50]. A germinal version of a data converter allowed us to have a preliminary version of the data and to start immediately the work on data visualisation, allowing us to present a demo few month later [104]. The project officially ended in September 2018.

[9] *fr:* French national library.

The perfect overlap between the project goals and my thesis ones, the friendly environment, the challenges and ambitious goals that were assigned to EURECOM and me in particular, the opportunities of exchange with people with different background and professionalism, all those elements turned out to be a fruitful context for the development of my doctoral formation and the development of this thesis.

1.3 Research Questions

As seen before, libraries contain detailed information about artists, works, performances, scores and recordings. Representing the relations between those elements may result in a quite complex structure. This information is normally encoded in several different formats, often relying on different cataloguing practices and making use of different naming conventions. This forbids interoperability, accessibility, knowledge extraction.

- *Which model can be successfully applied for better representing those data?*
- *To which questions should this model be able to give an answer, in order to give benefit to final users and music scholars?*
- *Which strategies should be applied for leveraging them in a Knowledge Base (KB) using this model?*
- *How to make those data more accessible to researchers and developers?*

The answers to those questions can enable further research, which can exploit this classical music metadata.

- *How graph-based algorithms can support recommender systems, involving knowledge representation in the process?*
- *Which information is possible to extract from editorial playlists?*
- *How is final-user consumption impacted by music specialised KB?*
- *Apart from metadata, is graph representation suitable for representing also the music content?*

1.4 Summary of contributions

This work contributed to the research with the following outcomes:

- a model and a set of controlled vocabularies – realised thanks to the expertise of the cultural institutions – for describing music in detail;
- a Knowledge Graph (KG) focusing on classical music and containing data about artists, works, performances, scores and recordings. The graph, realised through Semantic Web technologies and published in the Web of Data, gives access to the fine-grained

metadata coming from the most important French cultural institution;
- a set of tools for converting data, creating an Application Programming Interface (API) on top of the SPARQL endpoint, visualising and exploring the data;
- approaches based on embeddings computed on structured metadata, titles, and symbolic music for ranking and recommending music;
- some demo applications which exploit the previous approaches and resources.

1.5 Thesis outline

The remainder of this thesis is organised in two main parts.

Part I is dedicated on the realisation of the DOREMUS graph, including the applied extract, transform, load (ETL) strategies. After having described some related work in Chapter 2, Chapter 3 and Chapter 4 introduces respectively the DOREMUS model and the controlled vocabularies. The data conversion and the output KG are detailed in Chapter 5. Chapter 6 focus on the access to the data, presenting SPARQL Transformer.

In **Part II** are detailed methods and applications for exploiting the music graph. In Chapter 7 is presented the state of the art, including some algorithms that are applied in following chapters. A strategy for computing music embeddings on the music graph is described in Chapter 8, while the use of the embeddings for playlist completion is discussed in Chapter 9. Three applications developed for let the final user to explore the DOREMUS data are introduced in Chapter 10. Chapter 11 reports a first experiment for generate graph embeddings on MIDI files, successfully applied to metadata prediction.

Finally, some conclusions and perspectives are outlined in Chapter 12. At the end of the manuscript, the reader will be able to find the list of papers published in the context of this thesis.

Part I

Building a Music Graph

A research work can hardly be carried on without data. An important part of this thesis has been dedicated on the realisation of the DOREMUS Knowledge Graph (KG), the biggest available dataset specialised in classical music metadata. Even if the KG has been realised by a team composed by different professionalism, I had the chance to be strongly involved in most of the steps of its realisation, being responsible of the conversion pipeline and of the management of the triplestore.

In this part, we discuss all the steps that brought from source data to their consumption. This work is also at the base of the research described in the second part of this manuscript.

Chapter 2

Related Work

This chapter introduces some related work about music knowledge representation and access in the Web of Data. Section 2.1 consists in a short introduction to Semantic Web. Then, we will focus on how Semantic Web technologies can be applied for representing knowledge, for example the information about music (Section 2.2) and librarian data in general (Section 2.3). Finally, some solutions for an easy access to this knowledge is reported in Section 2.4.

2.1 Knowledge Representation in the Semantic Web

Semantic Web technologies emerged in the field of data management with the ambitious promise to realise the *Web of Data* [19], which nowadays consists of a growing set of interconnected datasets representing different parts of human knowledge. Some of these datasets are specialised in a well defined kind of objects – e.g. *GeoNames* contains geographical places [200] – or field – e.g. *3cixty* collects data about what to do in a city [191] – while some others have a more generalist and encyclopedic aim like *DBpedia* [11].

The building blocks of these dataset are **triples** of the form "subject-predicate-object", following the Resource Description Framework (RDF) data model [149] proposed by the World Wide Web Consortium (W3C). Each term in the triplet is identified by a Uniform Resource Identifier (URI), so that it automatically receives also an address in the Web which can be accessed by both machines and humans with a Web browser for retrieving information about the term itself in a suitable format. Apart from the HyperText Markup Language (HTML) format which targets human consumers, standard formats includes eXtensible Markup Language (XML), JavaScript Object Notation (JSON), Comma-separated values (CSV), next to more specialised format like RDF-XML, Terse RDF Triple Language (Turtle) or Notation3 (N3). Finally, a format called JSON for Linking Data (JSON-LD) extend the JSON syntax in order to enable the semantic representation of entities[1].

[1] A quite complete overview on the different format is available at https://medium.com/wallscope/

For defining how concepts in a specific domain interact each other, the Semantic Web relies on the definition of ontologies. An **ontology** is a data model that aims to represent the knowledge in a given domain (i.e. geography, literature, music), explicitly defining how entities are in relation each others. In Semantic Web, ontologies are normally intended to be adopted, re-used, and eventually extended by other people in the community, thus they may include human-readable labels, descriptions, and notes. Languages such as the *Web Ontology Language (OWL)* [117] allows to define classes and properties, specify domains, ranges and constraints, enabling automated reasoning on the data. W3C's *Simple Knowledge Organization System (SKOS)* [129] is widely used instead for representing vocabularies and taxonomies. SKOS allows to specify preferred and alternate labels, definitions, conceptual hierarchies and relationships.

RDF gives to the data the shape of a graph: the subjects and the objects (resources) are the nodes, while the edges are the predicates which link the resources. In recent years, the term *KB* has become popular in reference to dataset following semantic web paradigms, in order to remark their suitability to feed automatic learning algorithm that exploit the contained information, while the term *KG* has been used in reference to the graph shape[2].

2.2 Music Ontologies in the literature

Different models and ontologies have been so far proposed for representing the music information with Semantic Web Technologies.

An important role as conceptual foundation for many music and – in general – cultural ontologies is hold by the **Functional Requirements for Bibliographic Records (FRBR)**. Published by the International Federation of Library Associations and Institutions (IFLA) for the first time in 1998, this schema defines four distinct states in which a generic cultural object can exists: the *Work* – intended as the artistic or intellectual idea and aim – is realised through a specific set of choices in the content to which we refers as *Expression*; this one comes in the reality in a physical shape, the *Manifestation*, which can be produced in one or more single *Items* (Figure 2.1). For example, Victor Hugo's story of an hunchback bell-ringer of Notre-Dame Cathedral (*Work*) is formalised in a specific choice in the words which compose *Notre-Dame de Paris* book (*Expression*), which has been published in different editions (*Manifestation*) with a certain number of copies (*Items*).

Among the music models relying on FRBR, the **Music Ontology (MO)** [161] is the most known one in the community. This ontology extends the Timeline Ontology [160] and the Event

understanding-linked-data-formats-rdf-xml-vs-turtle-vs-n-triples-eb931dbe9827

[2] A nice definition of Knowledge Graph is contained in this blog post by Jo Stichbury: https://hackernoon.com/wtf-is-a-knowledge-graph-a16603a1a25f

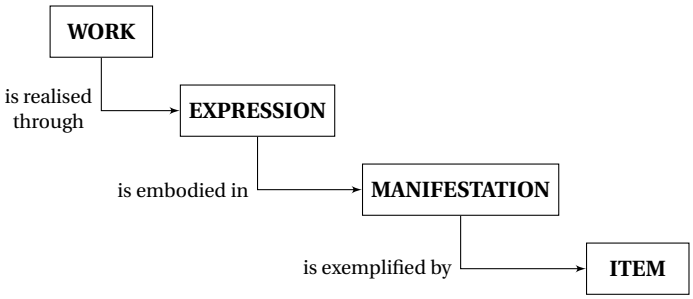


Figure 2.1 – FRBR diagram

Ontology [159], providing a set of music-specific classes and properties for describing musical works, performances and tracks, together with fragments of them. The authors foresee the use of taxonomies and vocabularies for populating the values of certain properties, like keys, instruments and genres. The ontology have been in the years extended with other modules, like the Audio Effects Ontology (AUFX-O) [201] and the Audio Features Ontology [9]. Several examples of interconnecting MO to other datasets, whether they describe music or other kind of data (i.e. DBpedia) are shown in [162]. Even with some attention to classical music – visible in classes and properties like composition, catalogue number, arrangement– MO reveal a strong connection with the track-based vision of the music. Some relevant absences which can confirm this statement are alternative roles in composition rather than the composer, alternative titles with specific properties (original title, given title, translation), details in the number of foreseen instruments, connections between performers and instruments in a performance. Beside the simplicity of adopting the model, MO is quite far from being able to represent the information coming from specialised classical music archives.

The **Performed Music Ontology (PMO)**[3], a specialised ontology for metadata related to music performances, has been developed in the context of the Linked Data for Production (LD4P) project [179]. The ontology extends the BIBFRAME librarian model and includes few specialised vocabularies. The **Context-based Music Recommendation Ontology (COMUS)** [182] is a specialised ontology for representing music preferences of persons in different situations, in order to define a dataset for music recommendation. The COMUS Ontology includes classes and properties for representing both the user preferences and the music metadata, even not reaching the expressiveness level of MO.

Other works focus instead on the music content itself, rather than on the music metadata. An attempt to represent the whole music theory fundamentals lead to the development of the **Music Theory Ontology** [163], with the final goal of computing analysis and inference by relying on the music rules. The information contained in a music score is instead represented in the **MusicNote Ontology** [36], enabling the detection of four kind of dissonance in Renaissance

[3] http://performedmusicontology.org/

pieces.

2.2.1 Schema.org

By the initiative of some of the most popular search engines in the world, a vocabulary for structured data for web pages has been created with the name of **Schema.org**[4] [71]. The aim of this community project is to enable search engines to have a semantic understanding of the content in the web page, in the context of Search Engine Optimisation (SEO). In order to get this, web pages should refer to the vocabulary through specific formats like Microdata, Resource Description Framework in Attributes (RDFa) or JSON-LD. The SERP take benefit of the presence of this metadata in the web pages, being able to provide better results and display informative snippets. Schema.org can nowadays be considered de facto standard in Structured Data for the Web.

Schema.org provides classes for describing persons, organisations, places, products, etc. It also includes classes like `CreativeWork`, `Event`, and their subclasses; among them, there are some music-specific classes: `MusicComposition`, `MusicRecording`, `MusicEvent`, `MusicGroup`, etc.

In the past, the community group took under consideration the possibility of modelling the structure of the `CreativeWork` class on the basis of FRBR, until their final resolution of avoiding it. The reason of this decision can be found in the difference of purpose of the two ontologies: FRBR is specific for describing cultural works and expressions, while Schema.org provides a way to markup a web page so that search engine can understand and use their content. According to Richard Wallis, Chair of the W3C *Schema Bib Extend working group*,

> *replicating the FRBR rules within the [generic] Schema.org vocabulary was much discussed in the Schema Bib Extend Community Group [...] It was concluded that reproducing those rules would be too complex.*
>
> — Richard Wallis, 2016[5]

Nogales et al. [135] explored a mapping between Schema.org terms and the vocabularies collected by the Linked Open Vocabularies (LOV) project [195]. This mapping has been realized in two steps: firstly matching classes which have exactly the same name; then, taking in account the mapped classes, matching their properties in the same way. The mapping has been developed in [136] through the use of dictionaries: a mapping exists if names of the classes involved are exactly the same or have a synonym in common. An additional manual check confirm the goodness of the match.

[4] http://schema.org/

[5] Source: https://lists.w3.org/Archives/Public/public-schemaorg/2016Feb/0024.html

In [67] a solution is introduced for expressing FRBR entities using concepts that belongs to Schema.org. This work proposes to map each level of the chain <Work - Expression - Manifestation - Item> into a entity of the `CreativeWork` class, keeping the information distributed among different objects. If the goal is improving the SEO, we consider this solution inadequate: when one searches for Beethoven's *Moonlight Sonata*, the target is the piece in its completeness, without accounting any distinction between work and expression.

2.3 Digital Libraries in the Semantic Web

Semantic Web technologies have a strong predisposition for representing the human knowledge, making it open and accessible for public consumption, and enabling connections between datasets. This predisposition has fed in last decade a new attitude for sharing the knowledge beyond the institutional and national borders, embodied by international consortia like the International Association of Musical Libraries (IAML) or in projects like Europeana [78], OpenGlam [59], and datos.bne.es [198]. The benefits that Semantic Web can offer to Digital Libraries (DL) have been reported by several works [100, 198], among which the most influential is the study made by W3C Library Linked Data Incubator Group in [189], and can be summarised as follows:

- it provides methods and standards for integrating different metadata sources, like bibliographic, controlled vocabulary, annotations and non-library sources such as Wikipedia, GeoNames, MusicBrainz, and others;
- it offers solution for interoperability among cultural institutions, promoting the re-use of resources through shared identifiers (URIs) and fostering interdisciplinary research;
- it triggers the passage from specific data structures to models whose durability and robustness is ensured by the semantic description of classes and relations;
- it increases the visibility of cultural data on the Web;
- it encourages a discovery approach of cultural information based on navigation on links ("following one's nose");
- going beyond library-specific formats, it opens the librarian knowledge to developers, researches and other communities;
- it enables advanced use of librarian knowledge, including smart search, reasoning and artificial intelligence (AI) applications.

Accordingly, Semantic Web technologies have gained a central role also on the music domain, that has reached the Linked Open Data (LOD) world. In [12], a traditional music DL environment is developed through the conversion of metadata in RDF and its enrichment through linking to external Linked Data (LD) resources, although the elements in the resulting graph continue to be conceived as separate records instead of interconnected nodes.

Different experiences about converting data from the librarian format MARC to RDF have been explored[6]. The *datos.bne.es* project has developed MARiMbA [198], a software for the conversion of MARC data from the Spanish National Library in RDF, using the FRBR model. Moreover, it manages also the following steps: interlink data, load data in a triple store and provide a simple visualisation of the data.

The need for harmonisation of musical metadata coming from different sources and formats led to different technical solutions, often making use of Semantic Web technologies. Among them, the **Distributed Metadata Service (DMS)** is a service that stands between the data and the consumers and that performs real-time conversion of each query to source-specific queries, the consequent conversion of each result in a common format and their combination, without needs for pre-processing [101]. In some cases, this approach can be impossible to realise because the structure of certain documents is not suitable for different kinds of queries. Another strategy relies on converter tools based on static mapping. This strategy often foresees an alignment to be performed after the conversion, for discovering co-references between sources, like in the **musicSpace** project [26].

The **Transforming Musicology** project created **InConcert** [140], a RDF dataset of performance metadata collected from concert ephemera, such as programmes, reviews, adverts, etc. The dataset has been created by converting and connecting data sources in other formats, using generic tools like Karma[7] [94] and D2RQ[8] [20], which perform the alignment to the chosen ontology (MO). A similar workflow made possible the creation of the **JazzCats** dataset[9], specialised in jazz performances [137, 139]. The full workflow – common to the two projects – is described in [138].

Other semantic music libraries that is worth to mention are the **MIDI Linked Data Cloud** [121, 123], a big archive of MIDI information represented in RDF, and the **Listening Experience Database (LED)** [6], a KB of annotations about music listening. A more complete list about music datasets on the web has been collected in **Musical Data on the Web (musoW)**[10] [49].

2.4 Data access

The access to RDF data by data consumer is a central discussion topic in the community. Among the most popular solutions, the **Linked Data Fragment Server** [197] offers a trade-off between the costs of servers for live querying the SPARQL endpoints and the costs of

[6] https://github.com/search?q=marc2rdf
[7] http://usc-isi-i2.github.io/karma/
[8] http://d2rq.org/
[9] http://jazzcats.cdhr.anu.edu.au/
[10] Also the DOREMUS dataset appears in the survey under the misspelled name *DoReMus*.
http://data.open.ac.uk/page/musow/e896e4ddd22820fd73cfa7b3c3535ec9

clients for downloading entire data dumps. A LD Fragment Server is able to return *fragments*, the collection of all the triples that match a certain triple pattern `?subject ?predicate ?object`. The LD Fragment Client is then in charge of solving more complex queries by merging and filtering the different fragments.

A large amount of Semantic Web literature [62,151] tries to give general responses to ad hoc API services, developed for individual applications or projects as a bridge between the endpoints and the developers. Works like [184] and the **smartAPI** [206] provides approaches for using LD on top of Representational State Transfer (REST) APIs, describing the Web services with the RDF. In this context we are instead interested in the opposite case: the manipulation of the RDF via Web services. "The use of HTTP for accessing, updating, creating and deleting resources from servers that expose their resources as Linked Data"[11] is regulated by the LD API specification[12] and the W3C Linked Data Platform 1.0 specification. The OpenPHACTS Discovery Platform for pharmacological data [69], LDtogo [141] and the BASIL server [47] use SPARQL as an underlying mechanism to implement APIs and provide LD query results.

Influenced by these works, `grlc` [122] decouples query storage from API implementations by leveraging queries uniquely and globally identified by stable and de-referenceable URIs, automating the query construction process. The software generates automatically Web APIs from SPARQL queries contained in GitHub repositories. Moreover, it includes Swagger[13] for generating a user interface (UI) which document the API and enable to easily test it.

The **W3C RDFJS Community Group**[14] is heavily contributing to the effort of offering a tool to JavaScript developers for using RDF data. The major outcome of the initiative is a low-level interface specification for the interoperability of RDF data in JavaScript environments [16]. RDFJS brings the graph-oriented model of RDF into the browser, allowing developers to directly manipulate triples.

Recent works realised an interoperability between the GraphQL language[15] and RDF, performing in this way a conversion in JSON of the data in an endpoint [187]. The same syntax of GraphQL allows to produce a JSON object with different levels of nested nodes. Some of these solutions rely on automatic mappings of variables to property names (Stardog[16]), while others rely on a schema (HyperGraphQL[17]) or a context (GraphQL-LD [186]) which the developer is in charge to provide. None of those approaches implements any strategy for detecting and merging bindings referring to the same entity.

[11] https://www.w3.org/TR/2015/REC-ldp-20150226/
[12] https://github.com/UKGovLD/linked-data-api
[13] https://swagger.io/
[14] https://www.w3.org/community/rdfjs/
[15] https://graphql.github.io/
[16] https://www.stardog.com/
[17] https://www.hypergraphql.org

2.5 Conclusion

After introducing briefly the fundamentals of Semantic Web technologies and ontology design for the layperson in the subject, this chapter has summarised some of the literature related to three different research areas, that next chapters will further study and apply to the classical music domain.

Several ontologies are available for describing music, among which it is possible to find models that specialise in music content representation, or in a particular *state* of the music piece (i.e. the performance), or in fulfilling a given goal (i.e. recommendation). The Music Ontology is the state-of-art for representing the metadata related to popular music; nevertheless its design hinders more complex structures, like the ones of classical music as described in librarian archives.

Different studies and experiences have revealed the benefits of realising Digital Libraries using Semantic Web technologies. Multiple datasets have been released during the years for sharing the information about music, realised with different processes and strategies according to the occurring challenges.

The attention of the Semantic Web community to the re-use of data by data consumers is gaining strength. Different techniques and tools have been proposed so far for solving some of the issues about the access to Semantic Web data, still leaving some open problems.

The work reported in the following of this part is giving a contribution to these research fields, investigating the best solutions for our goals.

Chapter 3

A Music Model

As discussed in Section 2.2, the models and ontologies currently available are not capable of fully representing the music information contained in big music archives. The project context requires an expressive ontology capable of richly describing the music information coming from different stakeholders – conservatories, concert halls, musicologists, libraries, musical museums, radios – and reflecting the vision of each of them on the music object. For this reason, we collected a set of questions in human language (French and English), requiring that the model was able to answer them.

This chapter introduce the DOREMUS ontology, a model for the description of music catalogues, result of the joint efforts of the members of the DOREMUS project, including the author of this manuscript. In addition, I contributed to the model creation as technical reference point, making the modelling group aware of logical and coding constraints and taking care that the model was compatible with the following information extraction goals of the project.

The DOREMUS model is detailed in Section 3.1, while Section 3.2 presents an approach for mapping it to the Schema.org vocabulary. The evaluation of the model through question answering is contained in Section 3.3. Finally in Section 3.4 there are some conclusive statements and future work intentions.

3.1 The DOREMUS ontology

The DOREMUS model is built upon **FRBR-object oriented (FRBRoo)** [55], an ontology for representing cultural objects which has in turn been born as a dialog of the librarian FRBR model (mentioned in Section 2.2) and the CIDOC Conceptual Reference Model (CIDOC-CRM) [54]. We already discussed about the former in Section 2.2. CIDOC-CRM is an ontology developed for the museum domain. One of its main characteristics is the importance given to events: no objects can exists without a specific creation event, and events are required for specifying the

object location in a museum or describing its appearance through observation. CIDOC-CRM popularity is proved also by an important number of extension of its core ontology [132]. The harmonisation of FRBR and CIDOC-CRM gives birth to the Work-Expression-Event triplet[1] pattern of FRBRoo (Figure 3.1): the abstract intention of the author (Work) exists only through an Event (i.e. the composition) that realises it in a distinct series of choices called Expression(s). Thinking as example to the book *Moby Dick*, the artistic object takes birth when the idea (Work) of the author Melville are written (Event) in the succession of words (Expression). The relations between these classes and the relative subclasses represent one of the strength of the model thanks to the wide expressiveness gained from this. In FRBRoo, one can link a work with another one (a specific critic edition or the French translation), add more details about the creation event (where and when it took place), add derivatives works (the 1956's movie *Moby Dick*) or works that are components of a complex one (the critics essays contained in a particular edition).

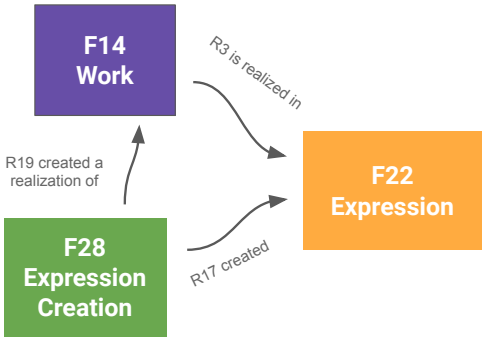


Figure 3.1 – The triplet pattern in FRBRoo

The choice of extending FRBRoo relies on different motivations.

- **It is a librarian model.** Being popular in librarian archives, FRBRoo appears familiar to cataloguers and fits well with other kind of contents.
- **It is a bridge to other cultural objects.** The model is ready to be used for describing the interconnection of different arts. FRBRoo provides properties for linking a work such as a musical piece with the poetry that has been adapted in the lyrics or with the film having it in its soundtrack.
- **All triplets are optional.** The Work-Expression-Event pattern ensures that each step of the life of a musical work can be modelled separately, following the same triplet structure. Thinking about a classic work, we will have a triplet for the composition, one for any performance event, one for every manifestation (e.g., the score), all connected in the graph. Each triplet contains information that at the same time can live autonomously

[1] Not to be confused with an RDF triple.

and be linked to the other entities. This provides the freedom of representing, for example, a jazz improvisation as extemporaneous performance not connected to a particular pre-existing work, or to collect all the recordings of a piece of world music.

- **The event expressivity.** In FRBRoo, the creation of a work (physical or performative) can be modelled as a unique event, which in turn is composed of a series of different activities, each one carried by a specific person. In our case , this way of representing the creative process matches perfectly music performances – in which every musician give a distinct contribution to the sound – or music composition – in which for example we can separate the work of the composer and the lyricist.

On top of the FRBRoo original classes and properties, the **DOREMUS Ontology** provides specific ones in order to describe aspects of a work that are related to music, such as the musical key, the genre, the tempo, the Medium of Performance (MoP)[2], etc. [37, 38, 39]. Each part of the music production process is considered as an Event that gives birth to a new Work and a new Expression: this leads to the creation of classes like Performance Work or Recording Expression. For the description of music-specific concepts like the key, the genre or the MoP, we publish controlled vocabularies, realised and enriched by an editorial process that involved also librarians, in order to overcome multilingualism and alternative names issues. The vocabularies would be further described in Chapter 4.

The graph depicted in Figure 3.2 shows a real example from our data: Beethoven's *Sonata for piano and cello n.1*. [3] The FRBRoo triplet contains all the information about the work and its composition. Then, the information about the performance and publication are linked to the triplet through specific properties. The nodes represented as circles normally take the form of URIs taken from controlled vocabularies (the function "composer" or the genre "sonata") or are entities that may be matched to external datasets (the person of Beethoven or the places Berlin and Vienna), that can have alternative labels (i.e. in different languages) and additional information. Each one of these nodes represents a link between different works, performances, etc., making everything connected in a large graph. Following the naming convention coming from the extended ontologies, DOREMUS classes and properties are introduced by a uppercase *M* (for classes) or *U* (for properties), together with a incremental number and a human-readable label, like in *M2 Opus Statement* and *U12 has genre*, in the same way classes and properties are introduced by *E* and *P* in CIDOC-CRM, and by *F* and *R* in FRBRoo.

We point out the modelling of the casting as a positive example of the expressiveness of the model that allows to declare all the MoPs required for a particular work and, for each of them,

[2] For MoP we intend any source that can produce a sounds, targeting both instruments and voices. More details can be found in Chapter 4.

[3] http://data.doremus.org/expression/614925f2-1da7-39c1-8fb7-4866b1d39fc7

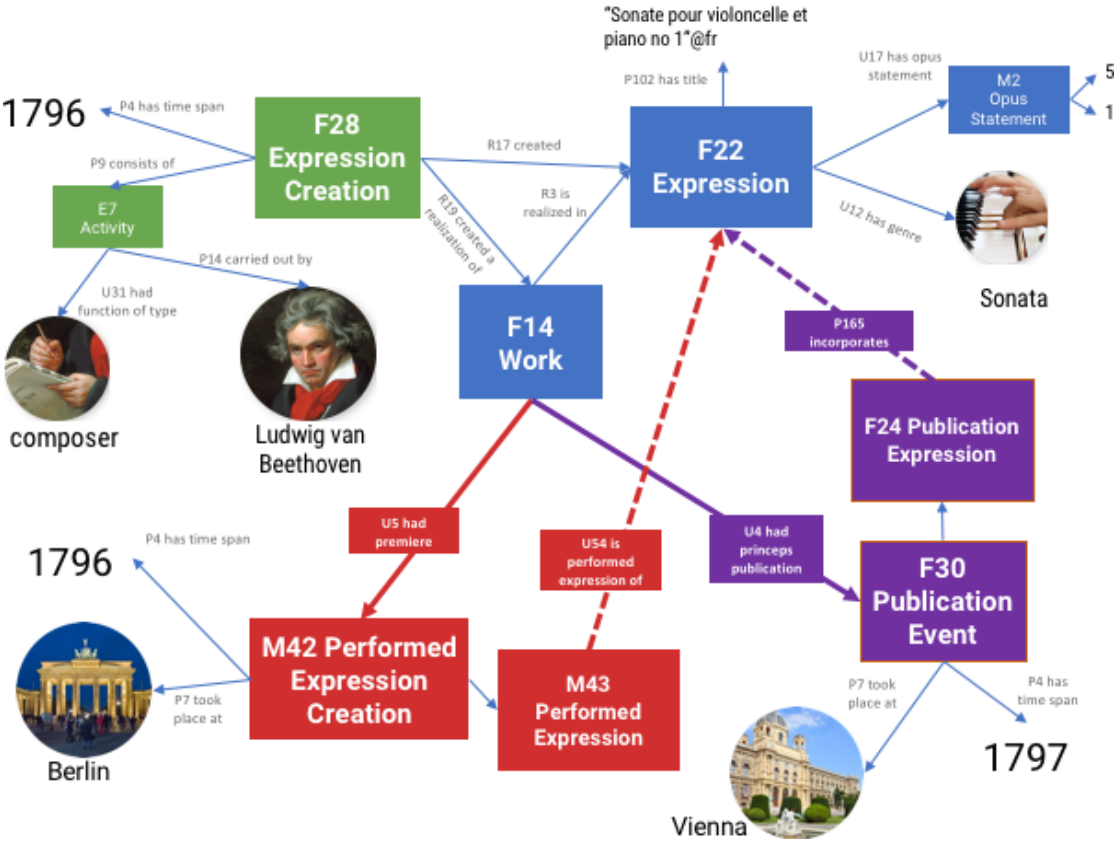


Figure 3.2 – The DOREMUS model: full schema representing the information related to Beethoven's *Sonata for cello and piano n.1*. The music work and expression (in blue) are linked to the entities representing the composition event (in green) the first performance (in red) and the first publication (in purple).

declare the foreseen quantity, the eventual responsibility of soloist for some of them, the interpreted role (for operas), etc. Figure 3.3 contains another example schema representing a performance.

The OWL implementation of the DOREMUS ontology and the documentation are available at http://data.doremus.org/ontology.

3.2 Mapping to Schema.org

The expressiveness of the DOREMUS ontology is counterbalanced by a certain complexity, which can make the model hard to consume by external applications. The need to have a simpler version of the DOREMUS data motivates the research of mapping of the ontology into simpler models. The choice falls on Schema.org because of its popularity in the community, its contribution in SERP optimisation and the presence of some classes for representing music.

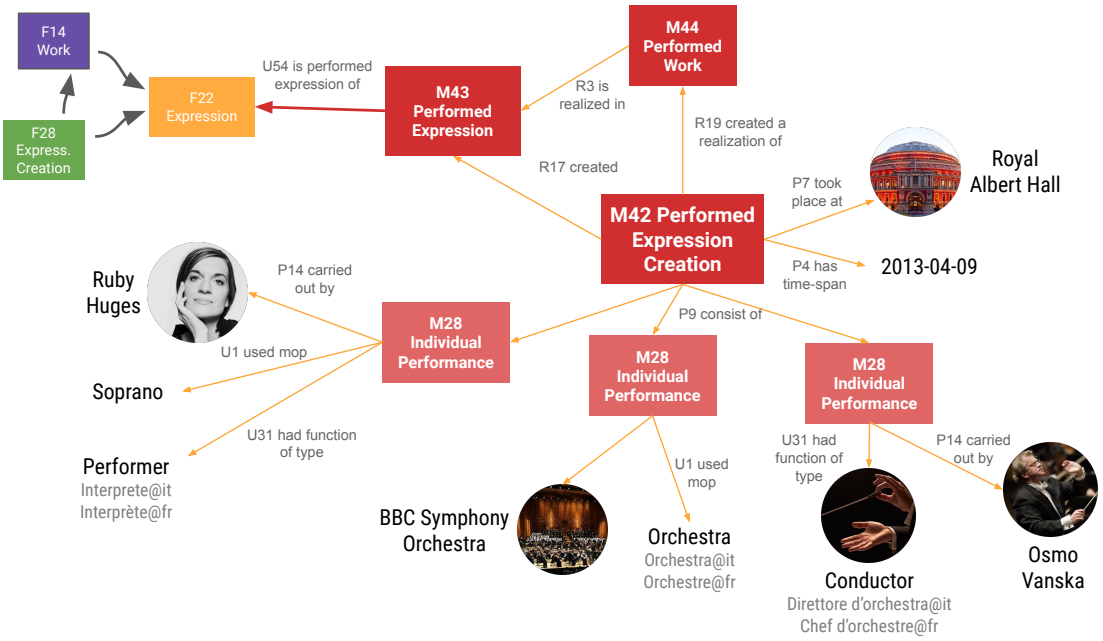


Figure 3.3 – The DOREMUS model: a performance

Starting from the example of Beethoven's *Sonata "Quasi una Fantasia"* expressed with the DOREMUS model (Figure 3.4), we aim to represent the same information using Schema.org. We expect to map the nodes marked in gray with a literal values, while yellow and green nodes with classes.

The mapping approaches mentioned in Section 2.2.1 were not suiting the case of DOREMUS or FRBRoo. With few exceptions[4], the involved ontology and Schema.org have no classes with exactly the same name. Also, matching similar names could be wrong: the DOREMUS `F1 Work` and `CreativeWork` (in its subclass `MusicComposition`) match if we consider the names, but some properties that belong to the latter (like title/name, [musical]Key and genre) are not attached to `F1 Work`, but to `F2 Expression`.

For this reason, we developed a novel method for passing from a complex ontology (DOREMUS) to a simpler one (Schema.org). This method has been presented for the first time in [108] and is based on the observation of the graph. The main idea is to identify a suitable starting node and progress following the links until the graph borders. This method assumes a sufficient knowledge of the models that are going to be mapped, and is structured as a series of recipes to follow. Even is explicitly developed for this purpose, the method is general and can be easily applied to other models. For the sake of simplicity, we refer to DOREMUS or FRBRoo classes and properties using the prefix `mus` (e.g. `mus:F1 Work`) and to Schema.org ones with `sdo` (e.g. `sdo:CreativeWork`).

[4] e.g. `Event`, `PublicationEvent`

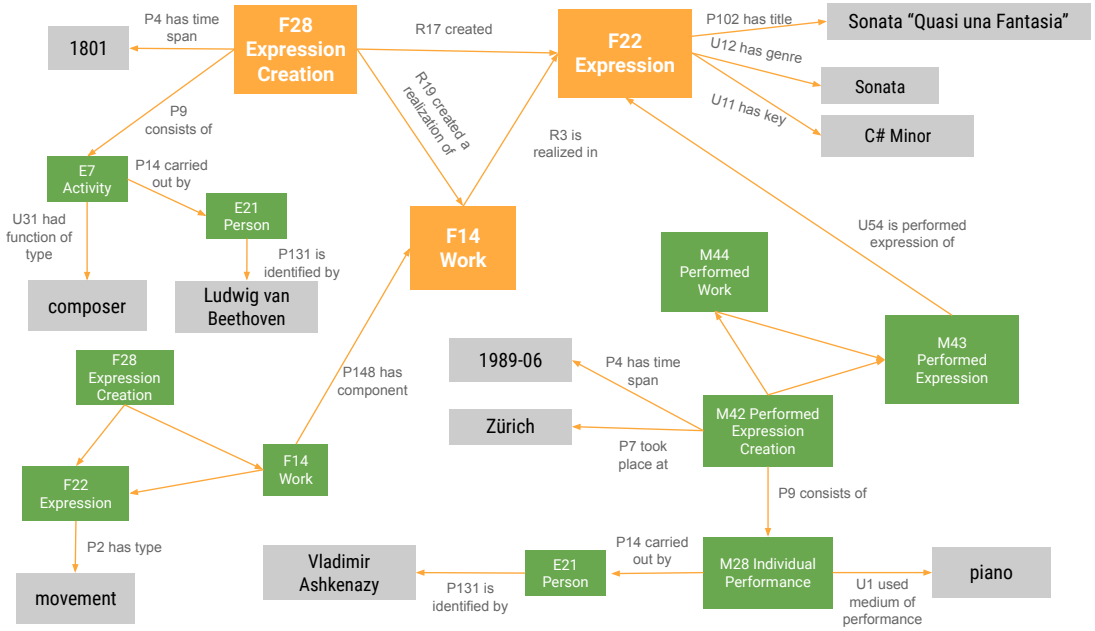


Figure 3.4 – The DOREMUS model: Beethoven's *Sonata "Quasi una Fantasia"*.

3.2.1 Choose the starting node

The most suitable starting point should coincide with the most significant class or group of classes in the starting ontology, DOREMUS. There are different way for evaluate it. As an example, it could be the class with the highest number of occurrences, as this can be evaluated by tools like Loupe [126], or by aggregating different metrics like in [150]. Another strategy could rely on the recognition of a frequent pattern in the ontology, like the Work-Expression-Event triangle in FRBRoo.

As a consequence, the choice will consider what we expect that people are going to search, reasonably textual queries like "Sonata Quasi una Fantasia" or "Beethoven Sonata Quasi una Fantasia". According to this, information about title and author gains a key role. We choose as starting node `mus:F2 Expression` that has the properties `mus:P102 has title`, and `mus:F28 Expression Creation` because of its link with the information about the composer.

3.2.2 Identify similar classes

We start to match the classes, starting form the ones we identified in the previous step. For each class in the source model (DOREMUS), we search the best class that can represent it in the target model (Schema.org), trying to respect one or more of these criteria:

1. They should have similar name.
   e.g. *mus:F28 Expression Creation → sdo:CreateAction*.
2. They should have similar description
3. They should have similar properties
   e.g. *mus:F2 Expression U11 has key → sdo:MusicComposition.musicalKey*
4. They should have similar properties value expected
   e.g. *mus:F2 Expression U12 has genre* and *sdo:MusicComposition.musicCompositionForm* have both "sonata" as possible value

The matches that better satisfy the criteria are: *mus:F28 Expression Creation → sdo:CreateAction* and *mus:F2 Expression → sdo:MusicComposition*.

3.2.3 Identify similar properties

For each class mapped, a mapping between properties should be performed. The criteria are similar to the previous ones:

1. They should have similar name.
   e.g. *mus:U11 has key → sdo:musicalKey*
2. They should have similar description
3. They should have similar value expected
   e.g. *mus:U12 has genre* and *sdo:musicCompositionForm* have both "sonata" as possible value

Each mapped property could have as value a literal (e.g. key, genre and all the "gray" nodes in Figure 3.4) or another class (e.g. the composer is a Person). In the latter case, if we have not previously mapped this class, we consider it as a new input for steps 3.2.2 and 3.2.3, until every node in the graph has been reached.

The result of a complete iteration of 3.2.2 and 3.2.3 is a set of classes and properties mapped, so that a new graph could be drawn (Figure 3.5). It shows that `sdo:MusicComposition` is repeated multiple times, each one with different linked information, depending on the fact that it maps a Work or an Expression.

3.2.4 Simplify the graph

Merging these nodes can produce the advantage of a simpler model, in which the information is distributed in as less nodes as possible. Such an achievement is positive for the consumption

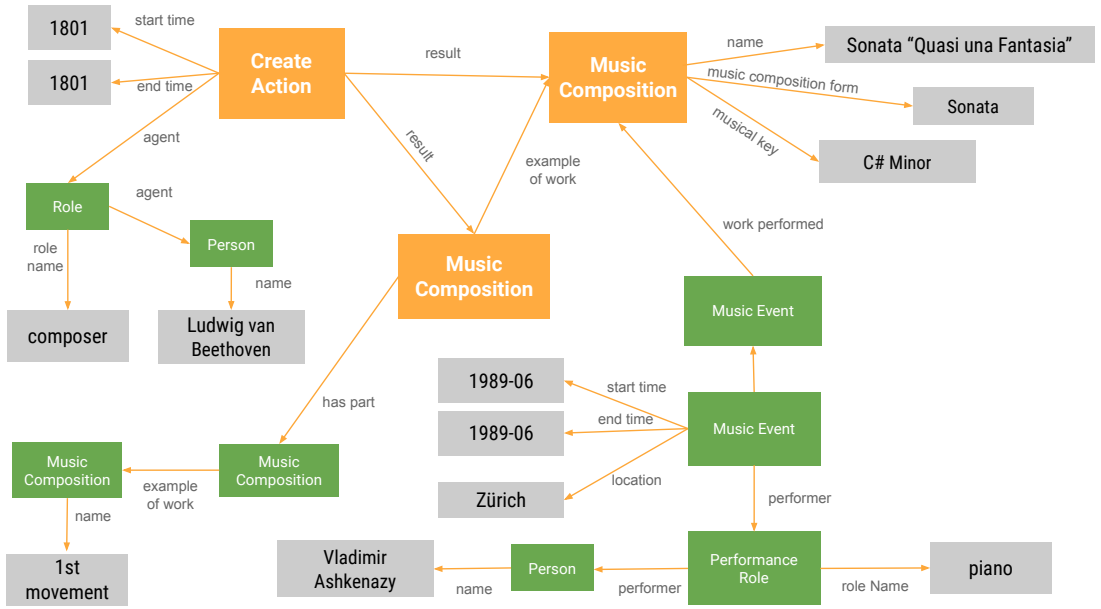


Figure 3.5 – Graph of Beethoven's *Sonata "Quasi una Fantasia"* after the first phases of mapping to Schema.org.

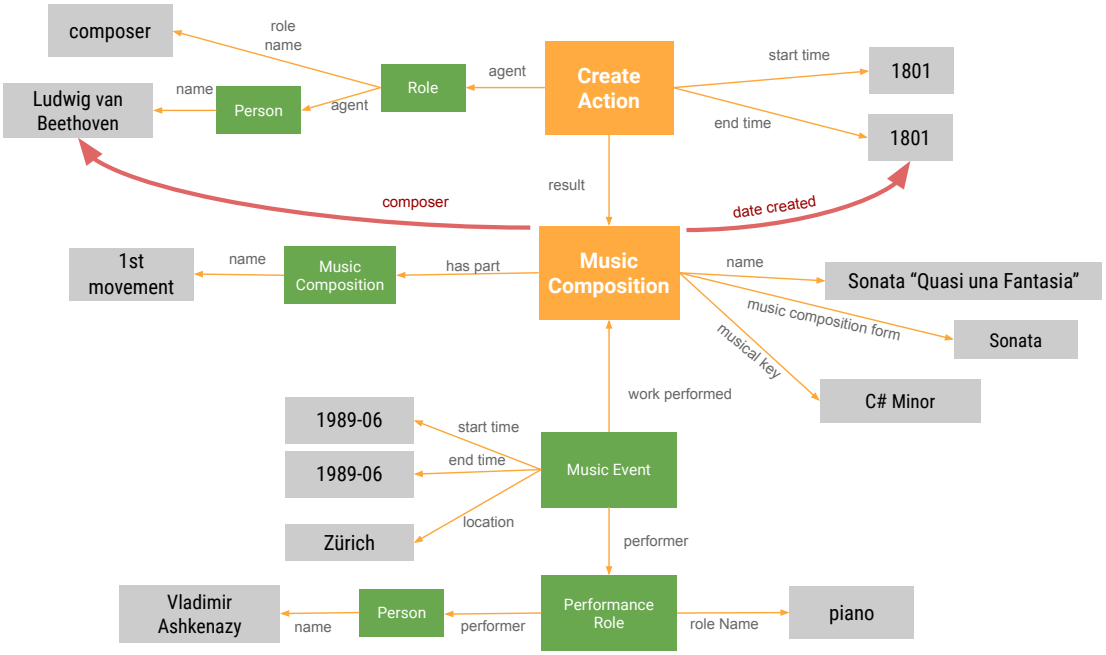


Figure 3.6 – Graph of Beethoven's *Sonata "Quasi una Fantasia"* described through Schema.org.

by search engines, that can display more information in a single search result. In order to do this, we identified some criteria for discerning good candidates for the merging. These criteria should not be considered as strict rules, but as common behaviours of the redundant classes.

Two nodes are redundant when:

1. They represents the same class or have a super-class in common in the target model (this criterion is required).
2. If they are both connected to a class, the connections are both realised with the same properties.
   e.g. *mus:F1 Work* and *mus:F2 Expression* are both mapped with `sdo:Music-Composition` and both are connected to `sdo:CreateAction` with the property `sdo:result`.
3. They are connected between them.
4. They have not properties in conflict (this criterion is required).
   e.g. They could not have different names or keys.
5. The effect of merging does not produce any upset of the graph except a simplification.

The `mus:F1 Work` and `mus:F2 Expression`, both mapped with *sdo:Music-Composition*, satisfy all these criteria. Moreover, we consider also that the difference of these two classes is slight since from the source ontology. As a consequence, the distinction between Work and Expression is simply not relevant to the Schema.org view: users simple search for a book, a movie, a music composition, without taking in consideration the separation between the idea and the realisation [67].

Redundant nodes are substituted with a new node with: 1. The same class of the original ones (or the most specific among the two) 2. The sum of their properties.

The result of this phase is a new graph as it is show in Figure 3.6. It is evident that the new graph has a simpler structure, because of the merging, the omission of some details (i.e. the part is no more marked with type “movement"), and the replacing of some nodes with primitive types. `mus:E52 Time Span` has been replaced with two properties, for start and for end. Instead, we mark in red some properties that in the FRBRoo model are linked to the Event and in Schema.org can be explicit also directly on the `sdo:MusicComposition` class; these properties could not be discovered through our method, but they could be added a posteriori. Table in Appendix reports the complete mapping for the properties involved in the graphs.

3.2.5 Limits of the mapping

As we stated before, a complete mapping can not be gained in the context of this work. However, we point out a set of DOREMUS concepts that have not correspondence in Schema.org

ontology. Among that, we miss information about the librarian cataloguing, the desired casting for the composition (is it supposed to be an orchestra or it is for piano solo?), the tempo (is it “Allegro" or “Andante"). With our strategy we simply do not consider these concept in the mapping.

Schema.org gives the possibilities to add new properties and types through its extension mechanism. Future works will investigate on understanding how identify suitable properties and classes to extend Schema.org.

3.3 Evaluation

Before the beginning of the project, a list of questions have been collected from experts of the partner institutions[5]. These questions reflect real needs of the institutions and reveal problems that they face daily in the task of selecting information from the database (e.g. concert organisation or broadcast programming) or for supporting librarian and musicologist studies. They can be related to practical use cases (the search of all the scores that suit a particular formation), to musicologist topics (the music of a certain region in a particular historical period), to interesting stats (the works usually performed or published together), or to curious connections between works, performances or artists. Most of the questions are very specific and complex, so that it is very hard to find their answer by simply querying the search engines currently available on the web. We have grouped these questions in categories, according to the DOREMUS classes involved in the question. Some examples of those queries are:

- Give me the list of works composed by Mozart in the last 5 years of his life;
- Give me the works of chamber music that involves at most violin, clarinet and piano, except from the sonatas for violin and piano and clarinet and piano;
- Give me all the works interpreted on at least one MoP different from the casting of the work;
- Give me all the performances in which a composer interprets his or her works;
- Give me the name of the vocal soloist most recorded by Radio France in 2014.

Among them, we can find questions that overflow the model, because they contain aspects that go beyond the music information and involve other kind of knowledge. An example is *Retrieve a list of works of chamber music composed in the 19^{th} century by Scandinavian composers*: it requires knowledge of the birth place of the composer, and if this place is located in one of the Scandinavian countries. We can state that these are very interesting questions,

[5]The full list is available at https://github.com/DOREMUS-ANR/knowledge-base/tree/master/query-examples

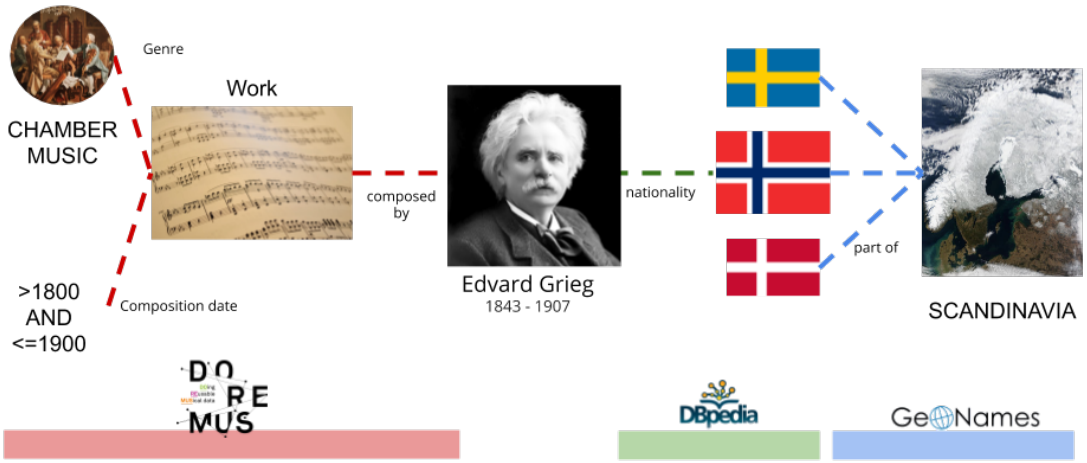


Figure 3.7 – *Retrieve a list of works of chamber music composed in the 19th century by Scandinavian composers* would require 3 different KB in order to be answered.

Category	Questions	Supported by model	Results in the data
A. Works	31	31	23
B. Artists	3	2	1
C. Performances	9	8	6
D. Recordings	11	9	7
E. Publications	5	5	3

Table 3.1 – For each category of questions, we provide the ratio of the number of human-readable queries, how many of them have been successfully converted in a DOREMUS query and how many of them produces at least a result when the query is submitted against the DOREMUS endpoint.

because they are the ones that can fully exploit the advantages of linked data technologies. In fact, this kind of queries are quite far from having an answer in a traditional data storing system (e.g. database). The Web of Data gives the possibility of performing federated queries involving the LOD cloud, in particular datasets such as GeoNames [200] or DBpedia [11] (Figure 3.7). For this reasons, the interconnection of the data is crucial.

Table 3.1 provides an overview of how many queries we can currently write for each category. Few of them find no results in the data. Other are hard to be written in SPARQL format because they involve specific details which are out of scope of the model (i.e. *Retrieve the works by artists that have been mutually lovers*). The conversion rate is anyway more than positive.

3.4 Conclusion

This chapter presented the DOREMUS ontology, an extension of FRBRoo for music.

The model has a very detailed expressiveness that allows, for instance, to describe different kinds of contributors (not only authors or performers), to detail the casting of a composition (with number, roles, notes for each instrument/voice), to specify performers at level of single performance inside a whole concert. This statement is supported by a series of specific questions which get an answer by querying the model.

On the other hand, the DOREMUS model is quite complex and hard to adopt if we look at the levels of distribution of information: from an Expression, one has to pass through Event and Activity to reach a composer, or through Casting and Casting Detail to get the MoP.

This complexity is indeed the heritage of both FRBRoo and FRBR. The DOREMUS classes defines 83 classes and 165 properties, which should be added to the 48 classes and 74 properties introduced by FRBRoo on top of the 84 classes and 161 properties of CIDOC-CRM[6], for a total number much higher than the one of MO (54 classes and 153 properties). The dualism Work - Expression increases the number of required entities and triples for describing each part of the music information, often not really carrying significant information [7]. It is interesting to note that other FRBR-inspired models – like MO – prefer to skip this difference and propose a unique entity `MusicalWork` which puts together the two elements.

Another negative heritage of the extended models is the name convention for classes and properties, which foresees a succession of an uppercase letter, a number and the name of the class or the property, the latter always expressed as a verb. For this reason, DOREMUS requires names like `U54 is performed expression of` in place of the shorter `performance of` of MO, leading to query readability and speed issues. Some properties like `R17 created`, `R18 created`, `R21 created`, `P94 has created` consist in duplicates of the same action applied to different domains or ranges, making the model more error prone. Finally, the absence of a specific Music Work class[8] it turned out to be a impacting problem, making hard to distinguish music pieces from other kind of works like text used in the lyrics, artistic objects used in scenes, etc.

A set of elements that are strictly connected to a librarian and cataloguer vision of the music object are included both in FRBRoo (e.g. `F40 Identifier Assignment`) and DOREMUS (e.g. `U172 has statement of responsibility relating to title`), introduced by the need for tracking the original source of specific statements. The results is mixture of metadata – the ones describing the music and the ones describing the metadata of the music – and in general could make the model be considered too librarian-specific. Further work could overcome this mixture by experimenting new Semantic Web approaches like RDF* and SPARQL* [77].

[6] These numbers do not include inverse properties

[7] A common example is for entities of type *F14 Individual Work*, which quite often are just linked to the Expression, the Expression Creation and the provenance information, like in http://data.doremus.org/work/7259a748-6dd2-3e3d-b9de-7617d0a2b794.

[8] In the dataset, music work are `F14 Individual Work` with a type 'musical work'.

These approaches enable the annotation of a RDF triple, using it in turn as subject or object of an RDF predicate; in this way, an additional layer of information is created which keeps separated the two levels of information.

All these reasons may potentially hamper the adoption of the DOREMUS model by a large public. The simplification of the ontology – for which a first attempt has been performed using the Schema.org vocabulary – is therefore crucial and requires further work.

Chapter 4

Controlled Vocabularies for Music Metadata

Describing music is an activity that involves an important number of terms coming from domain-specific glossaries. In addition to the cross-domain concept of genre, we can mention musical keys, instruments or catalogues of compositions. Libraries and musical institutions have different practices for describing this kind of information. In the best case, they make use of thesauri that are often available in different incompatible formats, and that can be either internally defined or standardised by larger communities such as the IAML. In other cases, this information is codified in free text fields, delegating to the editors the responsibility of following the living practice about syntax and lexical form.

The use of vocabularies opens up different possibilities, like the definition of labels in different languages or of alternate lemmata in the same language (i.e. the French terms “ut majeur" and “do majeur" which both refer to the key of *C major*). Different kinds of relationships between terms can be defined and it is possible to define a hierarchy between them (for example, “violin” is a narrower concept with respect to “string”) which can produce, as benefit, a more powerful advanced search for the final user. Previous research demonstrated how an RDF structure helps reasoning engines to discover links between different levels in the hierarchy of instruments [96].

Publishing Semantic Web vocabularies is not new in the field of music. The Musical Instruments Museum Online (MIMO)[1] published the biggest taxonomy of musical instrument in RDF, as result of the contribution of institutions and universities all over the world. The librarian practice draws on the UNIMARC[2] thesauri of musical forms (genres) and medium of performance standardised by IAML. Historically adopted by librarians worldwide, these thesauri have recently been published in the Web of Data, marking the growing interest in this technological environment. The BnF relies on an authority vocabulary in RDF for subject headings called Répertoire d’autorité-matière encyclopédique et alphabétique unifié

[1] http://www.mimo-db.eu/

[2] They are commonly named after the UNIMARC standard for librarian records, in which they are widely used.

(RAMEAU),[3] containing a list of labels for entities of encyclopedic interest which includes also music genres and instruments. A musical key vocabulary is published as side resource of the Music Ontology [161], consisting in a list of English labelled concepts, with some additional information—like the mode (*major*/*minor*), the tonic, etc.—, without any links describing semantic connections between them.

On the one hand, a large number of thesauri cover few well-defined categories (genres and MoPs), making the reconciliation of data coming from different sources difficult, also because of the different formats of these thesauri. A reconciliation that would add a broader and deeper nomenclature has a benefit, increasing both the number of elements and alternate labels. On the other hand, a large set of concepts – handled so far through error-prone free-text – is asking for standardisation in specialised vocabularies.

This chapter presents a set of controlled vocabularies for the description of the music information as LOD, with the primary goal of the interconnection of music information datasets. These vocabulary carry a relevant amount of structured information which, in the following of the thesis, have a core contribution in empowering recommendation engines.

The complete set of vocabularies, which have been introduced for the first time in [107], will be presented in Section 4.1, giving detailed information about their content. The process of realisation, collection and interlinking is described in Section 4.2, while we present an approach for literal dereferencing in Section 4.3, before the conclusion in Section 4.4.

4.1 Music Vocabularies

A controlled vocabulary is a thematic thesaurus of entities. SKOS [129] have been chosen as format because of its capability of defining preferred and alternate labels in each language, relationships between terms, comments and notes for describing the entity and help the annotation activity. In the case of the vocabulary of *Catalogues of works*, the used ontology is the RDF version of Metadata Object Description Schema (MODS) [199], that suits the need of defining identifiers, publication date, subject, etc.

Each vocabulary fulfils a set of requirements, including multilingualism, open and public access, presence of definitions. It must also be suitable for different contexts of use and conceptual models of musical information, which is guaranteed by the presence in the editorial team of experts from different types of cultural institutions (libraries, radio broadcasting networks, concert halls).

The vocabularies are all available in the DOREMUS triplestore, which enables the HTTP dereferencing of URIs. Alternatively, the vocabularies can be explored by a web browser

[3] http://rameau.bnf.fr/

starting from http://data.doremus.org/vocabularies/ or accessed in Turtle format[4]. Each vocabulary is licensed for free distribution, following a Creative Commons Attribution 4.0 license,[5] and it is open to the community for any kind of contribution.

We collected, implemented and published 23 controlled vocabularies belonging to 18 different families, containing more than 11000 distinct concepts, involving 26 different languages or dialects, defining 610 links between terms. In the following paragraphs, we describe the content of those vocabularies, subdivided in two groups.

4.1.1 Collection of interlinked vocabularies

This group includes vocabularies that were already available in the Web of Data, in the community or internally to a specific institution. When two or more vocabularies share the same high-level topic – e.g. the musical genre – we call that group *family*. In order to interconnect the different knowledge sources, an alignment process is needed for discovering when terms coming from vocabularies belonging to the same family refers to the same concept. This process will be detailed in Section 4.2.1.

Musical genres. This family includes vocabularies about the genre of a musical work. By genre, we mean the main categories by which we describe the works, like rock, funk, opera, gospel, polka, jazz, including genres of world music. The term genre is very broad and also includes musical “forms" that gained in the centuries their own genre definition like symphony, concerto, sonata.

We collected, republished as SKOS and interlinked the following vocabularies:

- **IAML**, 607 concepts, multilingual. This list, largely adopted in librarian environments, was available as a set of labels and codes, in some cases with definitions or editorial notes. We converted this big vocabulary to SKOS from different sources (librarian tabular data, online HTML version). After our publication process started, a SKOS version[6] has been published by IFLA, which is however less rich than ours in terms of alternate labels[7]. We provide `owl:sameAs` links from our vocabulary to the IFLA version.
- **RAMEAU**, 654 concepts, French, hierarchised. It is published as Linked Data by the BnF. We extracted from this large nomenclature the part related to musical genres.
- **Diabolo**, 629 concepts, French, hierarchised. It is the set of labels used in the disc

[4] https://github.com/DOREMUS-ANR/knowledge-base/tree/master/vocabularies
[5] https://creativecommons.org/licenses/by/4.0/
[6] http://iflastandards.info/ns/unimarc/terms/fom/
[7] DOREMUS version count 2990 distinct terms between `skos:prefLabel` and `skos:altLabel`, while IFLA one just 1482.

catalogue of RF. It also includes some `skos:related` links, e.g. between *spiritual* and *gospel*.

- **Itema3**, 40 concepts, French. It is used in the technical documentation of the concert archive of RF.
- **Itema3-MusDoc**, 172 concepts, French. It is used in the musical documentation of the concert archive of RF.
- **Redomi**, 297 concepts, French, hierarchised. It is used in the musical work documentation of RF.

Medium of performance. Any instrument able to produce sounds can be considered as a medium of performance or MoP. In this family of vocabularies, we can find musical instruments coming from different cultures (western, oriental, African, Indian, etc.), the voices in different ranges (soprano, alto, etc.), aside from group of instruments (orchestras, ensembles) and voices (choirs).

We collected, republished as SKOS and interlinked the following vocabularies:

- **MIMO**, 2480 concepts, multilingual, hierarchised. The *Musical Instrument Museum Online* comes from the joint international effort of different music institutions and museum. Despite being the most complete vocabulary of instruments, it does not include voices. MIMO is publicly available as Linked Data.[8]
- **IAML**, 419 concepts, multilingual, hierarchised. Despite its smaller granularity, this vocabulary has a good coverage for voices and groups. Like for the homonym genre vocabulary, also in this case an official version from IFLA is online,[9] less rich both with respect to the languages covered[10] and to the number of concepts (392).
- **RAMEAU**, 876 concepts, French, hierarchised. As in the genre case, we selected the part related to MoPs.
- **Diabolo**, 2117 concepts, French, hierarchised. It is the set of labels used in the disc catalogue of RF. For ethnic or traditional instrument, it includes also the reference to the relative geographic area, possibly referenced to Geonames.
- **Itema3**, 314 concepts, French. It is used in the documentation of the concert archive of RF.

[8] http://www.mimo-international.com/
[9] http://iflastandards.info/ns/unimarc/terms/mop/
[10] 4249 labels for the DOREMUS version, 2591 for IFLA

- **Redomi**, 179 concepts, French, hierarchised. It is used in the musical work documentation of RF.

4.1.2 New vocabularies

This section presents vocabularies for which we did not rely on any previous material, because it was not existing or not suitable for our goals. We designed these vocabularies on the basis of real data coming from institutions, enriched by an editorial process that involved also librarians. Since the work has been conducted in French, the definitions of the terms are so far available only in this language. However, every label has been translated at least in English and Italian in order to facilitate their reuse.

Musical keys. 30 concepts, English, French, Spanish, Italian. This vocabulary contains the set of keys used in western music, labelled with the tone followed from the type of scale (e.g. *C major*). The concept are linked among them by specific properties for keys relationships, like *relative*, *parallel* and *closely related* keys[11]. It contains also `owl:sameAs` links with the key vocabulary of Music Ontology.

Musical modes. 22 concepts, English, French, Italian, Latin, hierarchised. The word *mode* generally refers to a type of scale, coupled with a set of characteristic melodic behaviours. They are mostly used for describing ancient or medieval music.

Catalogues of works. 152 MODS resources. A *thematic catalogue* or *catalogue of works* is a recognised editorial list of all known works of a composer. In practice, a classical composition can be univocally identified by the catalogue code and number. For example, *Eine kleine Nachtmusik* is identified with *K 525*, where *K* is the Köchel catalogue of Mozart's work. Each resource contains the information about the catalogue editor and publisher, the language of drafting, the date of publication. The subject artist of each catalogue is disambiguated through the DOREMUS dataset [5, 112].

Types of derivations. 16 concepts, English, French, Italian, Spanish, German, hierarchised. A work can be derived from another by transforming its material into another through orchestration, harmonisation, etc. All these types (with definitions) are collected in this vocabulary.

[11] The definitions are available in the ontology documentation, see mus:U83_has_relative_key, mus:U84_has_parallel_key and mus:U85_has_closely_related_key

Functions. 106 concepts, English, French and Italian, hierarchised. A music event—a performance, a composition, a recording, etc.—involves a number of different roles or *functions* like author, performer, conductor, sound engineer, etc. Additional details can also be provided to account for the different kinds of author, like composer, lyricist or arranger. These functions are identified in this vocabulary, together with their definitions.

Responsibility. 8 concepts, English, French and Italian, hierarchised. It allows to specify the type of responsibility exercised by a musician through its medium of performance – e.g. soloist, choir singer, etc.

Hierarchical Level for Work. 9 concepts, English, French and Italian, hierarchised. It allows to specify the level of granularity of a musical piece respect to the main composition – e.g. part, single work, set of works, movement, overture, scene, etc.

Vocal and instrumental techniques. 19 concepts, English, French and Italian. This vocabulary contains different instrument playing or voice production techniques which can modify the output sound and produce specific effects, like whistling, scat or yodel.

Other vocabularies – less related to the music information but used in the DOREMUS graph for describing different kind of material – are Carrier Type (e.g. magnetic wire, DVD), Color Content (monochrome, polichrome), Types of identifier (inventory number, ISBN), Noise Reduction Techniques, Conditions of Performance (indoor performance, studio performance), Performer Status (guest artist, headliner), Playing Speed, Types of recording equipment (digital, acoustic), Sound Spatialization Techniques (mono, stereo), Work types (music, choreography).

4.2 Modelling process

The modelling process, which is based on an interaction between music metadata experts and automatic data conversion and fusion tools.

An editorial committee grouping 7 members coming from different backgrounds (library, radio, concert hall) played an important role in the vocabulary modelling. First, existing vocabularies have been inventoried and assessed as candidates for being interlinked on the basis of their completeness and adoption. Next, the committee made choices about which new vocabularies to create and what should be their scope. These choices reflect the aim of producing powerful tools to describe recordings, publications and their contexts of creation, instead of producing exhaustive vocabularies about every aspects of the music. The committee relies on the members experience in music data management practices. The experts had

to confront their point of views – necessarily different because depending on the missions of their institutions – until the list of terms, their contexts of use and their definitions were coherent.

For generating the structured version of the vocabularies, we performed a preliminary conversion from spreadsheets or XML files to RDF, using the OpenRefine tool [74] or with specific scripts. The collections of concepts already in the Web of Data (like RAMEAU) have been instead extracted through specific SPARQL queries on their original endpoints.

In a second step, additional vocabulary-specific actions are performed. In some cases, hierarchy is inferred on the basis of specific properties and rules (e.g. in the IAML MoP vocabulary, the hierarchy is taken from the letters included in the last part of the URI). All the language tags are normalised in order to follow the ISO 639 standard[12]. Moreover, the indication of the use of Latin script is made explicit for transliterated labels in languages that use different alphabets. In this phase, some interlinking to external datasets is performed, using SPARQL queries (DOREMUS dataset, Music Ontology keys vocabulary) or APIs like GeoNames [200].

4.2.1 Vocabulary Alignments

The sets of vocabularies of musical genres and those of medium of performance, described in Section 4.1.1, group together a number of well-established or internally used within a given institution reference lists. There is an important overlap between the sets of entities (genres or musical instruments) described across these vocabularies in each of the two categories. For example, the music genre “folk song" is described both in the IAML vocabulary (labelled by the French “chanson populaire" and the English “folk song") and in the Radio France-hosted Diabolo vocabulary (labelled by “folksong").

A vocabulary alignment has been realised by the LIRMM Laboratory in Montpellier for automatically establishing links of identity between the elements of two vocabularies from the same category. Since our vocabularies are described in SKOS, the procedure comes down to discovering and declaring `skos:exactMatch` relations across the terms of two given vocabularies, e.g http://data.doremus.org/vocabulary/iaml/genre/fso and http://data.doremus.org/vocabulary/diabolo/genre/folksong.

The result is a set of pairwise alignments between the concepts of the vocabularies in each of these two categories (genres and MoP) to a chosen target vocabulary, being IAML for the genres and MIMO for the MoP. Automatic alignments have been performed through a string-based approach, looking both at preferred and alternative labels, returning in output a confidence score. These alignments have then been validated by the librarian experts through YAM++

[12] http://www.iso.org/iso/home/standards/language_codes.htm

online [15] – a multi-task web platform for ontology and thesaurus matching and validation,[13].

4.3 String2Vocabulary

A common task in what is called *knowledge graph population* – which is the generation of semantic triples starting from differently structured data sources – is the passage from plain text nodes or *literals* to a more representative object node or *entity*. Often, the target of this task consists of a set of vocabularies.

A *string2uri* algorithm – developed in the context of the Datalift platform [175] – performs an automatic mapping of string literals to URIs coming from controlled vocabularies in SKOS. The software reads a RDF graph and searches for exact matches between literal nodes and vocabulary terms.

Some experiences in knowledge base population of classical music data, have shown up some critical points. Often the title of a classical work includes or, even more, consists in the name of an instrument or a key or a genre (e.g. Ravel's *Bolero*), that should be excluded from the replacement process and be kept as textual literals. Moreover, the complexity itself of this data – involving an important number of properties – in addition to the commonly used file formats (i.e. MAchine-Readable Cataloging (MARC)), has led in the years to a cataloguing practise particularly prone to editorial mistakes. This is the case of musical keys declared as genre, or fields for the opus number that contain actually a catalogue number and vice-versa [112].

For these reasons, we adapted the Datalift strategy in a new *String2Vocabulary* open-source library.[14] The software uses the file name of vocabularies for grouping them in families: *mop-mimo.ttl* and *mop-iaml.ttl* are part of the family *mop*, while *key.ttl* is the sole member of the family *key*. This library accepts a configuration file that assigns a family to a RDF property. For each input graph, it searches for the properties one after the other, retrieving their values. Each value is then compared to all the terms of the vocabulary, until it finds one equal to the value. All variants for a concept label – namely `skos:prefLabel` and `skos:altLabel` – are considered in order to deal with potential differences in naming terms, and both graph values and terms receive a normalisation that has the effect of removing the punctuation, lower-casing the text and decoding it into the American Standard Code for Information Interchange (ASCII) format. Then, a substitution of that node with the found concept URI is performed.

String2Vocabulary works both with literal values and with entities labelled through `rdfs:label`. In the latter case, the label to be matched against the vocabulary and the whole node – with all its properties – is replaced. For maximising the possibilities of selecting, if it exists, the right concept, two searches are performed in sequence. The first requires that both the given text

[13] http://yamplusplus.lirmm.fr
[14] https://github.com/DOREMUS-ANR/string2vocabulary

and language match with the concept ones. If this search fails, a second one requires a match excluding the language information.

As additional feature, the configuration file allows to request the lemmatisation for certain vocabularies. Taking the MoP vocabulary as representative example, three sequential matches are tried:

1. singularising the first word of the label, for matching cases such as *“cornets à pistons"@fr*;
2. singularising the whole label, like in *“sassofoni contralti"@it*;
3. leaving the label as is for matching instruments that are always plural, like *“cymbals"@en*.

4.4 Conclusion

We have presented a set of multilingual vocabularies for the description of music-specific concepts using the Semantic Web framework. Two main contributions are the interconnection of already in-use vocabularies of genres and medium of performance and the realisation of previously-unreleased ones. We described our working strategies as an interaction between editors and an automatic system. A dereferencing library *String2Vocabulary* is presented as side works.

We have the intention of proposing to IFLA some modification to IAML vocabularies, based on the DOREMUS ones. However, we face the absence of any evident possibility of contribution coming from external subjects.

Those vocabularies are intended to become references in the field, with the goal of being reused even outside the context of the DOREMUS project. They are open for contribution and extension through the GitHub repository. Other projects, as the already mentioned Performed Music Ontology (PMO), have already recognised the value of this work, encouraging the adoption of the vocabularies[15].

[15] https://github.com/LD4P/PerformedMusicOntology

Chapter 5

Data Conversion

After having introduced the DOREMUS model and vocabularies, this information structure must be populated by data. The knowledge from the archives of the three partner institutions in the DOREMUS project – BnF, PP and RF – covers different part of the music information, from the works to the recordings, including scores, discs and concert programmes. Table 5.1 details the data sources for each institution, revealing an interesting number of records. However, the data comes in different heterogeneous formats.

The BnF and partially the PP uses a standard format called MARC, which will be detailed in Section 5.1. We developed a generic converter for MARC files in the DOREMUS format in Section 5.2. Other data sources are in the XML format, which can sometimes be an export of relational databases. All these XML follows different conventions about structure and tag names, for which we had to develop ad hoc converters.

This chapter reports the full process leading from the source files from partner institution to the RDF graph. It results in the construction of several knowledge graphs about music works and events, which are than interlinked, going to compose the DOREMUS graph (Section 5.3).

The conversion strategy have been published in [104] in the first preliminary version, and in [110] in the definitive version. The DOREMUS graph and the full conversion pipeline has been presented for the first time in [5].

	Person	Works	Recordings	Scores	Discs	Programmes
BnF	INTERMARC 47 852	INTERMARC 160 368		INTERMARC 86 140	INTERMARC 164 753	
PP		UNIMARC 5 762	UNIMARC 96 914			XML 3 837
RF		XML 62 550	XML 2 296		XML 9 343	

Table 5.1 – Data sources: numbers and formats

Figure 5.1 – An handwritten record in Radio France archives.

001 FRBNF139081882FR
100 $3 13891295 $w .0..b.....$a Beethoven $m Ludwig van $d 1770-1827
144 $w....b.fre.$a Sonates $b Piano $p Op. 27, no 2 $t Do dièse mineur
NUM SUB

Figure 5.2 – An excerpt of a UNIMARC record.

5.1 MARC and the librarian practice

The archives of cultural institutions like national libraries are the result of a cataloguing process which lasted centuries, started much before the advent of computers and electronic database. Less than one century ago, libraries were used to index their contents by using printed or handwritten cards, each one representing one or few records, stored in alphabetic order. Those cards, apart from the position of an item in the library, were containing also some metadata about the work itself, like the author, the number of parts, the editor. An example is shown in Figure 5.1. Those cards contained an important and crucial amount of the human knowledge.

In 1960s, the technological progress allowed the information to be recorded on electronic supports and be processed by machines. In this context, the Library of Congress of United States developed the **MARC** format, which became soon an international standard for representing the metadata in libraries.

MARC records are organised as a succession of lines (**fields**) introduced by a specific numeric code with 3 digits. Each field contains a specific information about the item described in the record and can host one or more **subfields**, which are concatenated in the field, introduced by a single alphanumeric character which defines which information the value represents and separated by a marker (the dollar sign $). An example of MARC record is in Figure 5.2.

The such defined syntax is embodied in different variant of the MARC format, which differs in the semantic of the fields. The MARC variant involved in our work are:

- **Universal MARC (UNIMARC)**, standard developed by IFLA for harmonising a set of different variants adopted at national level. Its adoption is particularly successful in Europe. The records involved in the DOREMUS dataset coming from PP follow the UNIMARC format;
- **InterMARC**, initially developed for being the French standard format for libraries. It is today used internally in French institutions like the BnF, which provides data for the DOREMUS project in this format.

MARC reflects perfectly the context of its birth, around 60 years ago. Its structure made of a succession of fields is the exact conversion of printed or handwritten records in a machine-readable form [113]. The alphanumeric tags for fields and subfields are a consequence of the state of the technology at the time, for which the storing space was expensive and the computational power low.

The format appears far from the current best practices for structured data. We identified some issues of MARC, some of them strongly interconnected:

- **Hard readability.** MARC fields are also not labelled explicitly, but encoded with numbers, with the consequence of having to receive a training or study a manual for deciphering the content. The semantics of these fields and subfields is not trivial: a subfield can change its meaning depending on the field under which it is found, and on the particular variant of the format.
- **Lack of interoperability.** The format does not foresee any possibility of sharing data between different institution. The presence of different variants aggravates the problem, requiring the conversion between formats.
- **Technical marginalisation.** Outside the librarian world, the MARC format is totally unknown. As a consequence, the software capable of read and manipulate MARC records is restricted to the librarian cataloguing tools. Any further exploitation of the data requires a parsing of the records and a serialisation to other formats. In addition, this parsing and conversion are challenging task for developers, given the cryptic syntax and semantic.
- **Unstructured information.** As explained before, MARC replace and replicates in the behaviours the old textual records, which were realised on paper cards. It is therefore not surprising that large part of the information continues to be expressed as free-text. The same field or subfield can contain information about different entities, like the first

performance and the first publication combined in the same field of the notes, without a clear separation. Moreover, depending on the editor that filled the record, different practices can be in place for describing the same information – we can for example find either “Op. 27 n. 2" or “Op. 27 no 2". Structured information have to be extracted from those text fields.

- **Mistakes in editorial work.** In years of librarian practice, an huge number of records have been realised. The creation of new records – for new element or copying old printed records – was largely charged on human editors. This unavoidably lead to some mistakes in the records. In the majority of cases, we speak about little typos or incongruous punctuation, in particular in free-text fields. In others, we can find misplacement of values, so that the genre may appear in the subfield dedicated to the opus number or vice versa.

About the limits of MARC and about going beyond this old format, some discussion comes also internally to the librarian world. In 2002, Roy Tennant – at the time working at the California Digital Library – wrote a highly referenced article [188] which directly and mercilessly stated that:

> *MARC must die.*
>
> — Roy Tennant, 2002

Even if MARC is still widely used, a slow transition to other formats is occurring, the most popular solution being the use of Linked Data, also thanks to the BIBFRAME Framework Initiative [99]. The benefits of moving from MARC to an RDF-based solution consist of the interoperability and the integration among libraries and with third party actors, with the possibility of realising smart federated search [8, 27]. The development of ontologies and vocabularies enables the exploiting of the knowledge contained in librarian archives, with the possibility of performing reasoning, machine learning, automatic classification, graph embeddings, etc.

The efforts of this research try to put a step further in the overcoming of MARC for music metadata. In order to achieve this goal, two tasks are necessary: data conversion and data linking.

5.2 From MARC to RDF

We developed `marc2rdf`, an open source prototype[1] for the automatic conversion of MARC bibliographic records to RDF using the DOREMUS ontology. The conversion process relies on

[1] https://github.com/DOREMUS-ANR/marc2rdf

UNIT OF INFORMATION	F22 Expression: Opus Number
PATH	F22 Self-Contained Expression U17 has opus statement M2 Opus Statement [U42 has opus number M12 Opus Number] + [U43 has opus subnumber M13 Opus Subnumber]
INTERMARC BNF	TUM : 144 $p, chain of digits TUM : 144 $p, chain of digits before the comma
TRANSFER RULE	Remove the abbreviation "Op." before the number
EXAMPLE	144 $pOp. 352 --> M12 = 352 144 $pOp. 27, no 2 --> M12 = 27, M13 =2

Figure 5.3 – Example of mapping rules describing the opus number and sub-number of a work

explicit expert-defined **transfer rules** (or mappings) that indicate where in the MARC file to look for what kind of information, providing the corresponding property path in the model as well as useful examples that illustrate each transfer rule, as shown in Figure 5.3. The role of these rules goes beyond being a simple documentation for the MARC records, embedding also information on some librarian practices in the formalisation of the content – e.g. format of dates, agreements on the syntax of textual fields, default values if the information is absent.

The converter is composed of different modules, that works in succession, as shown in Figure 5.4. First, a **file parser** reads the MARC file and makes the content accessible by field and subfield number. We implemented a converting module for both the InterMARC and UNIMARC variants. Then, it builds the RDF graph reading the fields and assigning their content to the DOREMUS property suggested in the transfer rules. Each entity is identified by an univocal persistent URI, which follows the pattern `http://data.doremus.org/<group>/<uuid>`, where the group is determined by the class of the entity (e.g. `expression`) and the universally Unique Identifier (UUID) is generated at conversion time in a deterministic way using the dataset name, the class and the identifier of the source record as seed.[2]. This strategy ensures that successive conversions of the same source files produce identical outputs. For managing the triples, we relied on Apache Jena[3].

A **preliminary interlinking** for places and artists is performed. For places, we relied on GeoNames [200], a which exposes with an API its large community-driven database about not only cities, regions, countries, but also relevant point of interests like theatres, churches, auditoriums, concert halls. When there is no suitable match in GeoNames, a `E53 Place` entity is created, possibly linked to the belonging city or country. We rely on a unique key composed by name, surname and birth date for interlinking the artists between the different data sources and with International Standard Name Identifier (ISNI)[4], the larger index for writers, artists,

[2] https://github.com/DOREMUS-ANR/marc2rdf/blob/master/URI.patterns.md
[3] https://jena.apache.org/
[4] http://www.isni.org/

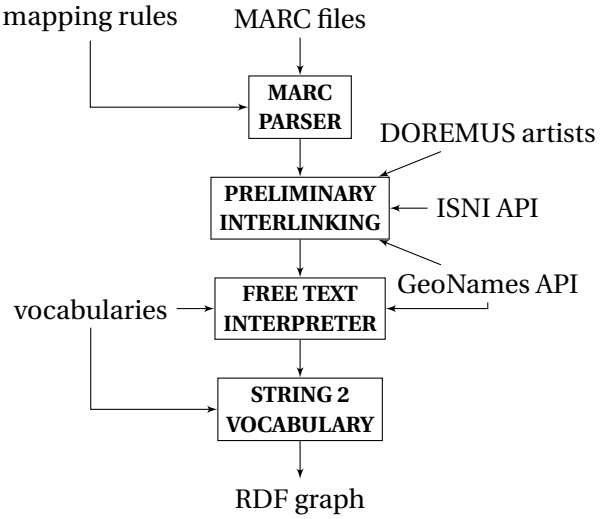


Figure 5.4 – `marc2rdf` application schema

performers, publishers, etc. The interlinking is performed following these steps:

1. The artist is searched among the ones already in DOREMUS dataset, using name, surname and birth date.
2. If the ISNI identifier is unknown, the ISNI API is queried. The eventually found identifier can be used to query the DOREMUS dataset again, in case the previous point failed.
3. In in previous points the artist was found in the DOREMUS dataset, the URI is re-used. Otherwise, a new entity is created. The information about the ISNI identifier is inserted in the dataset.

ISNI database enable the access to other services, providing the possibility of further enriching data about artist with text description, alternate names and pictures. Among those services, we mention Virtual International Authority File (VIAF)[5] (specialised on authors and works), the encyclopedic datasets Wikipedia, Wikidata, DBpedia, and MusicBrainz[6] [185], one of the most popular knowledge bases about music metadata, which started a few years ago its process of exposing its data as semantic triples through the platform LinkedBrainz [84].

Then the *free-text interpreter* extracts further information from the plain text fields, that includes editorial notes. This amounts to do a knowledge-aware parsing, since we search in the string exactly the information we want to instantiate from the model (i.e. the MoP from the casting notes, or the date and the publisher from the first publication note). The parsing is realised through empirically defined regular expression, validated and corrected by the use of

[5] https://viaf.org/
[6] https://musicbrainz.org/

vocabularies and of GeoNames for ensure the correctness of the identified types.

Finally, the *string2vocabulary* component – described in details in Section 4.3 – performs an automatic mapping of string literals to URIs coming from controlled vocabularies. As additional feature, this component is able to recognise and correct some noise that is present in the source MARC file: this is the case of musical keys declared as genre, or fields for the opus number that contain actually a catalogue number and vice-versa. These cases and other typos and mistakes have been identified thanks to the conversion process and the visualisation of the converted data, supporting the source institution in they work of updating and correcting constantly their data.

5.3 A set of interlinked graph

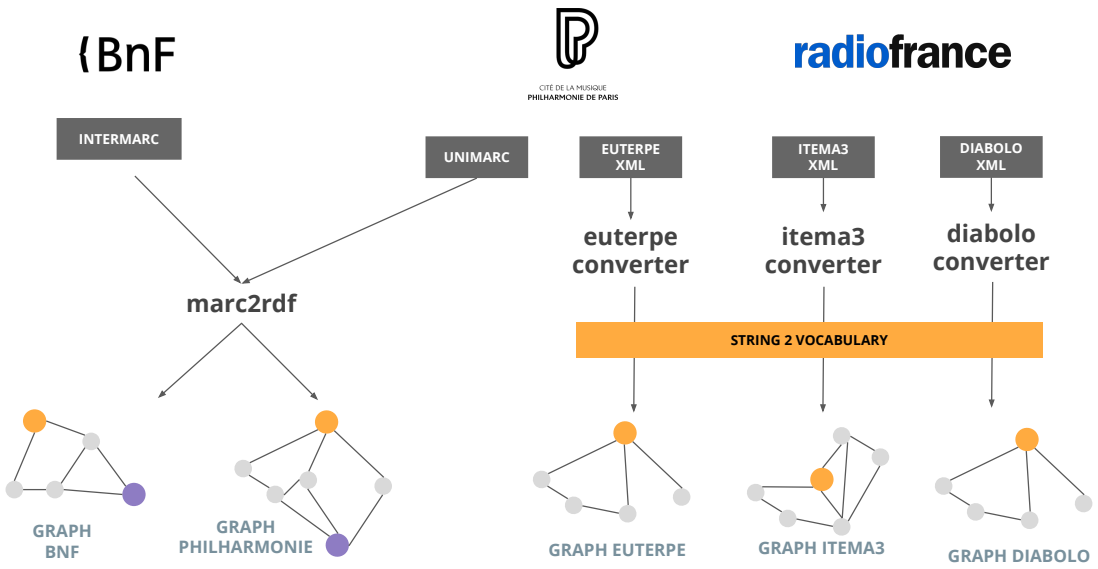


Figure 5.5 – The converters produce a distinct graph for each data source

Apart from MARC, we are converting other source bases (in XML), that are too specific to be handled by a single converter. Therefore, we developed a set of software that, even if they specifically target a specific dataset and cannot be considered general, they re-use some of the modules of `marc2rdf`. The converters have a generic work-flow: parse the input file and collect the required information, create the graph structure in RDF, run the *string2vocabulary* module, integrating during the conversion the interlinking to Geonames and ISNI.

This procedure creates different graphs, one for each source:

- **BnF**, including converted records about artists, works, discs and scores;
- **PP**, including converted records about works and recordings;

- **Euterpe**, including converted records about the foreseen concerts at the Philharmonie de Paris;
- **Itema3**, including converted records about recordings occurred at Radio France;
- **Diabolo**, including converted records about works archived at Radio France.

Those source databases are complementary but also contain overlaps – e.g. two databases that describe the same work or the same performance with complementary metadata. There is so a need for the automatically interlinking of the datasets, so that the resulting knowledge graph provides a richer description of each work. The interlinking between works has been performed by the LIRMM Laboratory, while the links have been validated by experts in BnF, PP and RF [3, 4, 5].

Class	BnF	Philharmonie	Euterpe	Itema3	Diabolo	Total
Person	69,948	8,419	9,269	9,040	1,503	89,872
Corporate Body	15,429	1,603	1,001	39	0	18,075
Expression	365,563	14,875	10,587	15,016	12,344	420,733
- with at least 1 performance	258,304	12,725	10,578	12,602	2,294	296,503
Performance	179,696	7,107	3,833	2,296	2,294	193,065
- with >1 performed works	24,974	2,615	13,520	1,922	0	43,455
Recording	165,223	3,406	0	2,296	0	170,925
Track	415,252	40,991	0	27,018	0	483,261
Publication (scores)	31,296	0	0	0	0	31,296

Table 5.2 – DOREMUS Knowledge Graph: overview of the content

Currently, the DOREMUS KG includes more than 90 million triples, which describe over 18 million distinct entities. The classes and properties used come mostly from the DOREMUS ontology, FRBRoo and CIDOC-CRM, counting in total 67 distinct classes and 178 distinct properties. Table 5.2 summarises the number of entities for the most representative classes and reports details about the presence of specific information. The Persons and Corporate Bodies includes respectively single individuals or groups which can play the role of composers, performer, publishers, etc. Being a dataset about concert programmes, the data about Euterpe are referred to Foreseen Performances. The total can be less than the sum of the single columns because of the interlinking previously described.

5.4 Conclusion

This chapter presented some of the core contributions of this research and of the DOREMUS project, including the DOREMUS graph and a pipeline for converting librarian metadata.

The DOREMUS graph is a big collection of music metadata published in the Web of Data,

making an important amount of human knowledge available for everyone. The uniqueness of this resource is remarkable when compared to other music-related datasets: we outline that the BBC open datasets have tracks only, the Dutch Library (part of Europeana) has only publications, Choral Public Domain Library (CDPL)[7] is specialised for chorus (with scores and MIDI). The more heterogeneous content of the DOREMUS dataset can become a bridge between these datasets and ease their interconnection. Another big music dataset like MusicBrainz follows a more commercial practice giving a central role to tracks, albums and artists (un-distinguishing the composer from the performer), at the expense of all the information connected to the work concept (genre, casting, key, etc), in contrast to the librarian structure which characterises DOREMUS.

The pipeline we used for building it can be generalised and applied to other projects, in particular in the field of Digital Humanities and libraries. Some of the tool we presented are already general and ready for new applications.

As repeated several times, the extraction of structural data from librarian archives is now a crucial goal for the libraries themselves. The conversion of the data, the inclusion of controlled vocabularies in the process, and the possibility of querying the data provides access to parts of information previously not accessible – like for example the questions collected in Section 3.3. On the other side, editorial mistakes and typos in the source files have been detected both at the conversion time – exceptions in parsing of fields or missed matching against the vocabularies – and in the visualisation of the final results, giving to partner institutions the possibility of correcting their datasets.

These results can be improved together with the conversion process in some future work:

- More robust Natural Language Processing (NLP) techniques should be used for parsing the free-text fields. The research can involve both the possibility of using a state-of-the-art algorithm – better if trained on a subset of those fields – and the implementation of specific ones for classical music.
- The detection of possible mistakes in the source files can be automatised, producing a list of probable candidates for the correction. The correction prediction can be an interesting research topic.
- Alignments of our data to established datasets – in particular MusicBrainz – are currently being generated.
- The dataset may take benefit of some human contribution. A collaborative UI may let end users to edit the information, enrich it, discover links between entities, improving the completeness and correctness of the information.

[7] https://www.cpdl.org/

Chapter 6

Developing smart Web APIs

The DOREMUS KG can be considered a 5-star dataset, exposing RDF data, providing URI dereferencing, and including links to external datasets [17]. In addition, DOREMUS data are accessible through a public SPARQL endpoint[1] realised with *Openlink Virtuoso*[2]. A set of RDF files in Turtle format are available to public download[3]. All datasets are licensed for free distribution, following a *Creative Commons Attribution 4.0* license[4] and have a Data Catalog Vocabulary (DCAT) description in the triplestore itself.

In Section 2.4, we discussed about some solution for building web services on top of Linked Data. In the context of the DOREMUS dataset, the complexity of the ontology structure needs a mapping of the data into simpler structure (Section 3.2). The result is a generic solution for transforming the SPARQL output in a format that is more suitable for its consumption in web applications and not only, which has been published under the name of *SPARQL Transformer* [106, 111]. This chapter is detailing the motivation and the main results of this approach.

This chapter is structured as follow. Section 6.1 contains some motivation and requirements which lead to the design of SPARQL Transformer. Those requirements were not completely satisfied in other works which aim to ease the consumption of RDF data, briefly reviewed in Section 6.2. We introduce the new JSON format for queries in Section 6.3, which feeds the SPARQL Transformer library detailed in Section 6.4. The work is finally evaluated in Section 6.5, while some conclusions and future work are presented in Section 6.6.

[1] http://data.doremus.org/sparql
[2] https://virtuoso.openlinksw.com/
[3] https://github.com/DOREMUS-ANR/knowledge-base
[4] https://creativecommons.org/licenses/by/4.0/

6.1 Motivation

RDF can potentially represent any kind of knowledge, enabling reasoning, interlinking between datasets, and graph-based artificial intelligence. Nevertheless, a structural gap exists that is limiting a broader consumption of RDF data by the community of Web developers. Recent initiatives such as EasierRDF[5] are strongly pushing the proposal of new solutions for making Semantic data on the Web *developer friendly* [23, 65].

We focus here on the output format of SPARQL endpoints, and in particular, query results in the JSON format [180]. This standard is part of the SPARQL W3C recommendation [75], introduced with the purpose of easing the consumption of the data by Web (and non-Web) applications. The format consists of a set of all possible bindings (of the form `<variable, value>`) that satisfies the query. This is not handy for efficient processing by clients, which would prefer nested objects (document-based data structures) rather than this representation of triples (graph-oriented data structures). An example of this is shown in Figure 6.1.

Given this situation, we identify four tasks that developers have to fulfil:

1. **Skip irrelevant metadata.** A typical SPARQL output contains a lot of metadata that are often not useful for Web developers. This is the case of the `head` field, which contains the list of variables that one might find in the results. In practice, developers may ignore completely this part and check for the availability of a certain property directly in the JSON tree.

2. **Reducing and parsing.** The value of a property is always wrapped in an object with at least the attributes *type* (URI or literal) and *value*, containing the information. As a consequence, this information is bounded at a deeper level in the JSON structure than the one the developer expects. In addition, each literal is expressed as a string value with a datatype, so that numbers and booleans need to be casted.

3. **Merging.** As the query results represent all the valid solutions of the query, it is possible that two bindings differ only by a single field.
   When the number of properties that have multiple values grows – i.e. multilingual names, multilingual descriptions, a set of images –, the endpoint returns even more results, one for each combination of values. The consumption of such data requires often to identify all the bindings which represent a given entity, merging the objects on the URI. The presence of more variables on which the merging can be performed can further complicate the merging process.

[5] https://github.com/w3c/EasierRDF

```
SELECT DISTINCT *
WHERE {
    ?id a dbo:City ;
          dbo:country dbr:Italy ;
          rdfs:label ?label .

    OPTIONAL { ?id foaf:depiction ?image }.

    ?id dbo:region ?region .
    ?region rdfs:label ?region_name .
    FILTER(lang(?region_name) = 'it')

} LIMIT 100
```

(a)

```
[{
    "id": "http://dbpedia.org/resource/Siena",
    "name": [{
        "language": "fr",
        "value": "Sienne"
      },
      {
        "language": "it",
        "value": "Siena"
      },
    ],
    "image": "./PiazzadelCampoSiena.jpg",
    "region": {
      "id": "http://dbpedia.org/resource/Tuscany",
      "name": {
        "language": "it",
        "value": "Toscana"
      }
    }
  },
  {
    "id": "http://dbpedia.org/resource/Milan",
    "name": {
      "language": "en",
      "value": "Milan"
    },
    "image": "./Flag_of_Milan.svg",
    "region": {
      "id": "http://dbpedia.org/resource/Lombardy",
      "name": {
        "language": "it",
        "value": "Lombardia"
      }
    }
  }]
```

(b)

```
{
  "head": {
    "link": [],
    "vars": [ "id", "label", "image", "region", "region_name" ]
  },
  "results": {
    "distinct": false,
    "ordered": true,
    "bindings": [{
        "id": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/Siena"
        },
        "label": {
          "type": "literal",
          "xml:lang": "it",
          "value": "Siena"
        },
        "image": {
          "type": "uri",
          "value": "./PiazzadelCampoSiena.jpg"
        },
        "region": { ... },
        "region_name": { ... }
      },
      {
        "id": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/Siena"
        },
        "label": {
          "type": "literal",
          "xml:lang": "fr",
          "value": "Sienne"
        },
        "image": {
          "type": "uri",
          "value": "./PiazzadelCampoSiena.jpg"
        },
        "region": { ... },
        "region_name": { ... }
      },
      {
        "id": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/Milan"
        },
        "label": {
          "type": "literal",
          "xml:lang": "en",
          "value": "Milan"
        },
        "image": {
          "type": "uri",
          "value": "./Flag_of_Milan.svg"
        },
        "region": { ... },
        "region_name": { ... }
      }]
  }
}
```

A

B

C

(c)

Figure 6.1 – A SPARQL query (a) extracting a list of Italian cities with picture, label and belonging region, of which the URI and the Italian name are also requested. In the standard output of the endpoint (c), the city of Siena is represented by both object A and B, while the transformed output (b) offers a more compact structure.

4. **Mapping.** The Web developer may want to map the results to another structure – i.e. for using them as input to a library – or vocabulary such as *schema.org*.

6.2 Related Work

The need for overcoming the issues about the usage of SPARQL output in real-life applications has inspired different works. One of the first proposed solutions consists in a strategy for representing the SPARQL output in a tabular structure, to address the creation of HTML reports [1].

Wikidata SDK [103] takes care of the reduction and parsing tasks through a precise function[6] that transforms the JSON output to a simplified version by reading the variable names. However this implementation does not address the problem of merging.

The conversion of RDF data can rely on the **SPARQL Template Transformation Language (STTL)** [42]. Those transformation templates (as strings) are exploited for shaping the results of the SPARQL query. Moreover, STTL exposes a significant number of functions, especially when combined with LDScript [43]. Among the limits of this approach is the absence of any support for converting the results to JSON-LD. No merging strategy is also studied in this approach.

The `CONSTRUCT` query format – included in the W3C SPARQL Specification [75] – can be seen as a way for mapping the SPARQL results into a chosen structure, following one of the standard SPARQL output formats, including JSON-LD. An attempt has been realised by the command-line library `sparql-to-jsonld`[7]. The need for three different inputs – a `SELECT` query, a `CONSTRUCT` or `DESCRIBE` query, and a JSON-LD frame – indirectly proves that a sole `CONSTRUCT` for shaping JSON with non predefined structure is not sufficient. Indeed, the `CONSTRUCT` keyword cannot generate trees, but only triplesets, which leads to the problem of how to change the structure of the query result. Frames overcome this problem, but, in our opinion, the combination is not easier for developers who would have to write and keep in sync the two parts (query and result shape). The complexity of writing a `CONSTRUCT` query – i.e. with respect to a `SELECT` one – can be an additional deterrent for its usage. Furthermore, literals are not parsed and they are always represented as objects, and aggregate functions are not supported.

JSON Schema is a format for defining the structure of a JSON object. Although it is a powerful tool for validation – for example – of forms and APIs, there are no evident benefits for JSON reshaping purposes [202].

The development of *SOLID* framework for decentralised LD applications [114, 196] gives popularity to its module *LDflex*[8] for retrieving and manipulating Linked Data. LDflex allows the user to browse nodes in the graph by accessing to JS properties. Thus, the paradigm of this

[6] https://github.com/maxlath/wikidata-sdk/blob/master/docs/simplify_sparql_results.md
[7] https://github.com/jindrichmynarz/sparql-to-jsonld
[8] https://github.com/RubenVerborgh/LDflex

module is different, consisting in navigating the graph following the links, rather than finding solutions to structured queries.

There is abundant work in SPARQL query repositories, which are typically used to study the efficiency and reusability of querying. For example, in [167] authors use SPARQL query logs to study differences between human and machine executed queries; in [83], these logs are used to understand the semantic relations between queried entities. Saleem et al. [174] propose to "create a Linked Dataset describing the SPARQL queries issued to various public SPARQL endpoints".

6.3 The JSON query syntax

As seen in the experiences reported in Section 6.2, the natural choice of format for defining and developing a transformation template involves JSON or its JSON-LD serialisation, which is usually added to the SPARQL query. The names of the variables used should match between the template and the query, making the developing process error-prone.

Our proposal is to use a single JSON object, called *JSON query*, with the double role of declaring how to find the information (query) and which structure is expected in its output (template). These properties put the JSON query at a certain distance also from SPARQL `CONSTRUCT`, in which the query and the final structure are two distinct parts of the query.

The syntax of JSON queries consists of two main parts (Listing 6.1):

- the prototype definition, which describes the output structure, expressed as an object and introduced by the `proto` property;
- a set of rules to be included in the SPARQL query, defined through a set of properties starting with the `$` sign, e.g. `$where` and `$limit`.

JSON queries can be expressed in two different formats, producing coherently the output: plain JSON and JSON-LD. The latter foresees a slightly different syntax (see Listing 6.2) in order to return an output compliant with the JSON-LD specification. This version of the query allows to specify a JSON-LD context, and can be used for mapping the results into a chosen vocabulary. We refer to the documentation[9] for more details.

An interactive Web application called **SPARQL Transformer playground**[10] has been developed in order to quickly test JSON queries. The application is live converting the JSON into a corresponding SPARQL query, so that the user can appreciate every single change. In addition,

[9] https://github.com/D2KLab/sparql-transformer
[10] https://d2klab.github.io/sparql-transformer/

```
{
  "proto": {
    "id" : "?id",
    "name": "$rdfs:label$required",
    "image": "$foaf:depiction",
    "region": {
      "id" : "$dbo:region$required",
      "name": "$rdfs:label$lang:it"
    }
  },
  "$where": [
    "?id a dbo:City",
    "?id dbo:country dbr:Italy"
  ],
  "$limit": 100
}
```

Listing 6.1 – The JSON version of the SPARQL query in Figure 6.1

```
{
  "@context": "http://schema.org/",
  "@graph": [{
    "@type": "City",
    "@id": "?id",
    "name": "$rdfs:label$required$bestlang",
    "image": "$foaf:depiction$required",
    "containedInPlace": {
      "id" : "$dbo:region$required",
      "name": "$rdfs:label$lang:it"
    }
  }],
  "$where": [
    "?id a dbo:City",
    "?id dbo:country dbr:Italy"
  ],
  "$lang": "en;q=1, it;q=0.7 *;q=0.1",
  "$limit": 10
}
```

Listing 6.2 – The JSON-LD version of the SPARQL query in Figure 6.1

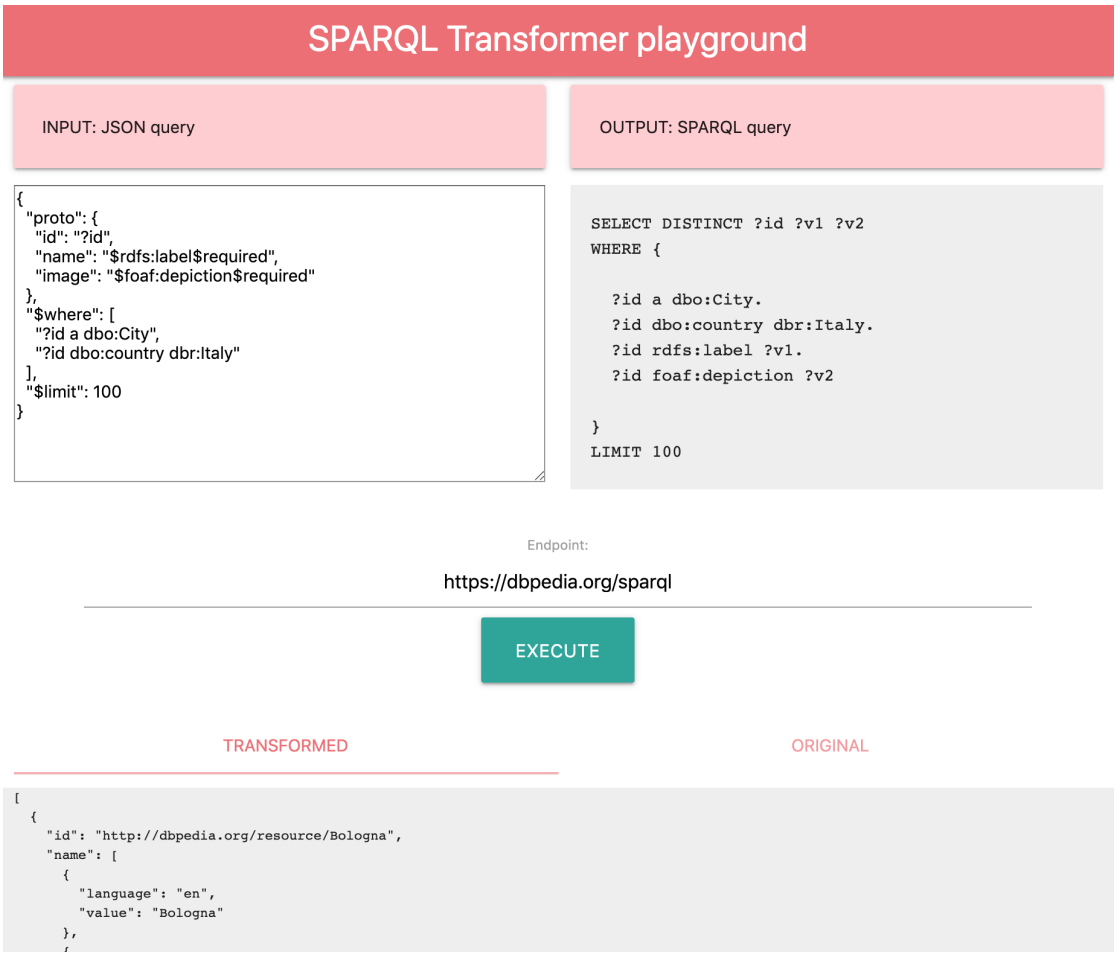


Figure 6.2 – User interface of SPARQL Transformer playground

it is possible to execute the query against a given endpoint, and the user interface offers the possibility of comparing the transformed output with the original one (Figure 6.2).

6.3.1 The prototype definition

By prototype, we mean the common structure that each object in output should respect. It is designed as an ordinary JSON object, in which the leaf nodes will be replaced by incoming data according to specific rules. In particular:

1. **variable nodes**, which start with a question mark "`?`" (like `?id` or `?city`), are replaced by the value of the homonym SPARQL variable;
2. **predicate nodes**, which starts with a "`$`" sign, are replaced by the object of a specific RDF triple;
3. **literal nodes**, which cover all the other contents, are not replaced and will be present as

is in the output, regardless of the query results.

In the transforming process, SPARQL triples will be automatically generated from the prototype. Referring to case 2, the following syntax is used:

```
$<SPARQL PREDICATE>[$modifier[:option]...]
```

The first parameter is the SPARQL predicate, which can be a property or a property path, e.g. `rdfs:label`, `foaf:depiction`, etc. This kind of node will be replaced by the object of an RDF triple having as predicate the one given inline. As subject, the variable of the sibling *merging anchor* is selected if it exists; otherwise, the closer merging anchor among the parent nodes. The merging anchors are all the fields in the JSON introduced with the `id` property. If this variable does not exist, it is set to `?id` by default. In other words, each level in the JSON tree may declare a specific subject through the merging anchor, which will be the subject of all the predicates in the scope. Listing 6.1 includes two merging anchors at line 3 and 7: the former acts as subject of the name, image, and region; while the region name refers to the latter.

The role of the *merging anchor* is crucial for the following steps. In fact, two result objects having the same id will be considered as the same item and their properties will be merged. This will happen at each level of the JSON tree. This controlled way of aggregating SPARQL results ensures a more compact while not less informative output, ready to be used by Web developers.

Both variable and predicate nodes can accept some modifiers appended at the end of the string, separated by the $ sign. These elements are taken in account when writing the SPARQL query. For example, `$required` avoids the predicate to be considered optional (the default behaviour), while `$var` assigns a specific SPARQL variable as object (e.g. `$var:?myVar`), so that it can be addressed in other modifiers. Other possibilities include filtering by language (`$lang:it` or `$bestlang:en;q=1, it;q=0.7 *;q=0.1`) or sample those values (`$sample`).

6.3.2 The root $-properties

A set of $-properties give access to the SPARQL features indicated by their name (`$limit`, `$groupby`, etc). These properties are directly assigned to the root of the JSON query object, and will not appear in the final output. Among them, some additional `WHERE` clauses – in the triple format – can be declared in the `$where` field. The `$lang` modifiers set the language chosen for all the `$bestlang` in the prototype. An exhaustive list of implemented $-properties is reported in Table 6.1.

Table 6.1 – Supported root $-properties

PROPERTY	INPUT	DESCRIPTION
`$where`	string, array	Add where clause in the triple format.
`$values`	object	Set VALUES for specified variables as a map.
`$limit`	number	LIMIT the SPARQL results
`$distinct`	boolean	Set the DISTINCT in the select (default `true`)
`$offset`	number	OFFSET applied to the SPARQL results
`$orderby`	string, array	Build an ORDER BY on the variables in the input.
`$groupby`	string, array	Build an ORDER BY on the variables in the input.
`$having`	string, array	Allows to declare the content of HAVING.
`$filter`	string, array	Add the content as a FILTER.
`$prefixes`	object	Set the prefixes in the format `"prefix": "uri"`.
`$lang`	string	Default language in the Accept-Language standard. [61]

6.4 Implementation

The implementation of SPARQL Transformer relies on three main blocks, each one having a specific function (Figure 6.3).

The **Parser** reads the input JSON query and parses its content. The prototype is extracted and a SPARQL variable – which here acts as a placeholder – is assigned to all the predicate nodes. Contextually, the SPARQL `SELECT` query is generated: the predicate nodes are translated into `WHERE` clauses according to the rules defined in Section 6.3.1 and taking into account the modifiers. The root $-properties are parsed and inserted in the query, which is then passed to the **Query Performer**. This module is in charge of performing the request to the SPARQL endpoint and returning the results in the SPARQL JSON output format. The Query Performer can be replaced by the user with a custom one, for fulfilling different requirements for accessing the endpoint (e.g. authentication) or for integration into more complex environments (as done during the integration with `grlc`).

Finally, the **Shaper** accesses the results, discarding the side information included in the `head` field and directly accessing the bindings. The latter ones are applied to the prototype in sequence, matching the SPARQL variables to the placeholders separately for each binding. In this phase, the data-type of the binding is checked, eventually parsing the value to Boolean, integer or float. When a result binding does not contain a certain value – which happens when the variable is `OPTIONAL` –, the property is removed from the instance. Then, the instances which have a common value for the merging anchor are identified and their properties are compared, in order to keep all the distinct values without repetition. Recursively, the same merging strategy is applied to the nested objects. Finally, they are serialised in JSON and returned as output.

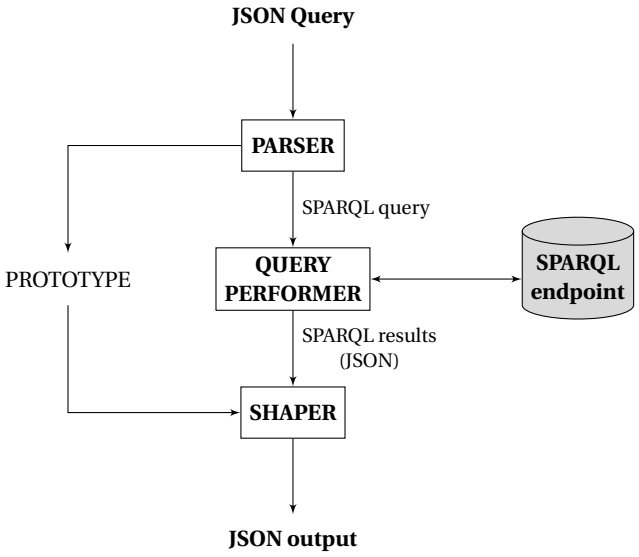


Figure 6.3 – The application schema of SPARQL Transformer

SPARQL Transformer is available in two different implementations in JavaScript[11] and Python[12], published respectively on the NPM Package Manager (NPM)[13] and the Python Package Index (PyPI)[14]. The JavaScript version has been recently converted in an ECMAScript Module [57] and it is designed to both work in Node.js and in the browser. The Python version return a `dict` object, which can be directly manipulated by a script or serialised in JSON.

6.4.1 Integration in `grlc` and Tapas

Thanks the collaboration with the Vrije Universiteit of Amsterdam (VU University), the library is now included in two other software which have been published in the Semantic Web community.

Since version 1.3, SPARQL Transformer is included in the `grlc`[15] framework, which is now able to generate Web APIs from the JSON queries contained in a given GitHub repository. The integration involved the Parser and the Shaper: the former is executed before each access to the SPARQL query, keeping in memory the prototype for being shaped once SPARQL results are back. The JSON query file can include the configuration options for `grlc` in an homonym field. For maximising the compatibility, the options can be specified as a string – following the YAML Ain't a Markup Language (YAML) format – or in JSON. The support to JSON queries

[11] https://github.com/D2KLab/sparql-transformer
[12] https://github.com/D2KLab/py-sparql-transformer
[13] https://www.npmjs.com/package/sparql-transformer
[14] https://pypi.org/project/SPARQLTransformer/
[15] http://grlc.io/

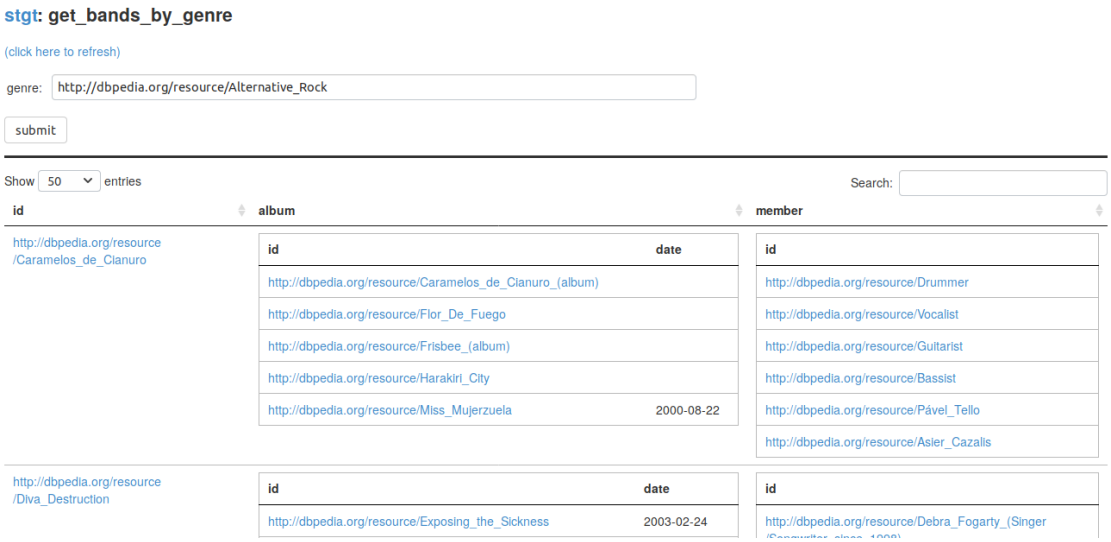


Figure 6.4 – Screenshot of the Tapas interface

includes all the features of `grlc`, such as the pagination and the selection of query parameters. In addition, a `lang` query parameter can change the value of the `$lang` property of the query, allowing the development of multi-language APIs. Further development involved the upgrade of `grlc` to the latest Python version.

Moreover, SPARQL Transformer queries are now also supported by Tapas[16]. Tapas is a small interface module implemented in HTML and JavaScript that reads the specification of an instance of a `grlc` API and turns it into a nice and simple HTML interface. The elements of the API specification are in a straightforward manner transformed into HTML form elements, which the user can fill in to access the service by pressing the *submit* button. Tapas asynchronously calls the API via `grlc` and shows the results at the bottom part of the same page using the YASR component of the YASGUI interface [168] to display the SPARQL query results in a user-friendly manner. We extended Tapas to also support SPARQL Transformer queries and display the results in an equally user-friendly manner. Unlike the flat tables produced by YASR for the common kind of SPARQL results, the nested results of a SPARQL Transformer query are shown as nested tables in Tapas. An example of this can be seen in Figure 6.4, showing a screenshot of the query interface and its results for an exemplary SPARQL Transformer query about music bands, with the nested tables derived from the nested structure of the SPARQL Transformer results. Tapas together with `grlc` thereby allow us to automatically generate an intuitive interface for technically-minded end users just from the query file in a completely general and generic manner.

[16] https://github.com/peta-pico/tapas

6.5 Evaluation

Two kinds of evaluations have been conducted for proving the benefit of this work:

- an experiment for measuring the compactness of the results and the execution time of SPARQL Transformer;
- a survey which measure the preference of users on a system that presents Linked Data query results through SPARQL Transformer, versus another that does so through traditional SPARQL results rendering. This evaluation has been carried by VU University.

6.5.1 Quantitative evaluation

We test the Python implementation of SPARQL Transformer on a set of five queries detailed in the DBpedia wiki[17] in order to ensure a certain generality. The set involves different SPARQL features (filters, ORDER BY, language filtering, optional triples). Those SELECT queries have been manually converted into JSON queries — with 1 or 2 levels of objects in the JSON tree —, making sure that the transformed query was equal to the original one (variable names apart).

Each query has been resolved against a local instance of the English DBpedia[18], with a traditional SPARQL client for the SPARQL queries and with SPARQL Transformer for the JSON queries. Each execution has been repeated 100 times, with a waiting time of 5 seconds between consecutive executions, in order to obtain an average result as much as possible not correlated to any workload of the machine.

The results in Table 6.2 shows that the average execution time of SPARQL Transformer is slightly higher with respect to normal SPARQL queries, never surpassing 0.1 seconds (limit of the instantaneous feeling according to [134]). The difference in percentage, computed as $100 * (t_{sparql} - t_{json}) / avg(t_{sparql}, t_{json})$, do not reveal any regularity in the time increment, even if some patterns suggest that it depends on the number of results and variables for each result. The same dimensions seem to impact also the gap in number of results, smaller in the JSON query responses because of the merging strategy. It is interesting to point out that such difference exists between all valid combinations of values for requested variables and the number of real-world object described. This is evident in the first query, about people born in Berlin, in which the combinations of names in different languages and birth or death date in different formats almost double the number of results. As a consequence, the Prince Adalbert of Prussia[19] appears in 8 distinct – and even non-consecutive – bindings because of its four names and two versions of its death date, correctly merged in the more compact transformed

[17] https://wiki.dbpedia.org/onlineaccess, Section 1.5

[18] The setup of the endpoint on a local machine relied on *Dockerized-DBpedia*, available at https://github.com/dbpedia/Dockerized-DBpedia

[19] http://dbpedia.org/resource/Prince_Adalbert_of_Prussia_(1811-1873)

version. The experiment is further detailed in the GitHub repository[20].

Table 6.2 – Differences in number of results and execution time between SPARQL and JSON queries. For each query, is also reported the number of requested variables.

	N.	N. RESULTS			TIME (ms)			
QUERY NAME	**VAR**	**json**	**sparql**	**diff %**	**json**	**sparql**	**diff**	**diff %**
1. Born in Berlin	4	573	1132	49%	168	101	67	50%
2. German musicians	4	257	290	11%	61	49	12	22%
3. Musicians born in Berlin	4	109	172	37%	59	51	8	14%
4. Soccer players	5	70	78	10%	210	203	7	3.7%
5. Games	2	981	1020	4%	121	70	51	54%

6.5.2 User Survey

In order to evaluate the usefulness of the query results as presented by SPARQL Transformer to potential (technically-minded) end-users and developers and to compare them to a more traditional, table-centric provision of SPARQL query results, we conducted a user survey. We hypothesised that the level of nesting would play an important role, as classical SPARQL results are flat tables whereas the JSON structure of SPARQL Transformer allows for nesting.

We therefore constructed a pair of queries in SPARQL Transformer syntax and its corresponding plain SPARQL version for each of three levels of nesting: no nesting (Level 0), one nested structure (Level 1), and two nested structures (Level 2). These queries are all about bands and their albums and members, and they can be run through the DBpedia SPARQL endpoint. An example of two nested structures as found in Level 2 can be seen in Figure 6.4 (the two nested structures being *album* and *member*). We then ran each of these six queries and stored the resulting JSON files (i.e. the files generated by SPARQL Transformer and the standard JSON files with the original SPARQL results, respectively). Moreover, we also ran these on Tapas to compare the user interface aspects that come with the different representations and nesting styles, and we made screenshots of the result tables. All these files, including queries, their results, and the Tapas screenshots, can be found online[21].

Based on these query results and screenshots, we then created a questionnaire, where we asked the participants for each of the six cases (JSON files and screenshots for each of the three nesting levels) whether they preferred SPARQL Transformer (referred to as "System A") or the classical SPARQL output (referred to as "System B"). The possible answers consisted of the five options *Strongly prefer B* (value -2), *Slightly prefer B* (-1), *Indifferent* (0), *Slightly prefer A* (1), and *Strongly prefer A* (2). We also asked the participants whether they consider themselves

[20] A notebook is available online at https://github.com/D2KLab/py-sparql-transformer/blob/master/evaluation/test.ipynb

[21] https://github.com/tkuhn/stgt/

		rating							
Type	Level	−2	−1	0	1	2	avg.	p-value	
JSON results	0 (no nesting)	6	6	4	13	26	0.85	0.0001980	*
	1 (one nesting)	5	5	3	21	21	0.87	0.000009063	*
	2 (two nestings)	3	9	5	17	21	0.80	0.0003059	*
Tapas interface	0 (no nesting)	4	8	3	19	21	0.82	0.0001275	*
	1 (one nesting)	3	10	2	20	20	0.80	0.0002685	*
	2 (two nestings)	4	7	3	16	25	0.93	0.00003589	*

Table 6.3 – The results of the user survey. The rating describe the preference for our system.

primarily researchers, developers, or none of these two categories, and we asked about their level of expertise with SPARQL and JSON. The questionnaire is fully anonymous and can be found online[22].

We then asked people to participate in this user survey via Linked Data related mailing lists (W3C SemWeb list), and internal group lists of Semantic Web groups at VU Amsterdam and EURECOM, in addition to the SIKS list which groups universities in The Netherlands. The form was accessible for 5 days. In this way, we got responses from 55 participants (40 researchers, 9 developers, 6 others). Their level of expertise on SPARQL and JSON was mixed, with average values of 2.44 and 2.87, respectively, on a scale from 0 to 4. Eight participants had no knowledge of SPARQL at all, while only one participant had no knowledge of JSON.

Table 6.3 shows the results of the survey (the full table can also be found online[23]). We see that we got the full range of replies for all questions, but also that a clear majority prefers our system slightly (1) or even strongly (2). The average values for both types (JSON and Tapas) and all three nesting levels are between 0.80 and 0.93, i.e. close to the value that stands for a slight preference of our system (1) and clearly above the value that stands for an indifference between the two (0).

To test whether the preference towards our system is statistically significant, we used a sign test in the form of a binomial test on the answers that were positive (preference of our system) or negative (preference of the existing system), excluding the zero cases (indifference). This test, therefore, does not take the distinction between slight and strong preference into account, but only which system was preferred. The final column of Table 6.3 lists the p-values of this test, showing that the effect is highly significant for all six cases.

The results, however, do not support our hypothesis that the level of nesting has an effect on the preference for our system. Throughout all nesting levels, the users expressed clear and significant preference for our system, but this preference did not increase with increased

[22] https://github.com/tkuhn/stgt/blob/master/eval/questionnaire-form.md
[23] https://github.com/tkuhn/stgt/raw/master/eval-results/questionnaire-results.ods

nesting levels.

6.6 Conclusion and Future Work

SPARQL Transformer offers to Web developers a different way of approaching RDF datasets. The adoption of a novel JSON format for defining both the query and the template makes it possible to realise self-contained files. When collected in a GitHub repository, these files can be easily transformed into Web APIs with `grlc`, completing the decoupling between query, post-processing and consumption in the application, and query results can moreover be presented in a simple and user-friendly manner via Tapas. The evaluation reveals that the restructuring and merging pipeline of SPARQL Transformer has an important impact in making the SPARQL results more usable and understandable by humans.

Differently from other works, SPARQL Transformer allows developers to use one single file for querying and mapping, and even with some limits – i.e. not being as expressive as SPARQL – can be of benefit for fast prototyping of web application.

Further development can improve SPARQL Transformer in order to fulfil a wider range of needs. The query support can be extended to other SPARQL operations, like ASK, INSERT and DELETE, going towards the realisation of full REST APIs on top of SPARQL endpoints. Aggregate functions (e.g. COUNT, SUM) should join the set of available features in the near future.

Future work will further investigate the use of JSON frames, in order to extract the Shaper component from the library and make it available for standalone use.

Currently, the JSON syntax does not foresee any standard way for representing dates, which are therefore represented as plain strings. Alternative representations for dates should be found taking into account developer requirements, even listening and involving them in the final decision. Possibly, the solution should also involve other related data-types, like `xsd:gYear` or `xsd:duration`.

We plan to run another evaluation of this work, this time focused on the creation scenario, consisting in an interview on query writing with SPARQL Transformer and on API management with `grlc`.

Finally, we are currently planning to offer more customisation possibilities to users. Some examples include the choice of a different merging anchor (currently forced to `id` or `@id`); the possibility of ignoring language tags in the results (avoiding the presence of a language-value object); and the chance of distinguishing between Internationalized Resource Identifier (IRI)s (as resource references) and IRIs in lexical forms.

Part II

Exploit the Music Knowledge

The main inheritance of Part I consists in a big Knowledge Graph specialised in classical music, which relies on a FRBRoo-like ontology and on a set of controlled vocabularies for representing music-specific terms, like genres, keys, instruments, etc. In Part II, the reader will find some studies, algorithms and applications that rely on the DOREMUS graph, with the goal of exploiting classical music knowledge in fields like visualisation, AI and music recommendation.

Data about user preferences are not involved in this research. In this context, our study on recommendation does not include one of the core part of RS: personalisation. The reasons about this choice are: 1. the difficulty in finding user-related data in the context of classical music, and 2. different motivations pushing towards algorithms supporting recommender systems, providing related items according to a similarity ranking.

If a RS has no specific knowledge about the active user, then it can only provide him with the same recommendations that would be delivered to an “average" user. Sometimes, the current track is used as seed item for triggering the recommendation, similarly for what happens in Spotify’s Radios. Such kind of recommendation are not only important for facing the *cold-start problem*[24], but also for developing systems that address a wide public rather than individual listeners, like for concert programming, radio broadcasting or editorial playlists realisation.

For the scope of this dissertation, we will abandon the FRBRoo distinction between the terms *work* and *expression*, always using the word *work* for referring to a musical piece.

[24] In the cold-start problem, recommendations are required for items that no one has yet rated or for users that did not have enough previous preferences set.

Chapter 7

Related Work

In this chapter we will outline some related work to the main topic covered in this Part of the manuscript, namely knowledge graph embeddings and recommender systems. In particular, some research about music recommendation exploiting metadata is introduced in Section 7.1. An overview of knowledge-based recommender systems based on Linked Data (LD) is present in Section 7.2, extended in Section 7.3 with approaches based on embeddings. Section 7.4 includes some solutions for context-based recommendation involving music and/or knowledge bases. Finally, in Section 7.5 we report a summary of Music Information Retrieval (MIR) research that rely on symbolic music (in particular MIDI).

7.1 Music recommendation and metadata

In the introduction of *Recommender Systems Handbook* [166], the authors identify six different kinds of recommender systems (RS), among which the most popular are undoubtedly Collaborative Filtering (CF) and Content-base. The former recommends items which have been liked by similar users – i.e. users that share with each other a certain number of liked items. The latter recommends items that are similar to – i.e. have similar features with – the ones the user already interacted with. Both classes of RS suffer the so-called cold-start problem: when new users join the system, the algorithm has no data for computing the recommendation. For this reason, this systems are often used in combination with other kind of RS, for example Knowledge-Based ones, which rely on some domain-specific Knowledge Base (KB) [24].

In music RS research, the knowledge available in the web is considered a valuable source of information [93], as in the whole field of MIR [204]. Even if less used than acoustic features, editorial metadata (EM) – intended as the set of expert-defined information (composer, genre, etc.) – are commonly considered as crucial resources for recommender systems [178, 183].

An experiment of content-based recommendation based uniquely on EM is reported in [21].

In this work a tag cloud – including information as genres, record labels, years of release activity – is produced for each artist and vectorised through latent semantic analysis, so that distance-based rankings can be applied for recommendation. Some researches suggest that music specific features – even not expected ones like the musical key – can have an interesting impact in computing the similarity between artists [79].

7.2 Recommender Systems and Knowledge Graphs

In late 2000s, the first meeting between LD and RS gave to the latter two crucial benefits, which have been exploited in the following years. On one hand, the Web of Data provided the access to a big amount of structured knowledge, together with practices and tools for representing and interlinking different existing sources. On the other hand, the graph structure itself of LD were found to be apt for computing the recommendation, feeding the emerging research in graph-based RS [63].

LD-based RS implement different strategies and approaches. In [205], association rules are used for recommending relevant properties when editing an item in Wikidata, based on the class of the item and on co-occurrent properties on inserted ones. LD are represented as 3-dimensional tensors in [52], where adjacency matrices for each property are combined for computing their similarity. The approach is extended by including a CF component in [90].

Oramas et al. [142] exploit semantic technologies for realising a graph containing information about user interactions – of type <*userX, downloaded,itemY*> – and enriching it with information coming from external sources, such as WordNet[1] and DBpedia. On this extended graph, features are computed by a neighbourhood mappings, taking in account distances in number of hops, number of connections and shared structures between items. Recommendation are computed by ranking items on the base of Euclidean distance of their feature vectors from the user one.

The distance between nodes in the KG has been used also in [10]. The involved graph is the result of the interlinking of different knowledge sources, such as DBpedia, Last.fm[2], MusicBrainz and AcousticBrainz. Than, similarity between artists are computed according to the Maximum Degree Weighted (MDW), which allow to reduce the impact of links to very large categories (i.e. Living People) with respect to more significant ones. The outcome is a web application which shows for each seed artist a graph of the more strongly connected ones.

[1] https://wordnet.princeton.edu/
[2] https://www.last.fm/

7.3 Recommending with Embeddings

Graph embeddings are the result of the transposition of word embedding techniques – notably word2vec [127] and GloVe [152] – to networks. Graph embedding algorithms produce a mathematical representation of the content of the graph, which are much more compact than other kinds of representation (e.g. adjacency matrix) and consequently easier and faster to process with Machine Learning (ML). The effectiveness of these techniques makes them very popular in different applications, from classification to recommendation, with an interesting number of algorithms developed for their computation [68].

In 2012, Perozzi et al. published **Deep Walk** [154]. The core idea of this work consists in the use of random walks in the graph in order to generate sequences of nodes. The number and the length of link paths between two nodes, impacts on the probability of those two nodes to be selected together in the random walk. In other words, the more two nodes share connections and the less edges compose those connections, the more those nodes will appear together in several walks. According to the intuition of the authors, we can deal with nodes in sequences as they are words in sentences, so it is possible to apply word embedding models to those sequences. The result is a vector space in which distances in the graph are kept.

DeepWalk has been extended by **node2vec** [70] with the inclusion of two parameters P and Q, which rule on the generation of random walks. In particular, the parameter P impacts on the probability that the random walk immediately revisit the previous node. The parameter Q controls the probability that the random walk moves towards increasingly further away nodes, enabling to discover peripheral parts of the graph. In other words, higher values of P promote random walks that explore a local neighbourhood around the starting node, while high values of Q encourage walks that cover wider areas of the graph. Node2vec can be applied also to weighted graphs, in which the weight of an edge affects the probability that it participates to the walk.

Different embedding strategies have been implemented in rdf2vec [169], entity2rec [147], graph2vec [133], and many others[3].

Embeddings are widely applied to recommender systems research [147, 148, 171], including music recommendations [58, 116, 131, 143]. Sometimes, the embeddings are computed on lists of textual properties or *tags*, which commonly includes indistinctly genres, moods, contextual information [116, 131], proving their effectiveness for coldstarting new songs.

Apart from graph embeddings, other kinds of embeddings have been studied in MIR in fulfilment of different tasks, such as genre classification [97].

[3] Regularly updated lists are available at https://github.com/MaxwellRebo/awesome-2vec and https://git.io/fjwx6

7.4 Context-based recommendation

In order to generate more relevant recommendations for a particular moment or situation, context-aware recommender systems (CARS) try to exploit the information related to the user context [7, 88]. This information can include time, location, purpose of the user, current mood, people around him, weather condition. Often, context-based components are used in combination with other kinds of RS in a Hybrid System. Taking in account the user context is among the challenges for next generation of music RS [177].

In [144] spatial and temporal context supports a knowledge-based RS of films, by pre-filtering the target on current shows in the area and re-ranking recommendation taking in account the proximity and characteristics of the cinema. In [46], contextual, collaborative, and content information are combined with deep learning for recommending next step during a trip. In VenueMusic [35], the acoustic features of songs commonly played in a given venue type – gym, restaurant, shop, office – are used to train a CARS recommending songs that match the venue atmosphere.

The strategy of a CARS may rely on intermediate features, which act as bridge between the domain of items (music tracks) and the domain of context (time, place, situation). The choice falls on the emotion in MusicSense [28] – which recommends relevant music according to the Web page the user is reading, matching them both to a common emotion – and in COMUS [182], in which the desired emotion is directly asked to the user.

In order to recommend relevant songs according to touristic venues, Recommendation based on Points of Interest (PoIs) are often realised using mappings between PoIs and songs. These mappings can be performed by automatically discovering links on expert-defined sub-graphs of DBpedia [87] or with a manual annotation made by volunteers [25]. Suitable recommendation when driving a car are studied in [13, 81].

7.5 Symbolic Music for MIR

Apart from being codified in audio tracks, the music content can also be represented as *symbolic music* [56], whose definition includes all notation-based formats, from scores to digital encoding formats, including MIDI. Instead of directly describing the sound, those formats contains the information that is required for producing it, in other words the set of “instructions" for playing the music work.

An extensive survey about genre classification based on symbolic music [44] report an interesting number of works which may rely on different sets of classes, on monophonic MIDI rather than polyphonic ones, or on genre-specific datasets (e.g. folk music). However, some interesting methods based on machine learning have been carried out. In [119], a unsupervised

k-nearest neighbours (kNN) algorithm is applied for genre prediction from MIDI. This work is extended in [31] using linear discriminant classifiers (LDN) and combining MIDI and audio features. In [118], four different data sources – audio, symbolic music, lyrics – are studied both separately and in combination.

Nowadays the computing power of modern machines made it easier respect to the past to run algorithms on directly on audio signal. However, symbolic notations is still largely used in those field which imply that machines learn to understand the “instructions", as for example automated music generation [41, 82, 170, 203].

7.6 Conclusion

A wide literature studies the application of Recommender Systems to music, to KG and to both music and KG. As stated in Chapter 1, the application of RS research to Classical Music is still at an early stage, and indeed none of the mentioned works is focusing it.

Symbolic music has been exploited in different works and for different tasks, but it has never found a common point with graph-based technologies.

For this reasons, we are discussing in the following of this Part some embedding strategies, which will be applied to Classical Music recommendation and to symbolic music representation.

Chapter 8

Embeddings and Similarity

Which artist is the most similar one to Vivaldi?

An artist, and in particular a musician, can be described using different features: as a person, one can use the date and place of birth and death; as a composer, once can consider the musical genre, the foreseen MoP, the key, etc. of the compositions; as a performer, one can think of the function and role of the artist, or the actual MoP played during the performance. This information is representative of the career of an artist.

We hypothesis that this kind of metadata plays an important role when one has to compute similarity between artists. For example, two musicians considered related by musicologist community, like Antonio Vivaldi and Tomaso Albinoni, share in fact many of those features: they lived in the same period, mostly in Venice, they are both violin performers and composers of an interesting number of concerts and sonata for string instruments. The impact of those metadata is partially revealed by previous works [116, 131], in which the involved feature sets are nevertheless flat collections of textual tags, rather than structured taxonomies as the DOREMUS vocabularies (Chapter 4). The vocabularies and the well structured information in the DOREMUS dataset can foster new directions of research and new questions. *Which features are more important for comparing two artists? Which ones for comparing two works? How can we mathematically represent those features, so that the mathematical similarity mimics human-perceived similarity?* Not all musicians are the same: some of them are instrumentalists and can be compared by played instruments, while others are just known as composers and this kind of comparison cannot be performed. *How can we measure the similarity of two elements when the overlap between their feature sets is not perfect?* In this chapter we start to investigate these topics, which will be further developed in the next ones.

Graph structures are particularly suitable for discovering connections between nodes. This is valid also for the DOREMUS KG, in which entities are linked through lower-level nodes. Two artists can share the same played instrument, two composers the same genre, two composi-

tions the same key. In turn, instruments, genres, keys are connected each other by hierarchical (e.g. `skos:broader`) or horizontal links (e.g. `skos:related`).

Our strategy relies on the combination of partial embeddings. We realise two levels of vectors:

- the *feature embeddings* are computed at the level of the single features (i.e. genre, instruments, etc.) directly on the knowledge graph;
- the *entity embeddings* represent the main entities (artists, works) and are the result of the combination of other embeddings of 1st and/or 2nd level.

This approach has several advantages. The individual contribution of each property in the feature vector is maintained, easy to identify among the embeddings dimensions, making it easy for a human to analyse the results. Having feature-specific vectors give the possibility of reusing them in different contexts – i.e. similarity of composer, performers, works, etc. As a consequence, recomputing the whole embedding for new artists and works is not required, skipping in this case the most computationally expensive task.

The contributions presented in this chapter involves approaches for computing embeddings at feature level (Section 8.1), combining them at entity level (Section 8.2), and computing the similarity between those embeddings (Section 8.3). Some conclusion is outlined in Section 8.4.

This research has been published for the first time in [107, 109].

In order to avoid any ambiguity in the nomenclature, we will make distinction between the terms:

- **feature**, a defined “human-understandable" property, with a name (i.e. genre, composition date, etc.) and a value (string, id, number, date);
- **dimension**, a single numerical element of a vector; several dimension can contribute in representing a single feature.

8.1 Feature Embeddings

The DOREMUS dataset contains information about **MoPs**, **genres**, **keys** and **functions**, defined through the controlled vocabularies, which include also hierarchies and relationships between items – e.g. *violin* is in the family of *strings*, *gospel* is related to *spiritual*, etc. The dataset has a good coverage also for **dates** (birth dates, concert schedules) and **places** (i.e. theatres, concert halls, birth places), the latter ones being linked to GeoNames [200]. In this section, we describe the strategies applied for the generation of their embeddings.

8.1.1 Music Embeddings

What are the closest keys to *C major*? Is it possible to decide which instrument between the *cello* and the *oboe* is more similar to the *clarinet*? The answer to those questions would provide application in different fields, from musicology studies to the development of specialised recommendation systems. Graph embeddings are a way to achieve those results.

For each of the music feature involved (MoP, genre, key, function), we can access to two kind of information, contained in as many distinct sub-graphs of the DOREMUS KG:

- the *graph of vocabularies*, which defines structural and semantic connections between entities, such as hierarchies, `owl:sameAs` links, properties in common, specific music properties – i.e. relationships between keys. Given that this information has been redacted by human experts according to logic or historical reasons, it represents the involved concepts for what they are;
- the *graph of usage*, which includes all the usages of the vocabularies in the DOREMUS dataset. We considered musical works for the genre and the key, castings and performances for MoPs, composition and performance events for functions. This information represents the involved concept for how they occurs in the reality of compositions and performances.

We computed the embeddings using *node2vec* [70], giving the 2 sub-graphs as input. We arbitrarily set to the graph of vocabularies a weight 6 times bigger than the graph of usage, in order to counterbalance the richly larger number of triples[1] and avoid to nullify the contribution of each one. After a post-processing step that removes all the literals and the extra nodes involved, a L2 normalisation is then applied in order to have values in {-1;+1}[2]. The process is represented graphically in Figure 8.1. The dimensionality of all embeddings is 100.

In order to appreciate the effectiveness of this strategy, we used t-SNE [194] for visualising the embeddings on a 2D image. As an example, Figure 8.2[3] shows the vector space of MoP and genre. By observing the groups of closer entities, we can clearly identifies clusters in both plots, which was manually annotated in the figure. About the MoP, it is interesting to observe that even if the hierarchy of the instrument families is preserved, the usage graph strongly influenced the result, by reflecting the differences of instruments in genres and periods. This is the case of the orchestra instruments group, which puts the *violin* closer to his orchestra colleague *clarinet* than to its 15^{th}-century relative *tromba d'amore*. The clusters in the genre vector spaces reveals similar behaviours, with a natural commingling given by the absence of

[1] More than 16 millions triples against around 100.000 ones for the vocabulary graph.

[2] The normalised vector is $y = x/z$, where $z = ||x||_s = \sqrt{\sum_{i-1}^{n} x_i^2}$.

[3] Higher-resolution images are available at https://github.com/DOREMUS-ANR/music-embeddings/tree/master/img

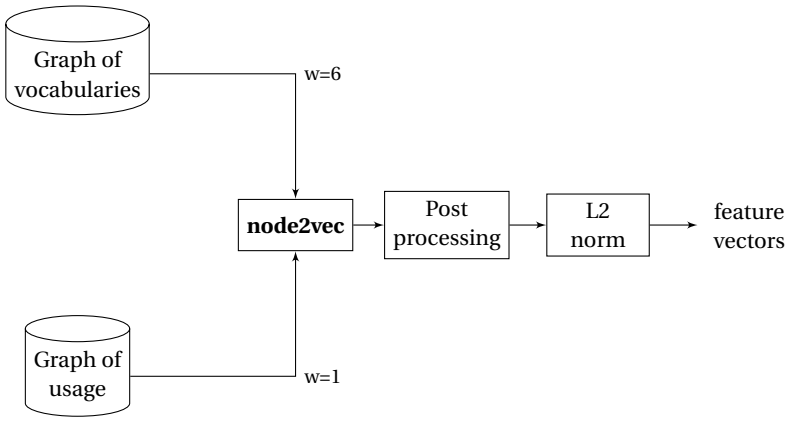


Figure 8.1 – Feature embeddings generation schema

clear genre categories, having elements like *Gregorian music* as junction between medieval and religious genres.

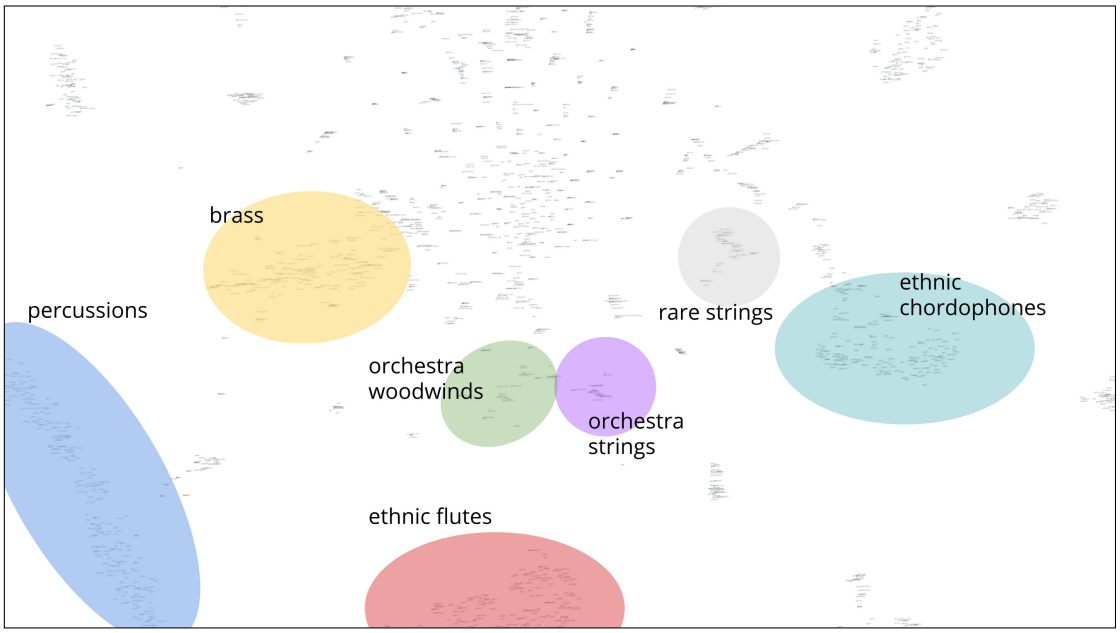
Considering the relevance outside the scope of this research, we published the embeddings for further usage at https://github.com/DOREMUS-ANR/music-embeddings.

8.1.2 Years and Places embeddings

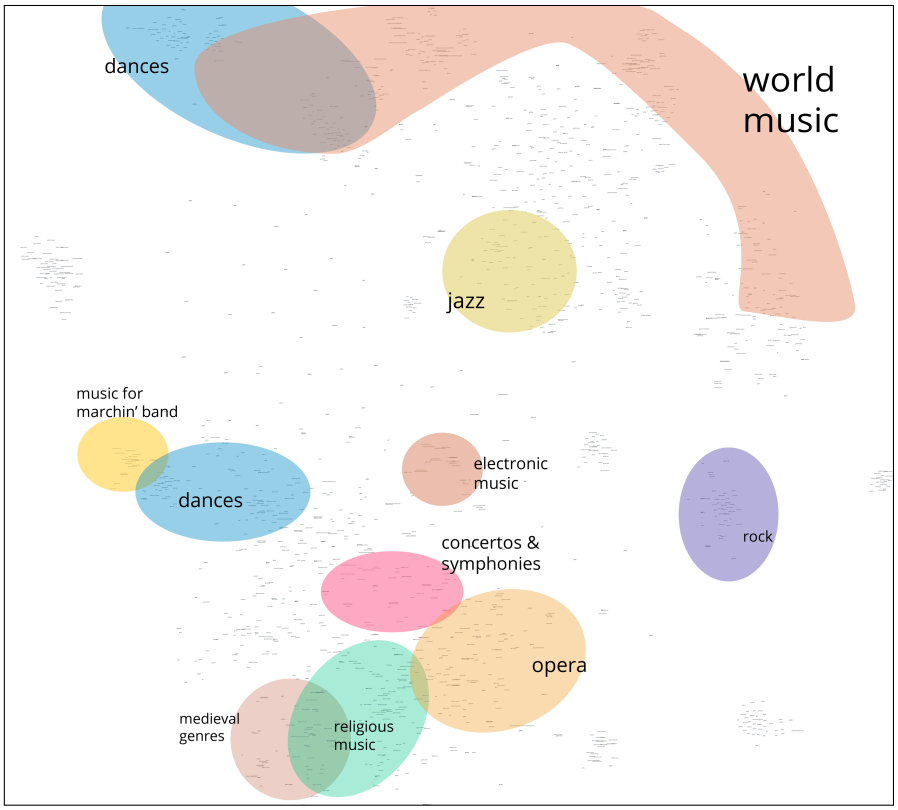
Artistic and cultural movements have always seen some kind of aggregation, in time and space. In music for example, the Classicism is commonly ascribed to a group of musician (Mozart, Haydn, Beethoven an others) which were all active in Vienna between 1730 and 1820. We considered appropriate to study the impact of those features from a quantitative point of view.

For dates, one solution is to map the full range of time involved in the dataset – from the I century B.C. to nowadays – into $(-1;+1)$. However, this strategy has two main drawbacks. First, it produces a mono-dimensional result, hard to compare together with multidimensional embeddings. Then, it represent the time linearly, while we know that changes – i.e. in politics, lifestyle, society, culture – are as bigger and faster as history reaches modern days. In [18], an embedding approach for time is proposed, which relies on word embeddings computed on event descriptions. Years are matched with their proper Wikipedia page, which contains textual descriptions of the events occurred in that year. This text is automatically annotated using DBpedia Spotlight [48] and embeddings are computed using the skip-gram algorithm. This strategy has been applied for collecting year embeddings.

In the DOREMUS dataset, GeoNames has been chosen for defining places, in order to include a interest set of links which connect every place to a city, region, country, continent. We used the embeddings for GeoNames entities published in [89]. These embeddings involve all the entities at the level of populated place (city) or upper (country, region, etc). The



(a) MoP



(b) Genre

Figure 8.2 – 2D representation of the vector space of *medium of performance* (a) and *genre* (b), with some recognisable clusters.

authors computed random walks on a weighted graph of places, in which all the nodes are connected with their neighbours in space (calculated on latitude and longitude) and the weights represents the distance between two nodes. In other words, the transition probability between two nodes is inversely proportional to the spatial distance, thus places closer in space are more probable to appear together in the same walk. The Glove algorithm [152] is then applied on these walks for generating the embeddings.

All embeddings have dimensionality 100 and a range of values that falls in $(-1; +1)$.

8.2 Embeddings Combination

More complex entities such as artists and works can be represented as combination of their corresponding feature embedding. For each entity type, a set of interesting features is chosen:

- for the artists: birth and death date, birth and death place, played instruments or voice range (MoP), embodied functions, and genre, casting and key of its compositions;
- for the works: composition date, genre, casting, soloist instrument, key, composer.

In order to create the embeddings of a specific entity, all the different values of each involved feature are retrieved from the knowledge base. Each work or artist can have zero, one or more values for each feature. Multiple occurrences of the same value are counted separately, so that the genres, keys, casting for which a composer is specialised are more represented also in the vector. Feature by feature, the embeddings are merged by averaging element-wise the feature embeddings: for example, the average vector of the genre of Giuseppe Verdi is the average vector of genre vectors representing lots of operas, some overtures, few requiem mass, etc., having the same dimensionality of the source embeddings.

The artist vector is realised through the concatenation of each average feature vector. The artist vector will then be concatenated again to other feature vectors, to realise the work vector. The same approach of cascade concatenation can potentially be extended to other cases, like single performances, concerts or playlists.

In some cases, there are no results for a certain feature: this can happen for unknown values – i.e. the birth date of a medieval artist – or not applicable properties to a particular entity – i.e. normally operas have not a specified key, some artists are just performers and not composer or vice versa. As a consequence, an array of null values of the same dimensionality of the expected vector takes the place of these missing results.

In order to speed up the computation and stemming the curse of dimensionality, a Principal Component Analysis (PCA) reduction from 100 to 5 dimension is realised at feature embedding level before the combination process. Thus, entity embeddings result as the concatenation of

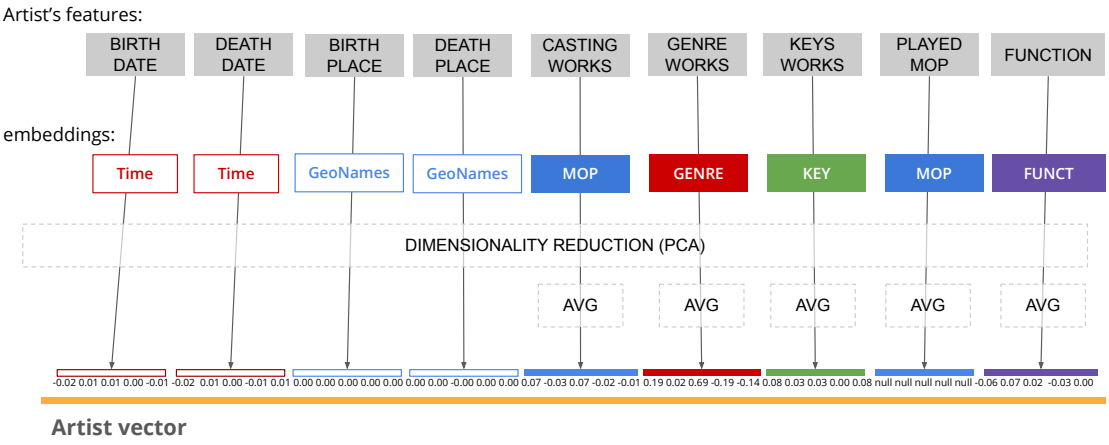


Figure 8.3 – Partial embeddings combination schema

5-dimensions feature embeddings – with a final total dimensionality of few decades – , rather than 100-dimensions ones, which would have much increased the final dimensionality.

Figure 8.3 sums up the key steps of the method. The Figure remarks that the different contributions of each feature are kept distinct in the output vector.

8.3 Euclidean Similarity with Penalty

We want to compute a similarity score between a seed entity s and a target one t, in order to rank the entity more similar to s. As seen in several related works [58, 116, 131, 143, 183], we opted for the Euclidean distance, which provide a way of assigning weights to the different dimensions.

As seen before, their vectors can contain some null dimensions, which we do not want to consider. Therefore, we remove from both vectors all the dimensions that are null for any of them. On this shorter version of the vectors, we compute the Euclidean distance d:

$$d(s,t) = \sqrt{(s-t)^2} = \sqrt{\frac{1}{N}\sum_{x}(s_x - t_x)^2} \tag{8.1}$$

with x being a specific one among the N dimension, considering only the valorised ones. s_x and t_x are value of x respectively for the seed and the target vector.

Defining d_{max} as the distance between an all-ones vector and its additive inverse one, we

compute the score according to the following formula:

$$similarity(s,t) = \frac{d_{max} - d(s,t)}{\mid d_{max} \mid} * (1 - penalty(s,t)) \tag{8.2}$$

The *penalty* is the ratio between the number of dimensions missing in t but present in s and the number of all the dimensions in s. Following the Iverson bracket notation[4] [95]:

$$penalty(s,t) = \frac{\sum_d [s_d \neq null][t_d = null]}{\sum_d [s_d \neq null]} \tag{8.3}$$

The presence of the penalty gives to the similarity measure a direction, potentially producing different similarity scores when s and t are inverted. Given that the data is not exhaustive, the decision has been to not penalise the dimensions that are missing in the seed, because no value would be fair for performing the comparison. On the other side, the penalty is necessary for avoiding that the similarity score gets higher values when comparing less features, making the absence of some dimension an advantage.

After some empirical experiments, this similarity measure return encouraging results, summarised in Table 8.1. The table contains the most similar artists or works to the given one, computed on the whole DOREMUS KG. For the artists, to Vivaldi corresponds other baroque artist, while opera composer are the most similar to Verdi. Around Schubert there are very popular pianists, among which his idol Beethoven. The work similarity works well by matching similar genres, while it seems to biased by the key dimension for *Für Elise*, proposing only works in *A minor*.

We eventually need to increase the influence of some dimensions over others according to the requested impact of each of them in the recommendation, prioritising one or another feature.In order to do this, it is possible to include a weight vector w in the (8.1), introducing the weighted Euclidean distance:

$$d(s,t,w) = \sqrt{w(s-t)^2} = \sqrt{\frac{1}{N}\sum_x w_x (s_x - t_x)^2} \tag{8.4}$$

In this new scenario, (8.1) can be seen as a sub-case of (8.4), in which w is a vector of ones w_{flat}.

[4] The logical proposition inside square brackets is converted into a number that is 1 if the proposition is satisfied, and 0 otherwise.

seed (artists)	top 3	scores
A. Vivaldi	T. Albinoni	0.996
	F. Geminiani	0.995
	A. Scarlatti	0.994
G. Verdi	G. Bizet	0.991
	R. Leoncavallo	0.991
	B. Godard	0.990
F. Schubert	W. A. Mozart	0.987
	F. Chopin	0.987
	L. van Beethoven	0.987

seed (works)	top 3	scores
Für Elise	Bartholdy's Songs without words	0.996
	Rondo for Guitar	0.990
	Schumann's Toccata Op. 7	0.987
La Walkyrie	Das Rheingold	0.999
	Les Troyens	0.998
	Lohengrin	0.998
The 4 seasons	Vivaldi's 12 concertos	0.999
	Bach's Concerto BWV 1057	0.998
	Leclair's 6 concertos	0.997

Table 8.1 – More similar artists and works according to the similarity measure (8.2), among a pool of 1396 artists and 2563 works

8.4 Conclusion

We have presented a content similarity approach for classical music, which relies on an embeddings combination strategy made by 2 steps:

- a computation at level of single feature, extracted from a semantic knowledge base and based both on structural relationship and usage in the data;
- subsequent combination of the single feature embeddings into the vectors of more complex entities.

Then we defined a similarity measure that apply the Euclidean distance with a penalty correction in order to deal with missing information in the vector. An empirical experiment shows at the same time some promising results and the limit of the approach, which needs to be tuned.

The embeddings about time and places may be more representative if computed directly on the work dataset, following the experiences of the other features. Moreover, some improvement may be obtained by including new features in the embedding vector, like textual information (title, description).

This measure allows to define weights to the different dimension of the embeddings, in order to promote some specific features in the similarity measure. A deeper study about how to find the best weights is included in Chapter 9.

Chapter 9

Playlists and Weights

Human experts have always played a central role in drawing up lists of musical works which can serve different purposes. Exemplary cases are the artistic directors that write the program of a concert on the base of the available casting and the allocated time; the editorial board of a radio that choose the track to play in their archive; or finally the editors of mainstream playlists in streaming services. Rather than individual listeners, these three activities target a broad public, that shares a single space and time for the concert, multiple spaces but a single time for the radio, multiple spaces and times for the playlists. In addition, the public of streaming playlists has the power of skipping tracks and changing or randomising their order. As a consequence, experts will produce those lists differently according to the goal.

Our intuition is that there are some hidden rules which are followed in playlist creation and decide which artist or which work should follow another. These rules are directly derived by the knowledge of experts themselves, which may apply them consciously or not, and which may not be able to even describe them. We believe that these rules can be extracted by studying the content of the playlists.

In this chapter, we propose a first contribution for the understanding of the editorial playlists realisation, with the long-term goal of producing recommendations for a wide audience alongside the ones based on user preferences. Our approach relies on a vector representation of music-related concepts (artists and works), in a format which allows to value the contribution of single lower-level features (genres, instruments, keys, etc.). We aim to address the following research questions: *Which features are involved in the process of a playlist creation? Are they the same for different targets (playlists generation, radio broadcasting, concert programming)? Can they have an impact on item ranking in automatically generated playlists?*

The homogeneity of the music signal, sometimes in combination with some tags, has been used as similarity measure in playlist generation [22]. We will similarly try to study homogeneity of metadata in playlists, in order to distinguish between important and less important

features.

In Section 9.1 we introduce classical music datasets made of playlists and concerts, and we study them in terms of homogeneity of single vector dimensions. The results of the study are used for producing weights for tuning a ranker based on Euclidean distance, as described in Section 9.2. A user evaluation is reported in Section 9.3. Finally, we conclude and outline some future work in Section 9.4. Two side works are presented afterwards, respectively focusing on playlist titles (Section 9.5) and playlist emotions (Section 9.6.)

9.1 Real world data: concerts and playlists

As already mentioned in Section 1.1.2, specialised datasets in classical music do not exists for tasks like recommendation and playlist compilation. In order to make possible any evaluation of our research, we decided to rely on real word data, which have been collected and interlinked to the DOREMUS KG.

9.1.1 Dataset description

Our strategies are tested on four different collections, which cover the goals specified in the introduction of this chapter (concert programming, radio broadcasting, playlists generation).

Philharmonie concerts. A set of 186 classical concerts held at the *Philharmonie de Paris* between 1995 and 2017, chosen among the ones with at least 6 different works played. This dataset is extracted directly from the DOREMUS KG.

Itema3 concerts. A set of 414 classical concerts recorded by Radio France between 2011 and 2015, chosen among the ones with at least 6 different works played. This dataset is also extracted directly from the DOREMUS KG and does not require interlinking.

Radio France web-radio. 105 slots of 3 hours that are broadcast in the web-radio channels of Radio France.[1] The slots belong to 5 different channels (*Classique Easy*, *Classique Plus*, *Concerts*, *Contemporaine*, *Jazz*) and have been realised by 3 experts of the radio network.

An interlinking process with the DOREMUS dataset is performed on the base of the composer name and the title, the latter often containing other kind of information (key, casting, opus number, etc.), like in *Sonate n.3 en La Maj op 69 pour violoncelle et piano*. The interlinking follows two steps: 1. the composer are identified through exact match on the label,[2] in order to limit the number of candidates to his compositions; 2. the title of the work is tokenised through empirical methods based on regular expressions and the use of the controlled vocabularies, in

[1] https://www.francemusique.fr/webradios

[2] Note that the DOREMUS dataset contains multiple alternate forms for each artist name.

order to separate the different parts of the string. These tokens have a type which reflect its content – opus, catalogue, key, etc. – and corresponds to properties obtainable by querying the DOREMUS KG.

The tokens are used as input to a variant of the Extended Jaccard Measure [190], designed in order to manage typed tokens and assign them different weights. This allows to prioritise matches on very discriminant token (i.e. the catalogue number) and being more relaxed on those that may differ (i.e. equivalent casting statements). Using again Iverson bracket notation as in (8.3), we define this measure between two records a (from web-radio) and b (from DOREMUS):

$$f(a,b) = \frac{w_{title} sim(a_{title}, b_{title}) + \sum_t w_t [a_t = b_t]}{w_{title} + \sum_t w_t} \tag{9.1}$$

In the equation, t are the types of token which can be found in both records[3], a_t and b_t the value of the token of type t in the two records, w_t are weights empirically defined in order to have the best results[4]. The similarity between two titles is the best value among different string similarities, all computed through Levenshtein distance, which involves:

- on the web-radio side, the record title
  - as it originally is;
  - after token extraction, so without opus number, order number, etc.;
  - alternatively cropped at the first occurrence of a special character (:, -, ,), which normally introduces subtitles or movement names;
- on the DOREMUS side, all the alternate titles available.

In a nutshell, matched values and string similarity – in the numerator of (9.1) – are normalised by the maximum obtainable result, in order to have a score $f \in [0,1]$. The match is considered reliable if the score is higher or equal to 0.7 ($f >= 0.7$).

Finally, only the lists with a number of elements equal or greater than 5 are taken into account.

Spotify editorial playlists. 65 playlists realised by the editorial team of Spotify, which contains in average around 50 tracks each. The playlists have manually been selected from the *Classical* section of the *Genre and Mood* page[5] and the metadata (artist, title) have been collected through the Spotify Developer API. The title and the composer name (extracted from the `artists` field that groups indistinctly authors and performers) have been used for the interlinking, apply the same strategies of the web-radio dataset with a 45% of coverage.

[3] Types presents only in a or in b are not taken in account because considered unknown.
[4] opus number = 90, order number = 70, catalogue statement = 15, key = 18, title = 10.
[5] https://open.spotify.com/view/classical-page

Table 9.1 summarises some statistics about those datasets. From now on, we will refer to any list of works coming from the dataset as a *playlist*, while the differences between concerts, radio programmes and playlists will be remarked when required.

		works			n. of extracted
	n. of playlists	per playlist	total	distinct	sub-groups
pp concerts	186	9.5	1773	1246	805
itema3 concerts	624	11.5	7166	7166	4046
web-radio	50	22.6	1128	893	878
spotify playlists	65	28.9	1880	1432	1555

Table 9.1 – Statistics about the datasets.

9.1.2 Dimensions homogeneity inside playlists

The embedding representation of musical works open possibilities about studying the internal configuration of the playlists from a quantitative point of view. In particular, it is possible to compute the mean and the standard deviation of each playlist referred to each single dimension. This would give a numeric information about how a playlist is homogeneous in terms of composers, period, genre, etc.

Because of the different size of each playlist, and in order to reduce eventual gradual transitions that can occur in the whole playlist duration, we split the lists in smaller groups of music works. This set of sub-groups G are selected through a window of size $T = 5$, sliding over all the possible centres between $[\frac{T}{2}; l - \frac{T}{2}]$, where l is the playlist length. In this way, we are performing a data augmentation, which provide more samples for the analysis.

For each dimension, we compute the variance *within* (σ^2_W) and *between* (σ^2_B) the groups in G, following the definition from the ANalysis Of VAriance (ANOVA)[6]:

$$\sigma^2_W = \sum_{g=1}^{G} \sigma^2_g \frac{n_g - 1}{n - 1} \qquad \sigma^2_B = \sum_{g=1}^{G} (m_g - m)^2 \frac{n_g}{n - 1} \tag{9.2}$$

where n_g is the population (number of works) for the group g, n the total population of the dataset, m_g the mean in the group g, m the mean among all groups in G. The comparison between the two variances allows to have a neutral index, not prone to discrepancies in absolute values which can be introduced by the different embeddings generation contexts (explicitly declared in Section 8.1).

A σ^2_W smaller then σ^2_B means that the groups maximise the internal variance on the overall one or, in other words, the groups have particular homogeneity over specific dimensions

[6] https://people.richland.edu/james/lecture/m170/ch13-1wy.html

respect to the overall dataset values. Figure 9.1 and 9.2 plot on the same axis the two variances, respectively computed on artist and work embeddings, showing some important gaps specific dimensions, such as the ones belonging to genre and casting. A quantitative measure of the maximisation of how much the variance *within* outclasses the one *between*, is given by the **variance ratio**:

$$\sigma^2_{ratio} = \frac{\sigma^2_B}{\sigma^2_W} \tag{9.3}$$

When $\sigma^2_{ratio} < 1$, the studied dimension includes values that variate more inside groups that over all the dataset. We can reasonably exclude or limit in weight those features in our similarity function. Instead, $\sigma^2_{ratio} > 1$ reveals a strong homogeneity along that dimension, which make the groups very distinct from each other and can play an important role in the playlist generation.

Table 9.3 shows interesting differences between the datasets. Concerts are generally more homogeneous in all the dimensions. This is particularly evident and predictable for the dimensions about casting and solo, whose low *variance within* can be explained by the fact that they are normally works played by the same group of performers. In general, higher values belongs to dimensions of composer and genre, to which we can also add the solo instrument – which has to be taken with a grain of salt because this is an element not always present in a composition. The values of key are mostly < 1 and do not encourage the assignment of strong weights. Surprisingly, the composition date is not always discriminant: while it plays a strong role in concerts – probably due to the aim of give a theme to the event – and a relevant one in Spotify playlists, the data reveal a low interest in web-radio programming for this metadata. Apart from this latter point, playlists and web-radio have very similar results, which peak variance ratio in few dimension of casting, solo and genre.

A similar experiment applied on artist embeddings (Table 9.2) reveals the historical period and the genre are the most distinctive traits, followed by the places of birth and death of the artist.

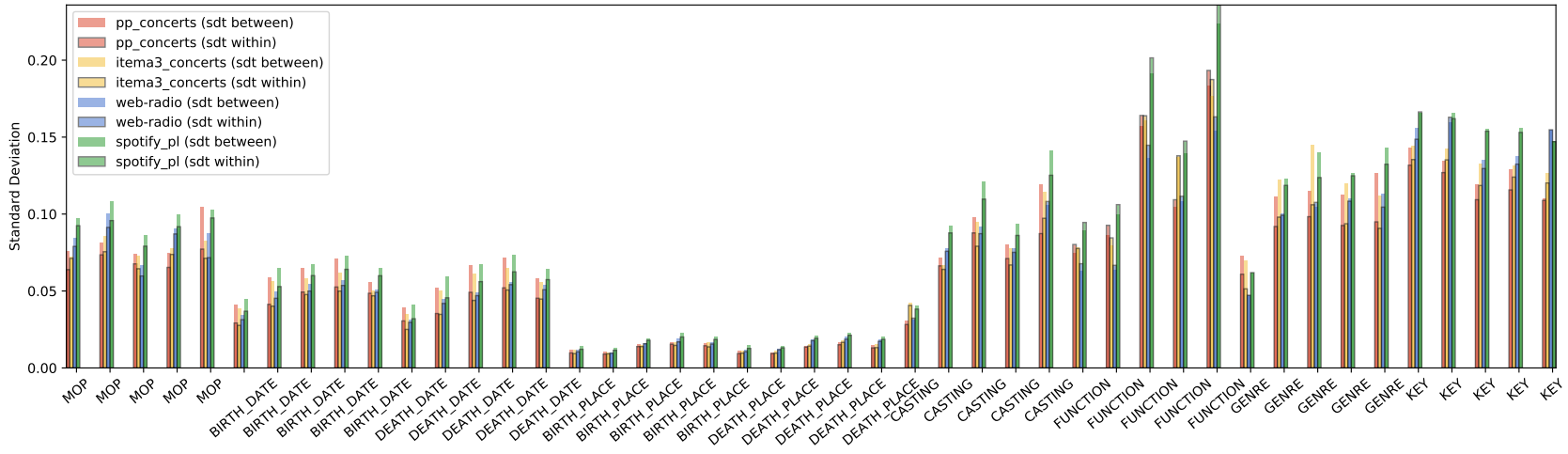


Figure 9.1 – Square root of variance *within* and *between* the playlists of each dimension for each dataset, computed on **artist embeddings**.

	mop					birth date					death date					birth place					death place				
pp	1.98	1.49	1.46	1.70	**3.42**	**4.00**	**4.16**	**2.99**	**3.30**	1.79	**2.66**	**4.72**	**3.38**	**3.65**	**2.79**	**2.06**	1.80	1.60	1.33	1.51	**2.18**	1.41	1.15	1.55	1.66
itema3	1.04	1.68	1.66	1.23	1.84	**3.88**	**3.86**	**2.26**	**2.35**	1.33	**3.85**	**4.28**	**3.77**	**2.76**	**2.42**	**2.62**	1.40	1.38	1.64	**2.09**	1.90	**2.02**	1.38	1.42	1.79
web-radio	1.31	1.47	1.59	1.16	**2.21**	1.47	1.48	1.41	1.26	1.14	1.24	1.32	1.17	1.13	1.28	1.60	1.35	1.10	1.60	1.37	1.62	1.30	1.23	1.43	1.29
spotify	1.24	1.64	1.42	1.38	1.24	**2.23**	**2.27**	1.58	1.69	1.36	2.71	**2.87**	**2.06**	1.92	1.56	**2.07**	1.64	1.38	1.65	1.52	1.96	1.45	1.43	1.38	1.42

	casting					function					genre					key				
pp	1.46	1.34	1.57	1.62	3.47	0.74	0.76	0.83	0.83	0.80	**2.06**	**2.15**	1.88	**2.19**	**3.16**	1.40	1.26	1.43	1.54	1.05
itema3	1.19	1.19	**2.05**	1.84	1.92	1.04	0.78	0.93	0.96	0.78	**3.39**	**2.45**	**3.52**	**2.66**	**2.33**	1.29	1.24	1.56	1.25	1.24
web-radio	0.88	1.09	1.21	1.14	0.92	0.76	0.83	0.78	0.88	0.80	1.00	1.04	0.88	1.06	1.37	1.20	0.92	1.17	1.17	0.99
spotify	1.26	1.24	1.50	1.39	1.63	0.80	0.77	0.80	0.80	0.81	1.01	1.16	1.65	1.05	1.36	0.98	1.09	1.04	1.08	1.00

Table 9.2 – Ratio of variance *between* / *within* for each dataset, computed on **artist embeddings**. In bold, the values greater than 2.

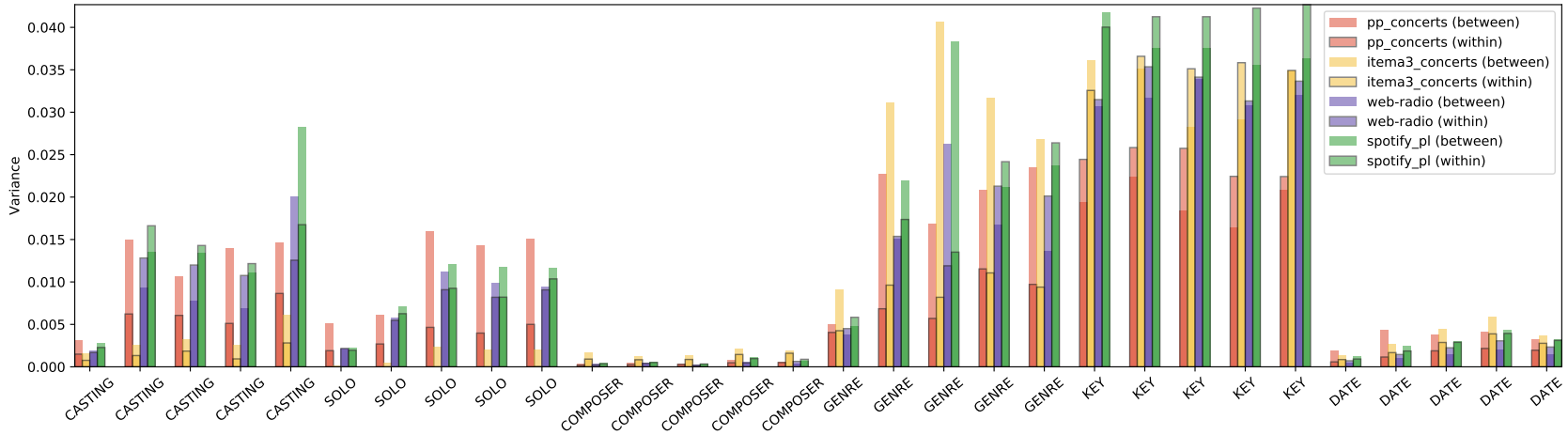


Figure 9.2 – Square root of variance *within* and *between* the playlists of each dimension for each dataset, computed on **work embeddings**.

	casting					**solo**					**composer**					**genre**					**key**					**date**				
pp	**4.4**	**5.8**	**3.0**	**7.5**	**2.8**	**7.3**	**5.1**	**11.8**	**13.0**	**9.1**	**4.3**	**3.0**	**1.4**	**2.7**	**1.5**	**1.5**	**11.0**	**8.7**	**3.2**	**5.8**	0.6	0.7	0.5	0.5	0.8	**11.2**	**14.2**	**4.1**	**3.6**	**2.8**
itema3	**4.2**	**3.7**	**3.0**	**7.4**	**4.7**	**18.3**	**15.8**	**38.9**	**0.43**	**11.3**	**3.2**	**2.2**	**2.5**	**2.2**	**1.4**	**4.6**	**10.4**	**24.7**	**8.2**	**8.1**	**1.2**	0.9	0.6	0.6	0.9	**2.4**	**2.5**	**2.4**	**2.2**	**1.8**
web-radio	**1.2**	0.5	0.4	0.4	**2.5**	0.9	**1.1**	**1.5**	**1.4**	1.0	**2.3**	**1.3**	**2.4**	**1.8**	0.3	0.7	0.9	**4.8**	0.6	0.4	0.9	0.8	0.9	0.9	0.9	0.4	0.5	0.4	0.4	0.4
spotify	**1.4**	0.6	0.8	0.8	2.8	**1.3**	**1.3**	**1.7**	**2.0**	**1.2**	**1.8**	**1.2**	**1.3**	**1.3**	0.6	0.6	**1.6**	**8.0**	0.7	0.8	1.0	0.8	0.8	0.7	0.7	**1.7**	**1.7**	**1.1**	**1.2**	**1.1**

Table 9.3 – Ratio of variance *between* / *within* for each dataset, computed on **work embeddings**. In bold, the values greater than 1.

9.2 Weights for playlist ranking

The Euclidean similarity described in Section 8.3 can be applied to rank the items in a simple content-based playlist generation system in which, given a seed work s, the more similar elements have higher chances to appear in the top of the list. Candidates are chosen among a pool of target works T, by ranking them with the similarity function (8.2) and selecting the top k results. The similarity is computed on the weighted version of the distance function (8.4).

The weight vector w is derived from the variant ratio values. A cutting threshold θ is chosen, so that values lower than θ are assigned to a fallback value β. The weight vector is then equal to:

$$w(\theta,\beta) = \begin{cases} \sigma^2_{ratio}, & \text{if } \sigma^2_{ratio} \geq \theta \\ \beta, & \text{otherwise} \end{cases} \tag{9.4}$$

Moreover, σ^2_{ratio} can be computed on single datasets or combining the results of different ones.

In order to find the best values for θ and β, we performed a simple experiment. For each dataset D with P playlists, we set as pool T the list of distinct works in all the playlists of the dataset. For each playlist $p \in D$, we select sequentially each item $x \in p$ and we run the recommender system on T with x as seed. Among the first n predictions y_n – with n in $(100, 200, 500)$ –, we consider positive prediction y^+_n the ones that appears also in p ($y^+_n = y_n \cap p$). A score is computed by averaging the number of positive prediction, normalised with respect to the playlist population:

$$score_n(D) = \frac{1}{P} \sum_{p \in D} \sum_{x \in p} \frac{\# y^+_n}{\# p} \tag{9.5}$$

As the concerts sets have particular and specific characteristics, we preferred to not take them into account. We repeated the experiment with different parameters – with $\theta \in [1.0, 1.1, ...2.0]$ and $\beta \in [0.1, 0.2, ...1.0]$ – and empirically found $\theta = 1.4$ and $\beta = 0.7$ as the values that maximised the co-occurrences predicted-expected. The same approach has been applied for finding weights for the artists similarity measure.

9.3 Evaluation

The recommender system has been evaluated by two groups, composed by experts coming respectively from the world of radio broadcasting (Radio France (RF), 4 members) and concert

halls (Philharmonie de Paris (PP), 3 members). We prepared a wizard interface[7] composed of 10 steps. Each step shows a seed item (artist or work) and the first 10 target items, ordered by similarity score. Some steps used the unweighted similarity measure, in order to compare with the weighted version. In some cases, the ranking is explained declaring the features which maximise the similarity, e.g. "*similar genre and composer*".

The evaluators are asked 1. to remove the wrong items by dragging and dropping them into a trash area (with no predefined dimension) and 2. to sort the remaining good items by relevance in the order they preferred. We collected, for each step, the dimension of the trash bin and the Spearman correlation applied between the original and the edited rankings [14], taking into account uniquely the good items, defined as:

$$\rho = 1 - \frac{6\sum_i^{n-1} d_i^2}{n(n^2-1)} \tag{9.6}$$

where d_i is the difference in ranking of the i-th element and n the number of involved element. The results are reported in Table 9.4, keeping separated the contribution of the two groups.

	SEED				RF		PP	
	artist	**work**	**explain**	**weights**	**corr**	**trash**	**corr**	**trash**
1	A. Vivaldi	–		✓	0.81	1.25	0.81	1.57
2	A. Vivaldi	–			0.49	1.75	0.64	2.43
3	I. Schubert	–	✓	✓	0.85	4.50	0.51	5.14
4	I. Stravinsky	–	✓	✓	0.50	4.50	0.29	5.14
5	L. van Beethoven	Moonlight Sonata		✓	0.17	3.50	0.14	3.29
6	L. van Beethoven	Moonlight Sonata			0.08	5.50	0.34	5.57
7	R. Wagner	Der Ring des Nibelungen	✓	✓	0.57	4.00	0.41	4.29
8	M. Davis	Spanish Keys	✓	✓	0.71	5.75	0.48	6.67
9	A. Reicha	Concert for clarinet in Gm	✓	✓	0.72	2.25	0.64	3.86
10	J. S. Bach	Sonata for viol and hapsicord	✓		-0.02	2.5	0.17	3.17

Table 9.4 – Evaluation scores for the ranking. The seed work is absent when evaluating the artist ranking. For each step, we marked the use of explanation and weights, the average Spearman correlation and the average dimension of the trash bin, separately for each group.

The results show an overall preference for the weighted version of the rankings, while revealing some differences between the two groups. The effect of the training – computed on radio data and reducing the importance of some crucial features for the concert programming – is visible in the better scores given by the RF group. PP evaluations put more items in the trash and have a shorter gap between weighted and unweighted ranking. A jazz piece (*Spanish Keys*) has been included as outsider in order to reveal the limit of the approach: the weights privilege the composition date and appear to not suit the high range of genres of 20^{th} century, given the

[7] http://overture.doremus.org/evaluation

number of wrong prediction. Regarding the *Moonlight Sonata* work, the low correlation scores suggest important changes in the list order made by experts, which have definitely different opinions about this known masterpiece. No obvious benefit can be observed regarding the presence or absence of the explanation in the user interface but this may also be due to the fact that information was not made sufficiently visible to get attention. An experiment involving the same evaluation board but with weights computed on the concert dataset is foreseen as future work.

9.4 Conclusions and Future Work

We have presented a study on editorial classical music playlists in order to understand the features that rule the playlist realisation. This study enable us to gave then weights to the different dimension of the embeddings, in order to promote some specific features in the similarity measure. The weight vector is the result of a study of the homogeneity of each dimension in different collections of works (concerts, radio programming, playlists), and appears to have a positive impact on the ranking of predicted playlists.

Among the possible improvements for the approach, we believe that a preliminary filtering of the candidate pool based on a single feature (composer, period or genre) could benefit both the results and the speed of the system, having to work on a relevant subset of items. Having realised that differences in features variations may change in the time, new strategies can focus on producing specific weights for each historical period, that dynamically applies in reference to the given seed. Further experiments can be conducted for studying the impact of this approach in conjunction with common recommendation techniques, i.e. for tuning the computed results or for cold-starting a collaborative recommender system.

Finally, as future research, we plan to study how to include the concept of *novelty* [86] applying constraints on the minimal distance or relying more on the not homogeneous features. Further experiment will be conducted for judging if these methods are extensible to other kind of music or other domains.

9.5 The role of playlist title: Title2Rec

This and next section, located as appendices to Chapter 9, include two other works which have been carried during my PhD work. Even if not directly related to classical music, they are interesting for completing our understanding of playlist creation with two crucial elements: titles and emotions.

The title of a playlist can potentially contain interesting information about the intention and the purpose of its creator. The title can suggest that the tracks in certain playlist are intended to

suit a certain goal (e.g. *party*, *workout*), a mood (*sad songs*, *relaxing*), a genre (*country*, *reggae*), or a topic (*90's*, *Christmas*). Our intuition is that playlists with similar titles may contain similar tracks.

I have been part of the D2KLab submission for the **RecSys Challenge 2018**[8] [33], organised by Spotify and focusing on playlist completion. Following the challenge rules, the target dataset is the Million Playlist Dataset (MPD), which contains metadata for 1 million playlists gathering more than 2.2 million distinct tracks, mostly belonging to popular music. The outcome is an ensemble strategy which involves different types of features, including sequential embeddings, title embeddings and lyrics features. Our work globally ranked 37/112 for the main track and 13/31 for the creative track, and was invited to be published in the proceedings and presented to the conference thanks to its original approach [130]. My contribution stands mainly in the realisation of Title2Rec, which allowed to predict tracks of a new playlist of which we know the title but not any element.

9.5.1 Algorithm

The title similarity could rely on pre-trained models and thesauri. However, we opted for computing a model that is specific for the playlist continuation task, using the sole data of the MPD.

We use *word2vec*[9] [128] for generating the embeddings representing tracks, giving in input to the algorithm the sequences of tracks as they appear in the playlists. The generated embeddings, which in other words encode the presence of the same track in the different playlists, are used for realising the playlist embedding p_{w2v}, computed as the mean of the embeddings of the tracks composing the playlist. The playlist embeddings are grouped in n clusters, applying the K-means algorithm. We empirically observed that, apart from very general clusters, we also created clusters containing specialised playlists, obtaining as a consequence groups of titles that belong to the same semantic area. For example, we may have a cluster containing playlists with *Christmas feels*, *December* or the emoji of Santa Claus (🎅), while another group encompasses playlists like *country* and *Alabama*.

Each cluster c expresses a composed label, which is the concatenation of the titles of all the playlist $p \in c$ separated by a blank space. These labels can be seen as a corpus of n documents (one for each cluster) that is used as input for the *fastText* algorithm [85]. Figure 9.3 illustrates the process of the Title2Rec model generation.

The model is exploited for recommending tracks starting from the title of a new created playlist,

[8] https://recsys-challenge.spotify.com/

[9] Genism implementation [164], embedding dimension 100, learning rate 0.025 linearly decaying up to 0.0001, window size 5, number of epochs 5

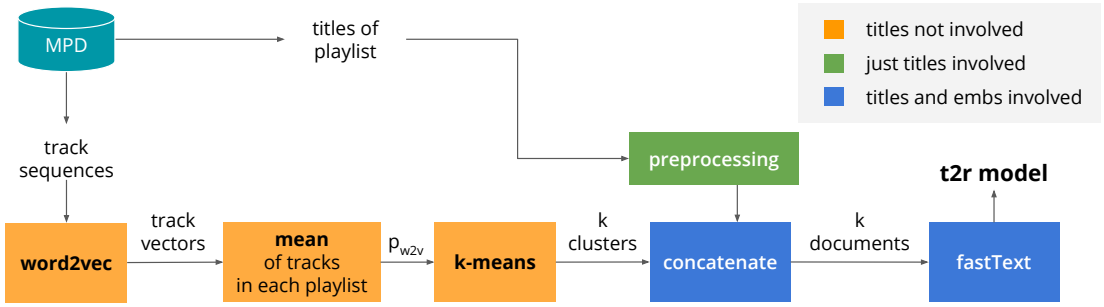


Figure 9.3 – Pipeline for generating the title embedding model used in Title2Rec.

which acts as seed for the recommendation, as illustrated in Figure 9.4. Because *fasText* is able to represent textual information at the level of n-grams from 3 to 6 character, the model can compute the embeddings p_{t2r} of any title, being this already seen in the dataset or totally unknown. The embedding of the seed playlist is compared using the cosine similarity with all the known playlist title embeddings. The subset P of the top-300 most similar playlists is extracted. Finally, the required number of tracks are selected among the ones available in P.

The tracks have been ordered to ensure that:

- the most popular ones in P are placed at the top of the list;
- the impact of each playlist is proportional to the similarity score of the title embeddings comparison

In other words, a track has a higher chance to be recommended if it is included in a large number of playlists in P and if most of them are among the top ones more similar to the seed.

9.5.2 Optimisation

In order to improve the performances of Title2Rec, we worked on different parts of the pipeline. Each optimisation has been tested by running the algorithm on a validation set of 1000 playlists. Then, only the edits that improved the scores with respect to the non-optimised version have been kept in the final version.

On each single title, we applied a pre-processing phase that foresees a series of tasks:

- lowercasing;
- detecting and separating emojis from words;
- transforming space-separated single letters into words (e.g. "*w o r k o u t*" becomes "*workout*");
- detecting and separating emoticons[10] from words;

[10] An emoticon is an image made up of symbols such as punctuation marks, e.g. :-). An emoji is a small

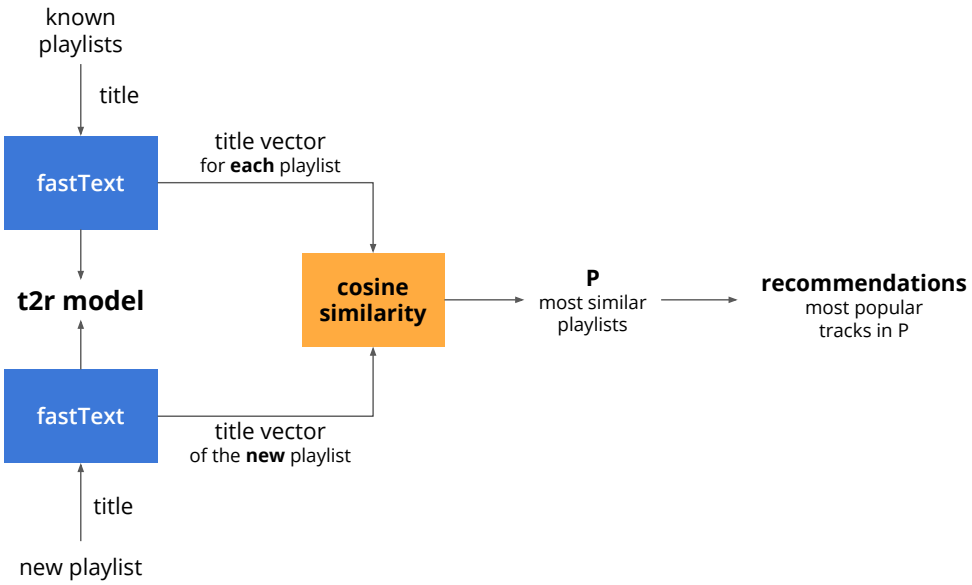


Figure 9.4 – Title2Rec: Recommendation algorithm.

- separating the skin code from the emoji;
- remove '#' from hashtags.

Other tasks that have been tested with no improvements are:

- detecting and separating punctuation from words;
- removing stop words;
- removing all spaces.

The latter point has been partially exploited because we noticed an improvement in the results by including in the corpus both versions of the title – keeping spaces (as in "green day") and removing them ("greenday").

Another optimisation step included the usage of different parameters for executing the pipeline. The clustering phase has been tested with different values of k (the number of clusters in output for the K-means algorithm). The value of 500 gives better results than smaller and bigger ones, which produce clusters that are respectively less specialised and less populated. The *fastText* training has been run with 5 epochs, a learning rate of 0.1 and different loss functions (ns, hs, **softmax**), window sizes (3, **5**, 10). The values in bold represent the best results.

Finally, some improvements come from the inclusion of the playlist descriptions in the training. On the whole set of descriptions in the MPD, we compute a Term frequency-inverse document

pictograph, commonly encoded as special character and visualised as image, e.g. 😊.

Approach	R-Precision	NDCG	Click
Most Popular	0.0373	0.0959	18.529
Title2Rec	0.0837	0.1260	12.007
Ensemble (main track)	0.1611	0.1710	3.6349
Ensemble (creative track)	0.1634	0.1717	3.5964

Table 9.5 – Results of Title2Rec compared with a baseline and with the full ensemble.

frequency (TF-IDF) model. Thanks to this, we are able to extract a set of keywords for each description by selecting the 3 words with the highest score. These keywords are added to the documents used to build the clusters. The contribution of the description is null when the playlist does not include any.

9.5.3 Results and Future Work

Title2Rec has been used in combination with 3 different Recurrent Neural Networks (RNN) in an ensemble algorithm. The strategy rely on the selection of the most represented recommendations among the results of the 3 RNNs, which receive in input sequential embeddings (for tracks, albums, and artists), Title2Rec embeddings, and a vector of features extracted from lyrics[11]. In case of title-only seed, the results are computed with Title2Rec.

Table 9.5 shows the results obtained by Title2Rec and the ensemble, also compared with a baseline realised by selecting the most popular 500 tracks. The playlist title has been revealed to be highly informative, being capable to reach on its own half of the score of the full ensemble.

Further experiments may improve the performance of the algorithm. The scores of order-dependent metrics can benefit of different sorting strategies, like the Borda count[12]. The use of pre-trained *fastText* model – alone or in combination to the ones computed on playlists' clusters – should be tested. Finally, we foresee a more systematic evaluation of the contribution of the single pre-processing tasks, in order to select the best ones.

9.6 The role of playlist emotions

A party, a Sunday afternoon at home, the commuting time before an exam. At different moments of their day, users search different music, which would arise in them a specific emotion [124]. This search impacts even in playlist realisation.

Following the intuition that emotions are mostly homogeneous inside playlists and state-of-

[11] Used uniquely in the creative track, in order to respect the Challenge rules.
[12] https://en.wikipedia.org/wiki/Borda_count

the-art techniques [91] for emotion detection can be applied to song lyrics, we studied an approach for classifying the main emotion of a given playlist within four different classes: *relaxed*, *happy*, *sad* and *angry*. The prediction relies on aggregating the emotion prediction computed at the song level through the analysis of their associated lyrics (we consider English lyrics). The experiment has been carried by two students of EURECOM under my supervision and has been published in [66].

9.6.1 Emotion Recognition in Song Lyrics

Previous studies identified some features, which are interesting for performing different kind of classification of songs [60]. Among those features, we selected a smaller subset, containing the features that maximise the performance of the later-described approach:

- **%Past_tense_verbs**: percentage of past-tense verbs over the total number of verbs;
- **%Present_tense_verbs**: percentage of present-tense verbs over the total number of verbs;
- **%Future_tense_verbs**: percentage of future-tense verbs ("will" or "'ll" + base form) over the total number of verbs;
- **%ADJ**: percentage of adjectives over the total of words;
- **%PUNCT**: percentage of punctuation over the total number of words;
- **%Echoism**: percentage of echoism over the total number of words, where an echoism is either a sequence of two subsequent repeated words or the repetition of a vowel in a word;
- **%Duplicate_lines**: number of duplicated lines of the total number of lines in the lyrics;
- **isTitleInLyrics**: *true* if the lyrics contain the title string;
- **Sentiment_polarity**: sentiment polarity, between -1 (negative) and 1 (positive);
- **Subjectivity_degree**: degree of subjectivity of the text between 0 and 1.

All the features have been normalised by subtracting the mean and scaling to unit variance. Each song is represented by a feature vector, obtained by the concatenation of a 300-dimension Glove word embedding of the lyrics[13] [152] and the 10 features described above.

The ground truth for the experiment is the perfectly-balanced *MoodyLyrics4Q* dataset [29], which contains 2000 manually annotated songs with the four different emotion labels – *relaxed*, *happy*, *sad* and *angry* –, which we interlinked with LyricsWikia[14] in order to obtain the lyrics. The feature vector is given in input to four different classifiers, listed in Table 9.6 together with their accuracy computed with a 10-fold cross validation. Having obtained the

[13] A single word embedding representing a song is realised by averaging the word embeddings of all tokens in the song

[14] http://lyrics.wikia.com/

best performances, the Neural Network[15] is chosen to be employed in the following of the experiment.

A confusion between the *"sad"* and *"relaxed"* classes is a common pattern in all classifiers (see Figure 9.5). To investigate the reasons, we downloaded and read the lyrics of some songs and we discovered that discriminating between *"sad"* and *"relaxed"* emotions is hard, also for humans.

Classifier	Accuracy
Neural Network	58.45%
Logistic Regression	57.87%
Support Vector Machine	58.04%
Xgboost [34]	56.89%

Table 9.6 – Accuracy results on MoodyLyrics4Q

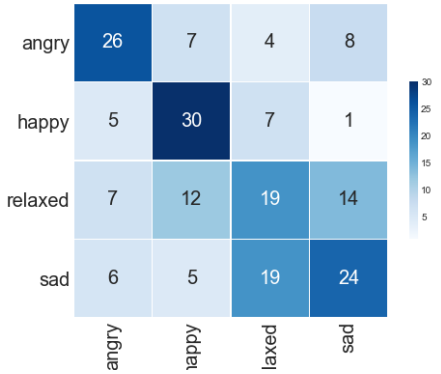


Figure 9.5 – Neural network confusion matrix

9.6.2 Playlist Classification

The network output consists in a probabilistic distribution of the emotion of the input song, in the form of vector `[sad%, angry%, happy%, relaxed%]`, where the sum of the percentages is equal to 100. The dominant emotion within a playlist is the one with the highest probability. The score s_i^x for the emotion i in the playlist x is equal to the sum of all the individual song probabilities normalised by the number of songs l, according to the formula:

$$s_i^x = \sum_{song \in x} \frac{s_i^{song}}{l} \tag{9.7}$$

[15] Feed-forward network with 2 hidden layer with sigmoid activation, and an output layer with softmax activation. Other parameters are batch size = 256, epochs = 100, optimizer = Adam, loss = categorical cross-entropy.

In order to face the absence of a dataset of annotated playlists, a perfectly balanced dataset has been generated by manually picking 40 playlists from Spotify, chosen among the ones representing an emotion in the playlist title, e.g. *"Sad songs"* or *"angry music"*. Within this environment, our algorithm reaches **80%** of accuracy. For better appreciating the predictor, we developed a web app available at http://data.doremus.org/emotion. The app requires the use of Spotify account in order to get a token for the Spotify Developer API. The application receives in input from the user the Spotify URI of a playlist. The lyrics are downloaded and classified, and then, the results are aggregated for predicting the dominant emotion.

Looking at scores, we discovered that normally playlists result quite homogeneous in representing a single emotion. Future work would investigate how the dominant emotion can be used in improving recommender system performances.

Chapter 10

Explore

In addition to the research implications described so far in Part II, the adoption of a KG can benefit directly the final user, providing him/her a more complete information and easier ways to access it. User access to information is nowadays declined in several different media, including the web, mobile apps and vocal assistant.

This Chapter presents three ways to access the DOREMUS KG. In Section 10.1 we describe an exploratory search engine for DOREMUS data, built using web technologies and SPARQL Transformer. An experiment of location-aware recommender system is presented in Section 10.2, while the DOREMUS Chatbot is introduced in Section 10.3. Finally, Section 10.4 contains some conclusions.

10.1 Explore the Music Graph with OVERTURE

Knowledge discovery is often entrusted to exploratory search engines. Instead of obtaining a precise results, the goal of exploratory search is learning something about a more or less vague topic, with a serendipitous attitude that push into continuing the search [146].

We developed a prototype of an exploratory search engine for DOREMUS data, under the name of OVERTURE (Ontology-driVen Exploration and Recommendation of mUsical REcords). The application relies on a Node.JS server sitting in front of a Virtuoso triple store (Figure 10.1). The queries are performed through SPARQL Transformer (see Chapter 6) which maps the results into a Schema.org format (following what described in Section 3.2). In this way, a simplified API for DOREMUS data is exposed and interpreted by the client part of OVERTURE, realised with the Angular framework[1]. The application is available at http://overture.doremus.org.

The UI has been designed with the goal of allowing the user to navigate the DOREMUS graph

[1] https://angular.io/

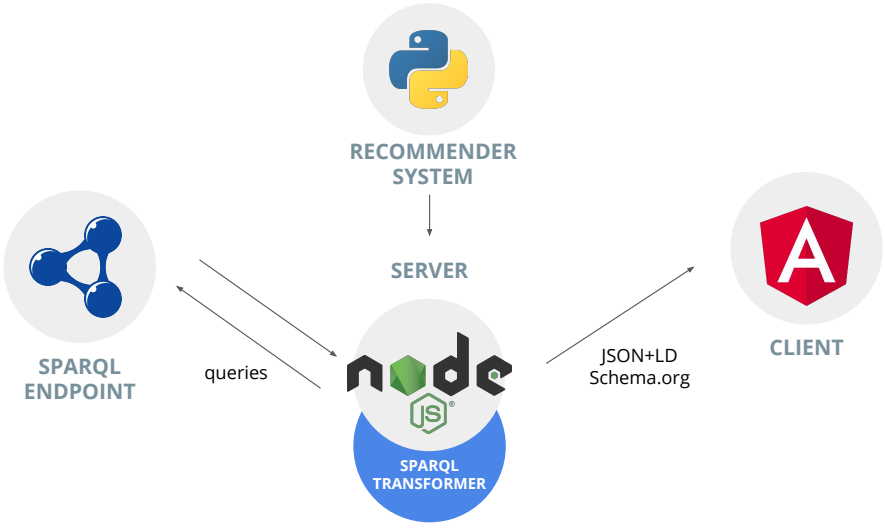


Figure 10.1 – Application schema of OVERTURE

according to its own structures. At the top, the menu bar presents the main concepts of the DOREMUS model: works (including expressions), performances (including recordings), scores, artist. In each of this sections, it is possible to perform a detailed advanced search (Figure 10.2). Works are searchable by facets, that include the title and the composer, but also keys, genres, detailed castings, making it possible to select very precise subsets of data, like all sonatas (genre) that involve a clarinet and a piano (MoP). Under the hood, the selection of facets modifies the SPARQL Transformer query by including some `VALUES` constraints. The hierarchical properties in controlled vocabularies (Chapter 4) allow the smart retrieval not only of the entity that match exactly the chosen value (i.e. *strings*), but also any of its narrower concepts (i.e. *violin*, *cello*, etc.), taking in account also the interlinks between vocabularies.

OVERTURE is available in English and French, exploiting the language selection feature of SPARQL Transformer. In this way, label selection follows a priority order, which depends on the chosen application language, with English and any other language as foreseen fallback.

Figure 10.3 shows Beethoven's *Sonata for piano and cello n.1* as seen in OVERTURE, as example of detail page. Aside from the different versions of the title, the composer and a textual description, the page provides details on the information we have about the work, like the musical key, the genre, the intended MoPs, the opus number. When these values come from a controlled vocabulary, a link is present in order to search for expressions that share the same value, for example, the same genre or the same musical key, providing the user with a graph browsing experience. A timeline shows the most important events in the story of the work – i.e. the composition, the premiere, the first publication. Other performances and publications can be represented below and it is possible to click on them for accessing to their detail page. On the right side, similar items computed according to the strategies defined in Chapter 9 are

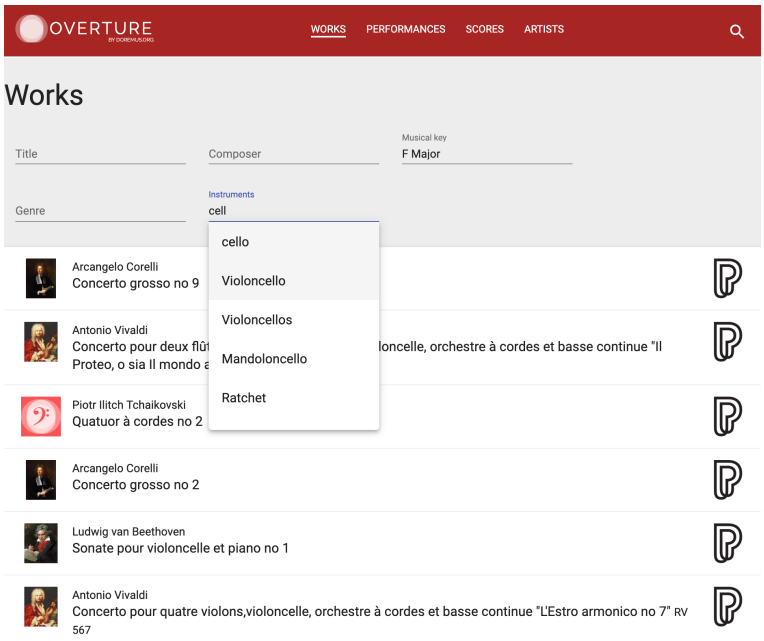


Figure 10.2 – OVERTURE: advanced search

shown.

OVERTURE has largely been used for manual testing the content of DOREMUS graph and the result of data conversion since its first development. This has been largely helped by the URI policy, which perfectly map the one of entities, so that `http://overture.doremus.org/expression/xxxx` is the OVERTURE page about `http://data.doremus.org/expression/xxxx`, making easy to jump from the application to the data.

10.2 Discover music in the city: CityMUS

The trending consumption of music content *on the move* fosters the attention of context-based recommender systems. The main challenge of those systems relies in finding strategies for a successful connection between distinct domains, such as the context and the music.

With the aim of combining the experience of exploring a city with the one of music, we carried out an experiment of a location-aware recommendation of music using the DOREMUS KG. Our strategy consists in exploiting arbitrary semantic connection or graph paths between PoIs and musicians using DBpedia as intermediary knowledge base. Even if inspired by previous experiences [25, 87, 153], our approach does not rely on domain experts' – who are in charge to select the best subset of classes and connection to be used – but is a completely unsupervised approach. The experiment has been realised in collaboration with two master students of EURECOM and published in [105]. We chose the city of Nice as an example for performing

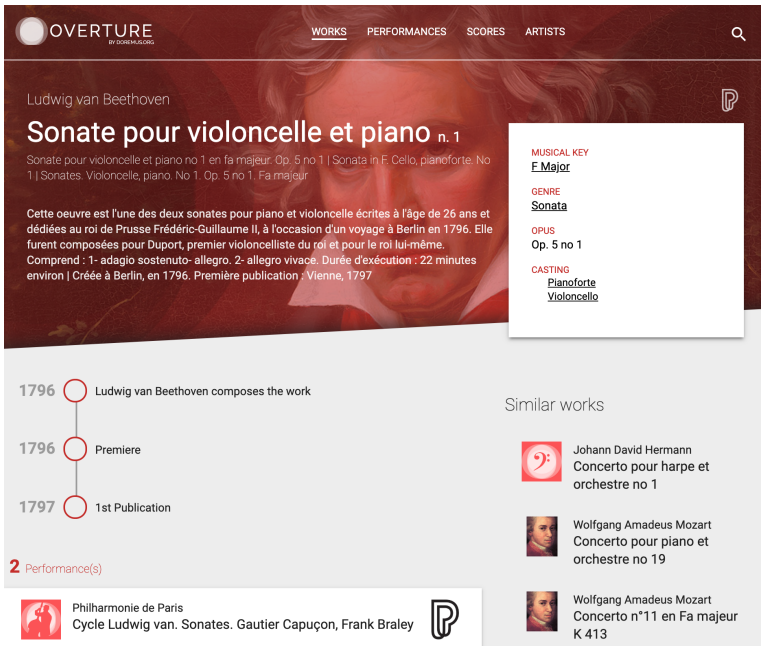


Figure 10.3 – OVERTURE: work detail

our experiment.

We rely on two specialised datasets: the DOREMUS KG – used for the data about artists, already interlinked with DBpedia – and the **3cixty** knowledge base [192]. The latter contains data about events and places from a touristic point of view for a defined list of city, among which Nice, which has been chosen for our demo. The PoIs in Nice have been retrieved through the 3cixty API[2] and matched to a DBpedia, taking in account only resources geographically located in Nice or that have `dbr:Nice` as `dc:subject`. The match relies on the aggregation of text similarity measures[3] applied on labels. The interlinking is validated using Google Maps API, taking as valid the links between places in 250 meters distance (70% of the total).

10.2.1 Path Finding and Scoring

Retrieving paths between two entities has already been solved by tools such as Relfinder [80] that searches the graph for possible paths with a given depth. However, increasing the depth generates both many more possible paths and increases the computation time. In our case, the required depth d is quite high: we need in average 5-6 edges to connect a PoI to an artist, covering the whole DBpedia depth and dimension (8.8 billion triples).

For this reason, a simplified version of Relfinder has been developed, with the implementation

[2] http://aplicaciones.localidata.com/apidocs/

[3] Partial Ratio, Token Set Ratio, Token Sort Ratio, Partial Token Sort Ratio, and the weighted combination of those (WRatio), all coming from https://pypi.python.org/pypi/fuzzywuzzy

of a bidirectional Breath First Search (BFS) [173]. We search for all the paths with depth $d/2$ of 3 from both the source (PoI) and the destination (artist) entities. Then, the two sets have been intersected in order to find the common nodes. Some joins recreate the full paths. This technique reduces the complexity from $O(b^d)$ to $O(b^{\frac{d}{2}})$, with an exponential reduction of computation times. Moreover, we decided to take in account only properties that are directed from the entities to be connected to the common node and ignoring eventual others. Finally, a pruning is performed in order to remove cycles[4] and preserve only the shortest path for each couple of entities.

Among all possible paths between each PoI and artist, we are interested in the shortest and discriminant ones: the experiment is not interesting if all the entities are connected with very common resources, such as classes or `dbr:Nice`. We define the generality formula:

$$gen = \frac{1}{|N|}\sum_{i}^{N} occ(r_i)$$

where r_i is the i_{th} resource of a path of length N and $occ(r_i)$ is the number of its occurrences in all retrieved paths. Given the biggest path depth $deep_{max}$ (in our case, 7) and path length $len(artist, poi)$, we define the similarity between a PoI and an artist:

$$sim(artist, poi) = 1 - k(\frac{\log(len(artist, poi) - 1)}{\log(2 * (deep_{max} - 1))}) - (1 - k)gen$$

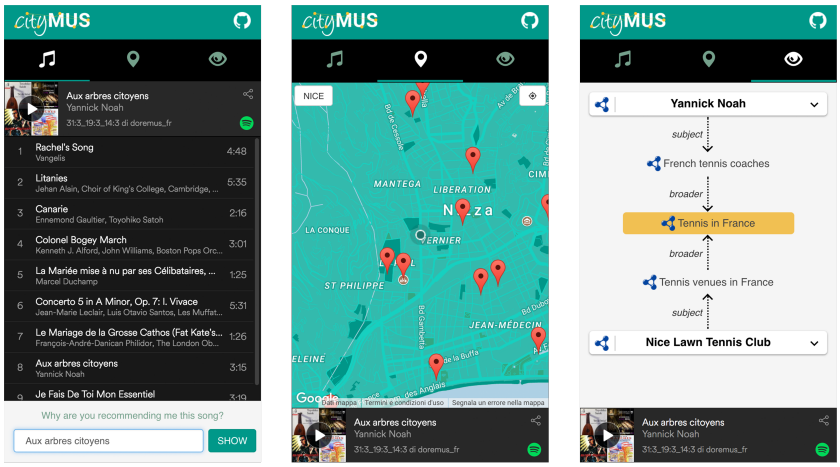
k is a variable set to 0.3. For each PoI, the artist are sorted by similarity score and the top 5 are selected.

10.2.2 CityMUS Application

CityMUS is a mobile web application available at https://citymus.doremus.org. The app uses the geo-location API for getting the user position. The server generates then a playlist of tracks from the artists connected to the closest 3 PoIs, with a different weights according to their distance. The Spotify APIs are used in order to display and play the tracks (Figure 10.4.a). The user can see the path of the song that is currently played (Figure 10.4.c) and navigate the map for discovering the songs related to other PoIs (Figure 10.4.b).

The results include some unexpected and in a way odd connections: the singer Yannick Noah was a tennis coach and is connected with the tennis lawn, while churches in Nice are often linked with some catholic musicians. This kind of connections are not necessarily bad, but for sure they raises some questions. Some explanation can be found in the structure itself of

[4] Repetitions of the same entity in the path.



a. b. c.

Figure 10.4 – The *CityMUS* app with its 3 views: **a.** Playlist, **b.** Map and **c.** Path Visualisation.

DBpedia, which connect Nice Cathedral and the Baroque composer John Dowland at distance 6 when including *Baroque*[5] and at distance 4 when including *Catholicism*[6]. Improving the results and at the same time keeping unsupervised the method would be hard, while further experiment may be performed by prioritising paths involving specify entity types.

10.3 A Music Chatbot

The year 2018 revealed a rising of voice-based AI, which massively reached our homes and brought the knowledge available on Internet within voice call. One the challenges related to this trending technology involves the design and development of smart conversational agents or chatbots [92], able to mimic the human conversation flow. Following the trend, we exposed part of the DOREMUS knowledge trough a chatbot application. The chatbot has been developed by two master students in EURECOM[7] under my supervision and it is available at https://chatbot.doremus.org.

The DOREMUS Chatbot is built on top of BotKit[8], a toolkit for easy development of chatbots (Figure 10.5). In particular, it is in charge of:

- providing the user with an access to the chatbot, thanks to its integration with Slack, Facebook Messenger, Google Assistant or web-based custom solution;

[5] 6-edges path: `dbr:John_Dowland`, `dbc:Baroque_composers`, `dbc:Baroque_music`, **`dbc:Baroque_art`**, `dbc:Baroque_architecture` (category), `dbr:Baroque_architecture` (resource), `dbr:Nice_Cathedral`

[6] 4-edges path: `dbr:John_Dowland`, `dbc:17th-century_Roman_Catholics`, `dbc:17th-century_Roman_Catholicism`, `dbc:17th-century_Roman_Catholic_church_buildings`, `dbr:Nice_Cathedral`

[7] Claudio Scalzo and Luca Lombardo.

[8] https://botkit.ai/

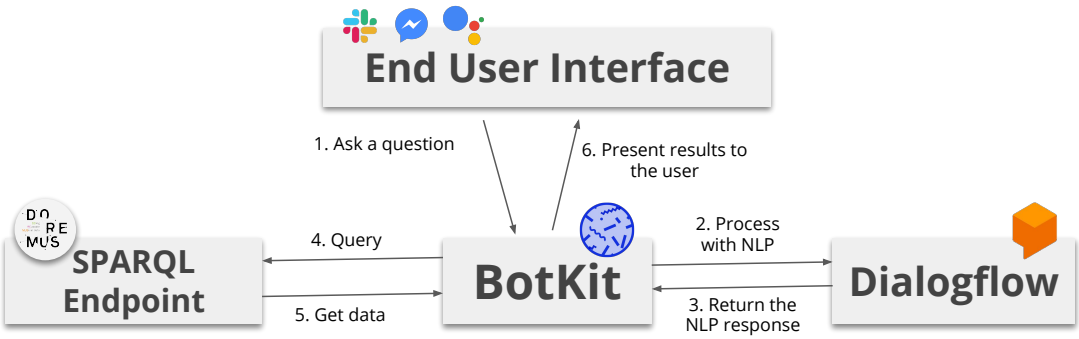


Figure 10.5 – DOREMUS Chatbot: application schema

- coordinating with a chosen NLP engine – in our case Dialogflow[9] – for interpreting the text messages;
- retrieving the interesting information from the SPARQL endpoint, in order to properly present it back to the final user.

The bot is able to successfully recognise and answer 4 categories of questions (intents) are grouped in a simple and clear way, according to what the user wants to retrieve from the DOREMUS KG:

- **`works-by`.** It retrieves a set of works according to different filter, such as artists who composed the works, instruments used, music genre and/or year of composition.
- **`find-artist`.** Finds a set of artists according to some filters, such as number of composed works, number of works of a given genre, etc.
- **`find-performance`.** It proposes to the user a future performance – filtered by city and/or date period – or shows him/her details about a past performance.
- **`discover-artist`.** It shows a card with a summary of an artist, with its birth/death place and date, a picture and a little bio. After the card visualisation, the application allows to obtain a set of works of the artist, sharing the dialog memory with the `works-by` intent.

Beyond being a way to publicly expose the DOREMUS data, the development of the chatbot allowed us to further validate the relevance of DOREMUS controlled vocabularies. The application makes strongly use of multi-language dictionaries of genre, MoP, and musicians, which are directly extracted from the DOREMUS endpoint. Their presence allowed to expose the chatbot in English and French[10] and take in account all the different synonyms. In addition, a spell-checking module has been developed for detecting and correcting misspelled elements, acting in the context of each dictionary.

[9] https://dialogflow.com/
[10] The bottleneck for including further languages stand in the absence of pre-trained NLP model.

10.4 Conclusion

Exploratory search, geo-location, conversational agent. The three applications here presented are three public online demonstrations of the data in the DOREMUS KG and of what it is possible to obtain exploiting the knowledge contained in the data. Moreover, most if the main results of this research are included in these applications, including the controlled vocabularies, the combined embedding similarity approach, and SPARQL Transformer.

Being initially developed for different purposes – demo of the data, student education –, they did not receive a proper evaluation session. However we believe that the developed approaches and ideas may be generalised and applied to different use cases and domains.

Some learned lessons found already application in other work. The development of OVERTURE – which began before the realisation of SPARQL Transformer – pushed for a more general solution for some recurrent problems during the implementation, finally embodied into the SPARQL Transformer module. Among the takeaways, it is appropriate to mention the use of domain-specific vocabularies for spelling correction, link discovery, multilingual access to the data.

Chapter 11

Learning MIDI Embeddings

Music metadata discussed so far are the product of a human cataloguing practice which lasted decades or centuries, which ensured high-quality data. Notwithstanding, human annotation is costly and not always affordable. This stimulated research in automatic extraction of metadata from the music content, being it an audio signal or a symbolic representation. In this chapter, we focus on symbolic music in Musical Instrument Digital Interface (MIDI) format [125], due to its high availability on the Web and its popularity in tasks like music generation [170] and music knowledge graphs [123].

So far, research for extracting genres [31, 119], emotions, and composers [64] with machine learning, relied on a preliminary feature selection, extracted with traditional MIR techniques. In this context, feature selection plays an crucial role, introducing the risk of over-fitting the model [72]. In a more related work towards an embedding-based symbolic music metadata classification, MIDI-glove[1] produces embeddings of notes from monophonic MIDI, but its consideration of MIDI note values leaves out e.g. timing and rhythm information, therefore producing representations of a single feature (pitch) instead of the whole MIDI content.

In this Chapter, we propose MIDI2vec, a new approach based on the embedding representation of MIDI content, in order to overcome the traditional feature selection problem. More specifically, our contributions are:

- conceptualizing relevant symbolic features (pitch, timbre, tempo, time signature) of MIDI space into a *graph space*;
- a systematically application of a well-known graph embedding generation method [70] to generate *MIDI embeddings*;
- the use of learned embeddings to predict metadata for two datasets, demonstrating that our method achieves a higher accuracy than symbolic feature-based approaches.

[1] https://github.com/brangerbriz/midi-glove

Traditionally, applications of machine learning to this problem have encountered limitations in feature selection, and more recent embedding-based techniques have been only used for other tasks (e.g. music generation [82, 170]) or on different data (e.g. music metadata [107, 109]).

To the best of our knowledge, this is the first time that embedding approaches are used for representing a whole symbolic music track.

The rest of the chapter is organised as follows. In Section 11.1, we describe our strategy to extract relevant symbolic data from MIDI files and to use it to build MIDI embeddings. In Section 11.2, we conduct an experiment to predict genre, subgenre, composer, instrument, and movement label on two different datasets using the MIDI embeddings. Finally, we conclude and outline some future work in Section 11.3.

This work has been realised in collaboration with VU University in Amsterdam.

11.1 Learning the embeddings

The MIDI format does not present a graph structure, but it consists in a time-based linear succession of events, called *MIDI messages*. Some examples are *Note On* and *Note Off* for representing played notes, *Program Change* for setting the instrument, or *MTC Quarter Frame Message* for specifying the playing speed according to the MIDI Time Code (MTC) protocol. Some of these messages are referred to a specific channel – which represents a single device emitting music – while others can apply to the whole MIDI [125].

In order to compute graph embeddings on MIDI data, we have to map them into a graph structure which preserves the informative content.

11.1.1 MIDI to graph

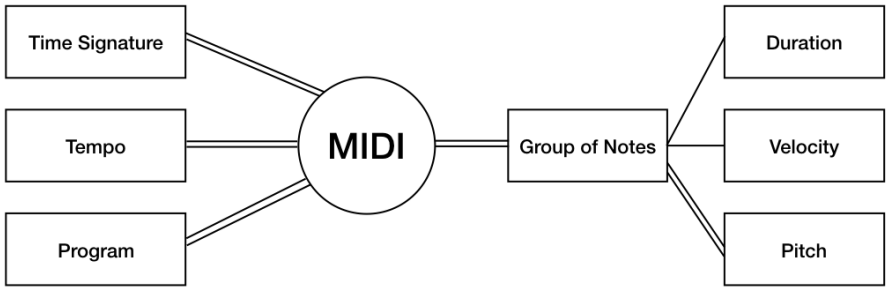


Figure 11.1 – Schema of the graph generated from MIDI. The double lines represent links of type many-to-many.

We propose a preliminary conversion of a MIDI file to a graph. As shown in Figure 11.1, a

MIDI node (the circle) will represent the MIDI file and will be connected to nodes representing different parts of the MIDI content – i.e. tempo, programs, time signature, notes. A MIDI node can be linked to one or more nodes for each type.

In the context of graph embeddings computation, literal values (text, numbers, etc.) are normally ignored [169] or some shrewdness is applied such as the use of contiguity windows [89]. In fact, literals can increase uncontrollably the number of nodes and make the graph very sparse[2], causing an exponential increase of the computation time and poor performances [157]. In our case, the crucial information represented as continuous data (e.g. the tempo) can not be excluded from the embeddings. We opted for partitioning the continuous values in ranges, in order to insert their information in the graph, while limiting at the same time the number of nodes.

In the following, some details about each type are given.

Tempo, computed in Beats per Minute (BPM). This value is computed from the MIDI tempo field (in Microseconds per Beat), according to the formula:

$$Tempo_{bpm} = 60000000 / Tempo_{midi} \tag{11.1}$$

The continuous values are then discretised in partitions, each one representing a range of 10 BPM.

Programs, representing the timbre of the channels, among the 128 different standard programs[3].

Time signature, represented as the concatenation of numerator and denominator. For example, $\frac{4}{4}$ is represented as *"44"*.

Notes, representing the pitches in the MIDI. The information about duration and co-occurrence of notes (e.g. in a chord) are not directly represented in the MIDI file. The duration is extracted by comparing successive *NoteOn* and *NoteOff* events in the same channel. Co-occurrent notes can be detected by comparing the same category of events among all channels, selecting the ones with overlapping Song Position Pointers (SPP). To include this information in the graph while limiting the number of nodes and edges, we extract all groups of notes starting (i.e. with a *NoteOn* message) at the same SPP. Each group is connected to its *duration* (the maximum duration of the notes in the group), *velocity* (their average velocity) and to all the pitches, represented with an identifier. Each group has an identifier deterministically computed from its content and it is linked to the MIDI node. In this way, two MIDI tracks sharing multiple

[2] A graph is considered *dense* or *sparse* if its number of edges is close or far to the number of all potential edges connecting each pair of vertices [53].

[3] The full list is available at https://jazz-soft.net/demo/GeneralMidi.html

chords will have more probability to appear in the same random walk, while other connections will be on single notes, duration, etc. This representation aims to track in the same time the presence of specific chords and of quick and long notes, which can respectively characterise a more virtuous or lyrical composition.

11.1.2 Graph to vectors

The output graph of the previous process is represented as an edgelist, which includes all couples of connected nodes, each one represented by an identifier. This edgelist feeds the *node2vec* algorithm, which simulates random walks on the graph and computes the transition probabilities between nodes, which would be mapped into the vector space. The parameters used in the configuration are: walk length = 10, number of walks = 40, window size = 5 and number of iterations = 5. The algorithm computes embeddings with 100 dimensions.

The whole library for producing MIDI embeddings is available at https://github.com/midi-ld/midi2vec.

11.2 Evaluation

We evaluate this strategy through two different experiments, involving two different MIDI datasets. In the first experiment, we use MIDI embeddings for predicting the music genre, while in the second, we predict a wider set of metadata.

In both cases, we rely on a Feed-Forward Neural Network made of 3 dense layers. The network receives in input the MIDI embeddings (100 dimensions) in batches of size 32. The label set used for training and testing changes according to the experiment. However, it is worth reminding the reader that those labels have not been used in the embedding task, and consequently are not directly included in the embedding information. The hidden layers count 100 neurons each and use rectified linear unit (ReLu) as activation function. The output layer uses a sigmoid as activation function and has a number of neurons equals to the dimension of the vocabulary of labels, which is represented with a one-hot encoding. These experiments are available as notebooks at https://github.com/pasqLisena/midi-embs.

11.2.1 Genre Prediction

In [118], the authors perform a genre classification task on a contextually published Symbolic Lyrical Audio Cultural (SLAC) Dataset, which contains 250 MIDI files classified according to a two levels taxonomy. The first level includes 5 genre labels (Blues, Classical, Jazz, Rap, Rock), while the second one further specialises each genre by 2 sub-genres, for a total of 10 sub-genre labels. The dataset is perfectly balanced among classes. We perform a 5-class genre

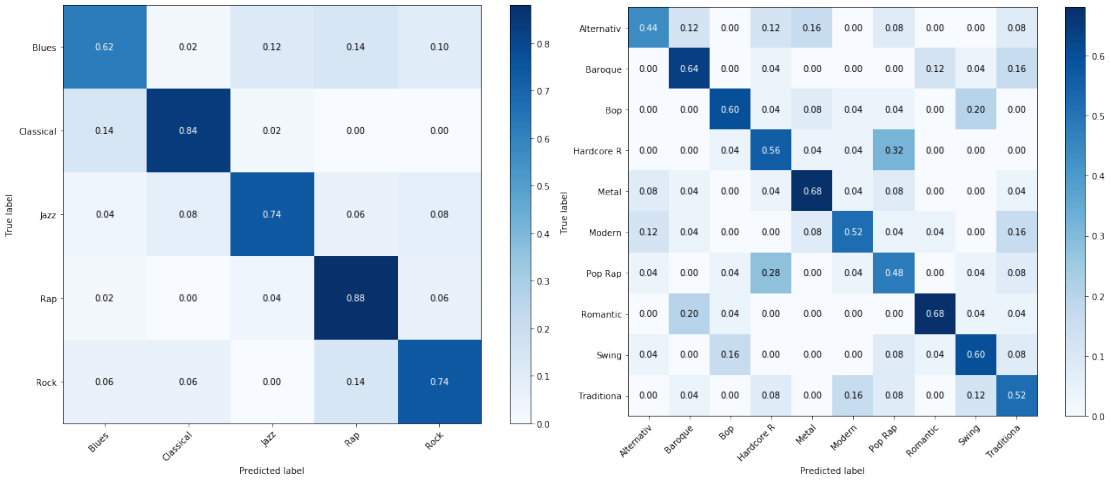
classification experiment as well as 10-class experiments on the same dataset.

Because different inputs are used for predicting the genre in [118], we select as baseline the results that rely on the sole MIDI content, even if our method outperforms also some other multi-modal results described in the paper. In particular, the method of the baseline relies on features extracted with classic MIR techniques, including spectral flux, chord frequencies and rhythmic density, using a software called jSymbolic [120]. For this reason, we use 10-fold cross-validation and provide a final score which is the average of the accuracy computed on every fold. The results are reported in Table 11.1, while Figure 11.2 shows the confusion matrix between the real and the predicted values. Even if there are not strong patterns, we can state that *Blues* is the hardest genre to identify. Figure 11.2b confirms that sub-genres belonging to the same parent genre are predictably easier to be confused.

In comparison with the baseline, our approach performs better, with an accuracy score greater than 91%. Moreover, it is worth to notice that the gap between the 5-classes prediction score and the 10-classes one is less strong, proving the effectiveness of the embeddings strategy in representing the features characterising the musical genre.

Approach	5 classes	10 classes
Baseline [118]	85%	66%
midi2vec+NN	91.99%	91.39%

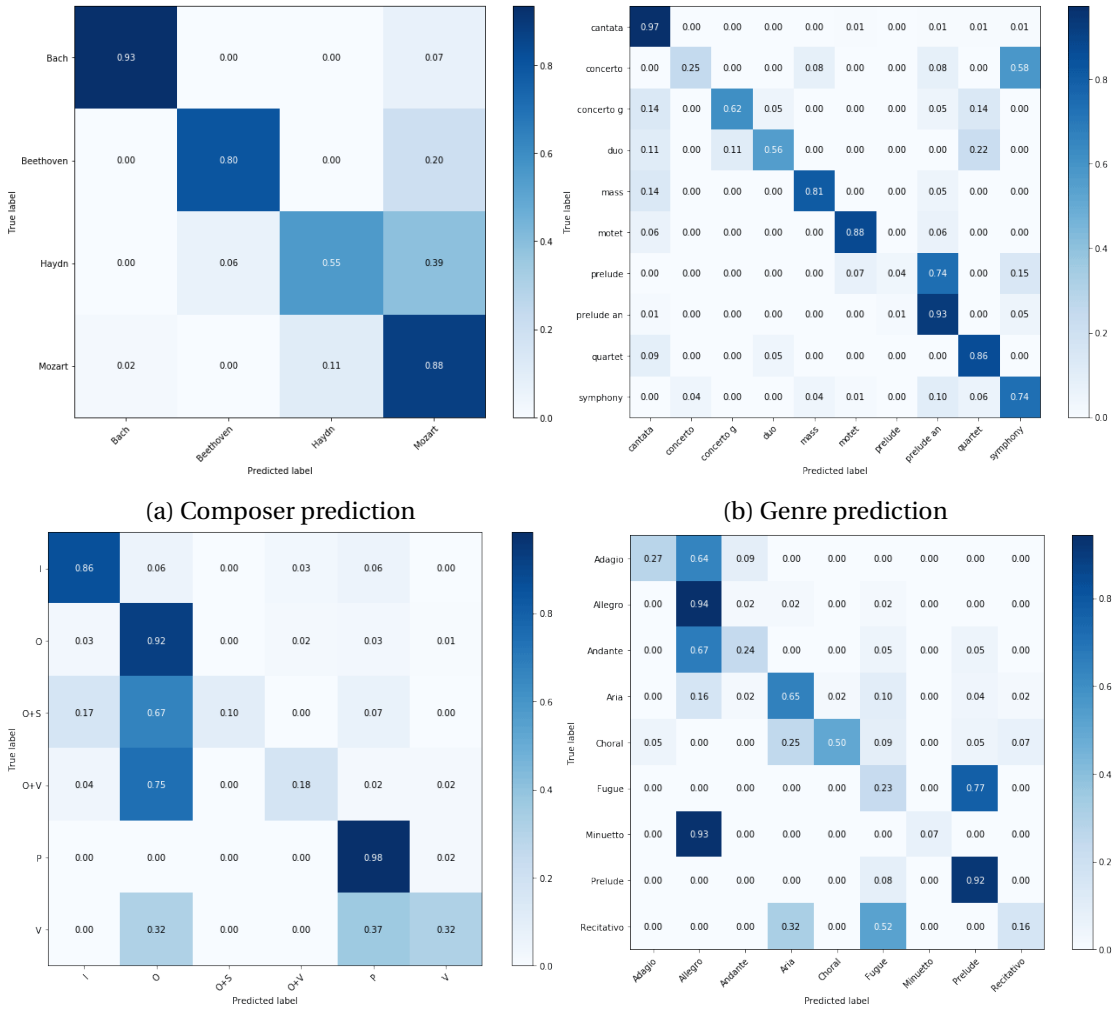
Table 11.1 – Accuracy of the genre classification.



(a) Genre prediction

(b) Sub-genre prediction

Figure 11.2 – Confusion matrices for the SLAC dataset.



(a) Composer prediction

(b) Genre prediction

(c) Instrument prediction

(d) Movement label prediction

Figure 11.3 – Confusion matrices for the Muse dataset.

11.2.2 Metadata Prediction

This task consists in predicting a set of metadata from the MIDI, namely the *composer*, the *genre*, the *instrument*, and the *movement*.

We started by downloading a corpus of 438 MIDI files from MuseData[4]. Those files refers to 139 classical music compositions, of which each file can represent a specific movement. MuseData provides also some metadata, like the composer name, the scholarly catalogue number, a label for the movement. Each composition is interlinked against the DOREMUS knowledge base.

The interlinking process consists of 3 successive steps:

[4] http://www.musedata.org

- interlinking of the composer through the exact match on the full name. This limits the candidate for the composition interlinking to the sole compositions of the interlinked composer;
- interlinking of the composition through the exact match on the catalogue number;
- if no catalogue number match is found, the titles are involved in the process. Titles can often contain other kinds of information, such as key, instruments, opus number, etc. For this reason, titles are tokenised with the method described in Section 9.1.1 and ranked according to (9.1).

Every composition can be linked to more than one MIDI file, in the case of works made of multiple movements. The movement labels have been cleaned by removing the order number, the key, the instruments and eventual comments in parentheses. For example, *"1. Allegro in E Major"* becomes simply *"Allegro"*.

The interlinking gives access to precise metadata – mostly coming from DOREMUS controlled vocabularies [107] – in particular composers (4 classes, i.e. Bach, Beethoven, Haydn, and Mozart), genres (10 classes), and instruments. For this latter dimension, given the big number of possibilities, we decided to reduce the number of classes to 6, including piano `P`, instrument (other than piano, including also small instrument ensembles) `I`, voice `V`, orchestra `O`, orchestra with voice `O+V`, and orchestra with instrumental soloist `O+S`. For instrument prediction, we excluded from the input 21 MIDI with unknown instrumentation and 3 others which did not fall into any of the previous classes, having a final source dataset of 414 items.

In addition, we consider also the movement label as feature to predict, considering only those ones which where occurring more than 10 times. Those labels include tempos (*Allegro*) and musical forms (*Prelude*), for a total of 9 distinct classes on 335 MIDI files. The dataset is not balanced among classes and has a strong presence of Bach works (76% of the total).

feature	n. items	n. classes	score
composer	438	4	93.23%
genre	438	10	93.24%
instrument	414	6	88.27%
movement	335	9	89.96%

Table 11.2 – Accuracy of the metadata classification.

Our evaluation is also based on a 10-fold cross-validation. The final accuracy (average of all the fold scores) is reported in Table 11.2. The best results belong to composer and genre prediction, even if good results can be seen for all the features. Looking at the confusion matrices:

- For the composers, the best results belong to Bach (the most present in the dataset). The two Austrian composers Mozart and Haydn are not surprisingly the two most confused

with one another, belonging both to the Classicism, differently from Beethoven (Classic-Romantic) and Bach (Baroque) [172] (Figure 11.3a);
- The genres are much more specific respect to the ones investigated in Section 11.2.1. As a consequence, the greatest confusion occurs between couples of very similar genres, such as [*concerto*, *symphony*] and [*prelude*, *prelude and fugue*] (Figure 11.3b);
- While the instrument prediction has great results in identifying works for orchestra, piano solo or small ensemble of instruments, it reveals some unreliable classification for voice-only pieces, probably due to the under-representation of the class in the dataset. In the same way, the approach is not able to distinguish compositions for orchestra only, orchestra and voice, and orchestra and soloist, all classified under the class `0` (Figure 11.3c);
- Even if the movement labels include heterogeneous meaning, the network correctly predicts 9 over 10 items. Some confusion patterns can be spotted. The *Fugue* tag is often predicted as *Prelude* (proving on the other hand a correct genre prediction) while the *Recitativo* falls under *Fugue* or *Aria*. The classes representing tempos (e.g. *Adagio* or *Tempo di Minuetto*) are often confused with the most represented class among them (*Allegro*). Some confusion is visible also between the two tags related to singing, *Aria* and *Choral* (Figure 11.3d).

Obviously, all those results should be analysed with a grain of salt, given the absence of balance between classes in the dataset. Nevertheless, they reveal the capability of the approach in dealing with specialised classification.

11.3 Conclusion and Future Work

Symbolic music content in MIDI files, and its embedding representation in vector space, are a powerful tool for automated metadata classification tasks. MIDI2vec can represent MIDI content in graph space and, subsequently, in vector space through learning graph embeddings. These embedding can successfully be used for metadata classification, outperforming previous methods using symbolic music.

We plan on extending this work in various ways. The prediction approach can be upscaled to larger, more challenging datasets, like the Lakh MIDI Dataset[5], which can additionally provide an interesting set of metadata thanks to the included matches to the MSD [158].

A MIDI ontology and a corpus of over 300 thousand MIDI in RDF format has been presented in [121]. Despite of being an interesting target for MIDI2vec, the extraction of crucial information – like the duration of a note – from the dataset is hard. In the current version, the ontology faithfully reproduces the event structure of the MIDI files, while significant edges – e.g. among

[5] https://colinraffel.com/projects/lmd/

simultaneous notes or consecutive events – are missing. We plan to extend or map the MIDI ontology in order to solve this issues and enable MIDI2vec for working on such corpus.

According to some intuition from other works in genre classification field [31], the computation should not necessarily involve the full length of the track. Experiments with different time spans or sample sizes among the graph edges can help in detecting a trade-off between the performances and the embedding computation time. Recent approaches for including literal values in graph embeddings [40, 98] could be included in MIDI2vec, in order to avoid any arbitrary choice that partitioning values implies. Finally, we will use MIDI2vec in more applied contexts, such as the task of knowledge graph completion in knowledge bases with incomplete metadata entries [123].

Chapter 12

Conclusions

Music knowledge is a valuable source for performing different tasks, from recommendation to genre prediction, among others. In this sense, the availability of high-quality structured data is a crucial factor for the success of automatic systems. This kind of data is hard to find in the context of classical music, which instead would benefit from dedicated approaches for representing and exploiting the complexity of its metadata, going beyond the track-based approach.

In this thesis we provide a first contribution to classical music research. This dissertation covers very diverse topics, which have as common focus music information and knowledge graphs. We rely on Semantic Web technologies, which gave us suitable methods to structure and publish music data, and exposing them to different state-of-the-art approaches. Some of them have been directly applied in our studies, while new ones can be investigated as future research.

In the following we summarise the content of this thesis, reporting some first implications of the obtained results. We will conclude by recapitulating the limitation of this work and suggesting some perspective for further research on this topics.

12.1 Summary of the Research

This thesis contributed in research application to the specific domain of classical music, broaching knowledge representation, data access and recommendation systems. The generation and exploitation of embeddings has been an important focus of this research, experimented on different kinds of music-related information, from metadata to symbolic music, including textual information like titles and lyrics.

The main outcomes of the thesis are:

- The **DOREMUS ontology**, a model for the description of music in detail, realised by a joint effort of librarian and technical members. The ontology is built on top of the librarian model FRBRoo, inheriting the Work-Expression-Event pattern and extending it with the addition of music-specific properties, which allow to fully describe compositions, performances, recordings, publications. The model is capable of answering complex questions, collected before the beginning of the project.

- A set of **controlled vocabularies for music metadata** in SKOS, which cover different fields like genres, MoPs, thematic catalogues, musical keys and others. The vocabularies ensures the multilingual disambiguation of music entities, thanks to a `string2vocabulary` library. Hierarchies and relations between concepts in the vocabularies enable a smarter advanced search and the possibility of generating graph embeddings, which give a mathematical representation to these concepts.

- The **DOREMUS KG**, a huge resource on classical music, published according to the LOD standards. The graph exposes fine-grained metadata, coming from the most important French cultural institution and describing artists, works, performances, scores and recordings. The realisation of this KG made use of a set of **tools for converting data**, among which a generic solution for parsing and transforming the librarian standard MARC, called **marc2rdf**.

- **SPARQL Transformer**, a solution for simplifying LD access by web developers. The library relies on a query object in JSON, which defines at the same time the desired structure in output and how to retrieve the values. An automatic reshaping of the SPARQL standard output is performed, together with a type parsing for number and booleans. A merging strategy is applied to entities described on multiple bindings, so that each graph node is represented by a unique object in the output JSON. The library has been developed both in JavaScript and Python, and has been included in the automatic API generation framework `grlc`.

- An approach for realising **entity embeddings** through the generation and recombination of partial embeddings of metadata features such as the genre, the casting and the musical key. These feature embeddings are computed using graph embedding techniques on two graphs: the graph of vocabularies – containing the semantic description of the entities (hierarchies and relationships), and the graph of usage in works and performances. Feature embeddings are averaged and combined for representing more complex entities like artists and works. A weighted Euclidean similarity metrics is proposed, including a penalty for comparing vectors with missing dimensions.

- A study of **editorial playlists**, which exploit the embeddings for estimating how much variation in genres, composers, keys, instruments, etc. is present within playlists, in relation to the variation between playlists. These variations are used for weighting a

ranking algorithm developed for recommending similar works to a given seed. The intuition is that the more a dimension is homogeneous within playlists, the more that dimension is important for producing relevant recommendation. The system has been evaluated by a pool of experts.

- Other works about playlists. **Title2rec** is an algorithm for the generation of new playlists of which is given only the title. The strategy rely on word embeddings computed on titles and descriptions of of playlists, which have been clustered based on shared tracks. In another work, we demonstrated how is possible to predict the **emotion of a playlist** by analysing the lyrics of its tracks.

- Demo applications for accessing the DOREMUS dataset:
  - **OVERTURE**, a web application for exploratory search, which have been used for visualising DOREMUS data and hosting the recommender system;
  - **CityMUS**, a context-based recommender system for accessing to relevant music in a city, relying on graph paths between artists and PoIs;
  - the **DOREMUS chatbot** able to give you information about artists, works and next concerts.

- **MIDI2vec**, an approach for producing graph embeddings from MIDI files. The strategy foresees 1. the transformation of the time-continuous MIDI information in a graph and 2. the use of graph embedding techniques for producing the vectors. The embeddings proved to be effective in genre and metadata prediction, revealing that symbolic music can play a role in a field nowadays dominated by audio analysis techniques.

As further summary for reader convenience, the links to all resources and tools realised in the context of this research have been collected in Table 12.1.

12.2 First implications

At the time of writing, several of the results reported in this manuscript started to find adoption by the community and application in other works.

The DOREMUS ontology and the controlled vocabularies are being endorsed by IFLA, as a de-facto standard for this community. We already mentioned the interest of LD4P for the DOREMUS vocabularies in the context of their Performed Music Ontology (PMO), while first attempts of including them in the MIDI Linked Data Cloud [123] have been carried out.

The DOREMUS KG is currently used by librarians internally within each partner institution and across the three institutions, allowing for the fast retrieval of results for complex queries. The detailed information about classical music – a unique resource among music datasets

Resource or Tool	URL
DATA	
BnF: Works, Artists, Manifestations	http://data.doremus.org/bnf
Philharmonie: Works and Concerts	http://data.doremus.org/philharmonie
Euterpe: Foreseen Concerts (PP)	http://data.doremus.org/euterpe
Itema3: Concerts and recordings (RF)	http://data.doremus.org/itema3
Diabolo: Works (RF)	http://data.doremus.org/diabolo
DOREMUS SPARQL endpoint	http://data.doremus.org/sparql
Example queries	http://data.doremus.org/queries.html
Evaluation queries	https://git.io/fjoOz
DOREMUS ontology	http://data.doremus.org/ontology
DOREMUS vocabularies	http://data.doremus.org/vocabularies
Data dumps	https://github.com/DOREMUS-ANR/knowledge-base
TOOLS	
`marc2rdf` converter	https://github.com/DOREMUS-ANR/marc2rdf
`itema3` converter	https://github.com/DOREMUS-ANR/itema3converter
`euterpe` converter	https://github.com/DOREMUS-ANR/euterpe-converter
`diabolo` converter	https://github.com/DOREMUS-ANR/diabolo-converter
SPARQL Transformer	https://github.com/D2KLab/sparql-transformer
String2vocabulary	https://github.com/DOREMUS-ANR/string2vocabulary
OVERTURE search engine	http://overture.doremus.org
DOREMUS chatbot	http://chatbot.doremus.org
Emotion predictor	http://data.doremus.org/emotion

Table 12.1 – Links to resources and tools

captured the attention of other projects (e.g. RondoDB[1] and MusicBrainz) and companies (Deezer), interested in interlinking their dataset with DOREMUS.

Thanks to the exploratory search engine, the DOREMUS data is open for access to a wide community of musicians, music theorists, connoisseurs and amateurs, who do not need to have any technical expertise in order to query the RDF graphs.

SPARQL Transformer is already deployed in two communities driven by H2020 projects which have adopted both SPARQL Transformer and `glrc`. MeMAD[2] uses it to generate automatically an API on top of a knowledge graph describing TV and radio programs which are also automatically annotated. The resulting semantic metadata is hence integrated in the professional Media Asset Management system Flow developed by Limecraft[3]. SILKNOW[4] [155] uses it to generate an API on top of a knowledge graph describing silk-related objects from 10 museums. The generated API is used to empower an exploratory search engine and a virtual assistant. In addition, SPARQL Transformer is progressively being adopted by small simple projects, which

[1] https://www.rondodb.com/
[2] https://memad.eu/
[3] https://www.limecraft.com/
[4] http://silknow.eu/

are clearly the main target of this work[5].

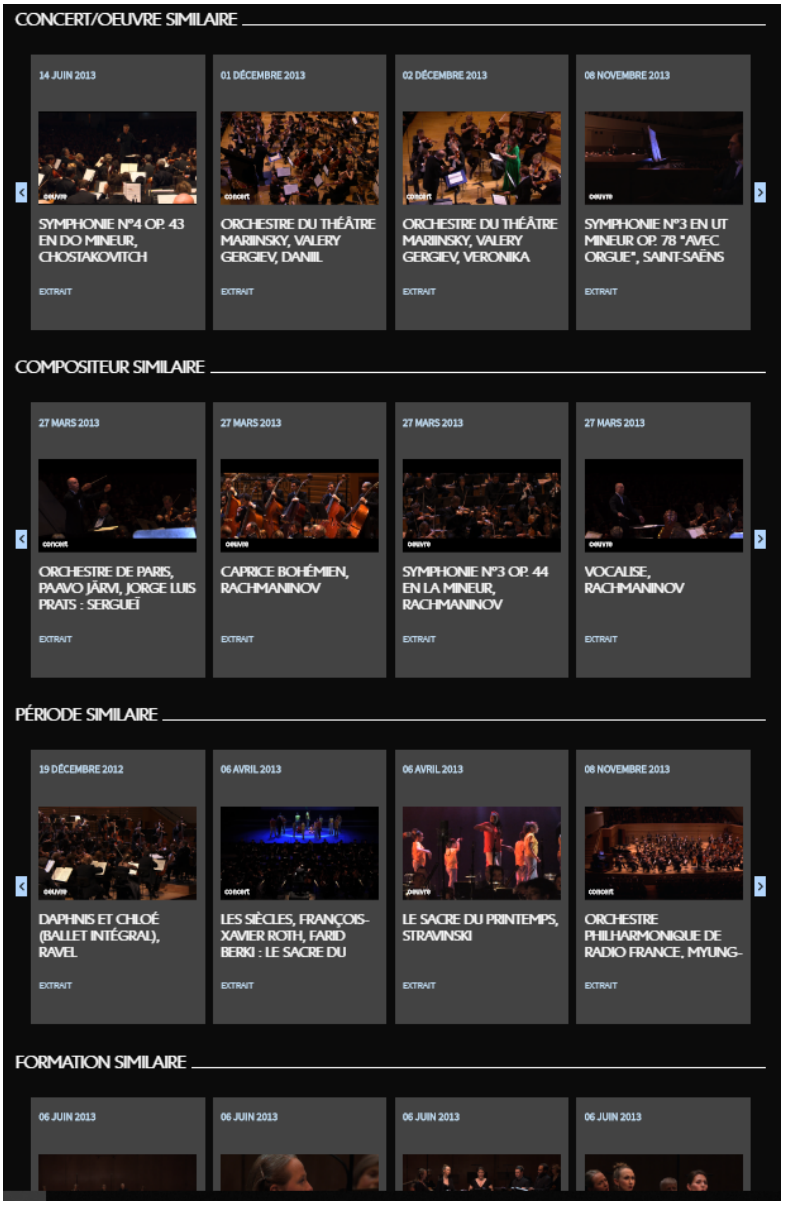


Figure 12.1 – Recommendation in *Philharmonie Live*

This ranking system is currently being considered for integration in *Philharmonie Live*[6], the multimedia portal of PP, for accessing audio and video recordings of events that took place in the concert hall. The system gives recommendation for relevant recordings to listen after the current one, on the base of each single feature (composer, period, casting, etc.) and of the combination of all of them (Figure 12.1). The different recommendations are realised by

[5] An up-to-date list is available at https://github.com/D2KLab/sparql-transformer/network/dependents
[6] https://live.philharmoniedeparis.fr/

weighting differently the similarity metric.

12.3 Limitations and Further Perspectives

The work presented in this manuscript could be extended or improved in many ways. In previous chapters, we reported limits and suggested future work relatively to the single contributions. Here, we give an overall look to the whole thesis, sum up the limitations and try to point out some research challenges for next future.

The DOREMUS model is capable of representing fine-grained information about music, with more detail in relation with ontologies designed for similar purposes. However, details bring complexity, and this can be a crucial threat to the adoption of the model and of the whole graph. We believe that an improved version of the DOREMUS may take benefit of the following ideas and strategies:

- When designing the ontology, domain experts tend to defend their point of view on the data, focusing their effort in the attempt of reproducing the source data structure (the same which they want to overcome), rather than reshaping the information in a way that can better suit the target format and a more general use. Although diluted by the presence of three different points of view which required harmonisation – the ones of libraries, concert halls and public radios –, this problem is present in DOREMUS, for example in the choice of FRBRoo as base. On our opinion, knowledge engineers should play a more determinant role in producing dataset beyond the simple transposition of structures from a format to another.
- A different solution – compatible with the previous one – is the adoption of a query-driven approach. Similarly to the Test-Driven Development, in which the software engineer defines interfaces and behaviours before the actual code content of methods and functions, the realisation of queries represents the first task, according to which the model and the mapping are derived [181].
- In Digital Humanities (DH) applications, where the importance of sources and attributions is equal to the one of the information itself, strategies for separately representing these two layers should be investigated, in order to obtain a trade-off between keeping queries simple and making the full information available where required. In this context, we consider interesting the recent development of RDF* and SPARQL* [77].

Differences between simple and complex data models – e.g. in relation to performances of systems based on these models – are an interesting research topic which has not been handled in this thesis. In this respect, our preliminary study about ontology mapping to *schema.org* should be continued, putting particular attention into making the strategy automated.

Some additional evaluation on the DOREMUS model may involve a comparison with its main competitors, in the first place the Music Ontology (MO). Although we know the differences in expressiveness of the two models, including MO as baseline would give more strength to the the model evaluation (Section 3.3).

The realisation of interconnected vocabularies about music genres and MoPs allowed the interoperability of dataset relying on different thesauri. However, the inclusion of these families of vocabularies in a KB results in the presence of identifiers belonging to different concept schemas and name-spaces. For example, you may find http://www.mimo-db.eu/InstrumentsKeywords/3582 (cello, from MIMO) and http://data.doremus.org/vocabulary/iaml/mop/vso (soprano, from IAML) for describing a single casting. This is not necessarily a bad thing, but may introduce confusion when looking at hierarchies, which may be incompatible and conflicting each other. We do not see any clear solution to this problem, while strategies for declaring multi-dimensional hierarchies may be investigated in a future research.

The data conversion lacks an extensive extraction of information from free-text fields, relying so far uniquely on empirical rules implemented with regular expressions. In source data, the kind information that is possible to find is mostly known – i.e. a given field contains the description of a performance with place, date, performer, roles. In this context, which is common in DH domains, the application of classical NLP techniques, enriched with the use of controlled vocabularies, may bring to interesting results.

Mistakes in source data (typos, unexpected field content) can be automatically detected thanks to missing matches with controlled vocabularies. We largely exploited this possibility, but without integrating an automatic error reporting mechanism, for extracting this cases and proposing corrections.

In classical music, titles are often including other kinds of metadata. For example, from the title of "*Sonata in C minor for piano*" we can know key, genre, and casting of the piece. We implemented and used a domain-specific strategy for extracting these metadata from work titles, we miss a proper evaluation of the approach, including a study about the possibility of extending it to other domains.

Speaking about limitations regarding SPARQL Transformer, some of them can be overcome in next developments, in particular the missing of some common SPARQL features (i.e. `UNION`). About the evaluation, we suffered the absence of a suitable benchmark for testing the library. We are aware about QALD Dataset [193]; however, this collection includes only queries selecting a single variable, not really exemplary for discussing the benefit of our approach. In addition, we are planning a user evaluation about the query writing, in order to improve the usability of the library. Indeed, we are aware that the learning curve for people with low confidence with Semantic Web structures may be steep, consequently we are planning to

produce more complete learning material. In addition, the feedback of community led us to think about decomposing SPARQL Transformer in two autonomous modules, the Parser and the Shaper, in order to exploit their features in different contexts, including Object-relational mapping (ORM) frameworks.

A crucial limitation to this research has been the absence of a proper dataset on which testing recommendation algorithms. We received and collected four datasets of playlists, coming from partner institutions or extracted from external services. The small dimension of the datasets and the almost irrelevant overlapping between them[7] forbid us to apply classical techniques and rely on popular metrics for evaluating our work. Any further research in this field can not avoid the collection of a ground truth dataset for classical music. This can be obtained in at least two ways: 1. extracting from existing dataset a relevant subset (which is not trivial given the lack metadata reported in Section 1.1.2) and 2. collaborating with music streaming services for the extraction of a new dataset from their databases.

More relevant experiments can be planned as future work. We would like to compare our ranking system with one in which are domain experts to assign the weights. In our work, we considered classical music as a unique block made of centuries of history, while experiments about changing the weights depending on the composition period of the seed may reveal more accurate performances. The integration of our embedding similarity strategy with *Title2rec* can produce an application in support to music experts, for producing playlists on the base of a title or few keywords. Finally, it would be interesting to apply the playlist emotion extraction strategy described in Section 9.6 also to lyrics-free music, for example relying on the music content represented in MIDI.

Even if it got started only towards the end of this PhD period, we believe that our work about MIDI embeddings is a promising challenge for introducing graph embeddings solution in MIR. In Chapter 11 we used an arbitrary mapping which aims to include all relevant information. However, we believe that a suitable graph representation of MIDI content with can benefit both the approach and its adoption. We identified in the MIDI ontology [121] the best candidate, which we intend to extend in collaboration with the authors. In addition, the interlinking between DOREMUS and the MIDI Data Cloud would enable new research about music recommendation relying on symbolic music and metadata. Further work involves the application of similar strategies to other symbolic music representation formats, like MusicXML.

[7] With overlapping, we mean the presence of the same track in multiple playlists.

Publications list

The research carried out during this reporting period has lead to the publication of the following scientific papers:

Journal

1. Lisena, P., Achichi, M., Choffé, P., Cecconi, C., Todorov, K., Jacquemin, B., Troncy, R. **Improving (Re-) Usability of Musical Datasets: An Overview of the DOREMUS Project**. In *BIBLIOTHEK - Forschung und Praxis*, AR 3189, March 2018.

Conference

1. Lisena, Pasquale: **Modeling, exploring and recommending music in its complexity.** In *20^{th} International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, Doctoral Consortium Track, November 19-23, 2016, Bologna, Italy.
   Also published in *Lecture Notes in Computer Science*, Vol. 10180

2. Lisena, P., Troncy, R., Todorov, K. Achichi, M. **Modeling the Complexity of Music Metadata in Semantic Graphs for Exploration and Discovery**. In *4^{th} International Workshop on Digital Libraries for Musicology (DLfM)*, Shanghai, China, October 28, 2017.

3. Lisena, P., Troncy, R. **Transforming the JSON Output of SPARQL Queries for Linked Data Clients.** In *The Web Conference (WWW)*, Developer Track, Lyon, France, April 23-27, 2018.

4. Lisena, P., Todorov, K., Cecconi, C. , Leresche, F., Canno, I., Puyrenier, F., Voisin, M., Le Meur, T., Troncy, R. **Controlled Vocabularies for Music Metadata**. In *19^{th} International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, September 23-27, 2018.

5. Monti, D., Palumbo, E., Rizzo, G., Lisena, P., Troncy, R., Fell, M., Cabrio, E., Morisio, M. **An Ensemble Approach of Recurrent Neural Networks using Pre-Trained Embeddings**

for Playlist Completion. In *12th ACM Conference on Recommender Systems (RecSys)*, Challenge Track, Vancouver, Canada, October 2-7, 2018.

6. Canale, L., Lisena, P., Troncy, R. **A Novel Ensemble Method for Named Entity Recognition and Disambiguation based on neural network**. In *17th International Semantic Web Conference (ISWC)*, Monterey, CA, USA, October 8-12, 2018.

7. Achichi, M., Lisena, P., Todorov, K., Troncy, R., Delahousse, J. **DOREMUS: A Graph of Linked Musical Works**. In *17th International Semantic Web Conference (ISWC)*, Resource Track, Monterey, CA, USA, October 8-12, 2018.

8. Lisena, P., Meroño-Peñuela, A., Kuhn, T., and Troncy, R. **Easy Web API Development with SPARQL Transformer**. In *18th International Semantic Web Conference (ISWC)*, In-Use Track, Auckland, New Zealand, October 26-30, 2019.

Posters and Demos

1. Lisena, P., Achichi, M., Fernandez, E., Todorov, K., Troncy, R. **Exploring Linked Classical Music Catalogs with OVERTURE.** In *15th International Semantic Web Conference (ISWC)*, Poster Track, October 17-21, 2016, Kobe, Japan.

2. Lisena, P., Troncy, R.: **DOREMUS to Schema.org: Mapping a complex vocabulary to a simpler one.** In *20th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, Poster Track, November 19-23, 2016, Bologna, Italy / Also published in LNCS, Vol.10180, Springer.

3. Lisena, P., Troncy, R. **Combining Music Specific Embeddings for Computing Artist Similarity**. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, Late-Breaking Demo Track, Suzhou, China, October 23-27, 2017.

4. Giammusso S., Guerriero M., Lisena P., Palumbo E., Troncy R. **Predicting The Emotion of Playlists Using Track Lyrics**. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Late-Breaking Demo Track, Paris, France, September 23-27, 2018.

5. Klotz, B., Lisena, P., Troncy, R., Wilms, D., Bonnet, C. **DrIveSCOVER: A Tourism Recommender System Based on External Driving Factors**. In *16th International Semantic Web Conference (ISWC)*, Poster Track, CEUR Proceedings Vol. 1963, Vienna, Austria, October 21-25, 2017.

6. Lisena, P., Canale, L., Ellena, F., Troncy, R. **CityMus: Music Recommendation when Exploring a City**. In *16th International Semantic Web Conference (ISWC)*, Poster Track, Vienna, Austria, October 21-25, 2017.

Tutorial

1. Lisena, P., Troncy, R. Tutorial on **DOing REusable MUSical data**. In *9th International Conference on Knowledge Capture (K-CAP'17)*, Austin, TX, USA, December 4-6, 2017.

2. Lisena, P., Troncy, R. Tutorial on **Music Knowledge Graph and Deep-Learning Based Recommender Systems**. In *15th Extended Semantic Web Conference (ESWC'18)*, Heraklion, Greece, June 3, 2018.

Résumé en français

12.1 Introduction

La musique est partout. Notre époque nous donne la possibilité d'accéder à la musique et de la reproduire à tout moment, n'importe où, à partir d'une multitude d'appareils connectés au réseau. Les récents progrès technologiques ont profondément modifié l'expérience d'écoute de la musique: au cours de la dernière décennie, nous sommes passés d'archives de musique locales (sauvegardées sur des supports physiques tels que des disques optiques et des lecteurs MP3, qui définissaient les limites en termes de stockage) à des catalogues potentiellement sans fin, appartenant à des services de musique en streaming, libres des contraintes des supports et dématérialisés dans des clouds informatiques. Dans ce contexte, le rôle des systèmes de recommandation dans la découverte de titres peut être déterminant. Par conséquent, augmente l'importance des données sur lesquelles reposent ces systèmes.

La musique classique est une niche dans le monde des services de musique en streaming. Ce créneau constitue en fait un super-genre qui regroupe une multitude de genres différents, du chant grégorien à la symphonie, du ballet à la musique de chambre, et implique des artistes avec un plus grand nombre de fonctions que leurs collègues de la musique moderne: compositeurs, chefs d'orchestre, instrumentistes, voix, solistes, membres d'orchestre, etc.

Les fans de musique classique sont sous-représentés sur les réseaux sociaux et les plateformes de diffusion de musique [176]. Les systèmes de recommandation nécessitent des stratégies spéciales pour traiter cette catégorie de musique, en tenant compte également de l'énorme matériel des siècles parmi lesquels la sélection des éléments pertinents [102]. La recherche sur le système de recommandation et la Music Information Retrieval (MIR) dans le cadre de la musique classique en est encore à son début, alors qu'elle attire de plus en plus l'attention.

Ce manuscrit de thèse porte principalement sur la musique classique et étudie comment représenter et exploiter ses informations. L'objectif principal est d'étudier les stratégies de représentation et de découverte de la connaissance, appliquées à la musique classique, dans des domaines tels que la population de base de connaissance, la prédiction de métadonnées et les systèmes de recommandation. Ce travail a contribué à la recherche avec les résultats

suivants:

- un modèle et un ensemble de vocabulaires contrôlés (réalisés grâce à l'expertise des institutions culturelles) pour décrire la musique en détail, qui utilisant les technologies du Web sémantique;
- un Knowledge Graph orienté á la musique classique et contenant des données sur les artistes, les œuvres, les performances, les partitions et les enregistrements. Le graphe, publié dans le Web des données, donne accès aux métadonnées détaillées provenant des plus importantes institutions culturelles françaises;
- un ensemble d'outils pour la conversion de données, la création d'une API sur le SPARQL endpoint, la visualisation et l'exploration des données;
- approches basées sur des plongement d'entités calculées sur des métadonnées structurées, pour classer et recommander de la musique;
- des applications de démonstration qui exploitent les approches et les ressources précédentes.

Cette recherche a été développée dans le cadre du projet DOREMUS[8] [2], au sein de laquelle trois grands instituts culturels en France, la BnF (Bibliothèques nationales de France), la Philharmonie de Paris (PP) et Radio France (RF), s'associent avec des entreprises et des institutions académiques afin de rendre disponible et réutilisable les connaissances musicales de leurs catalogues sur le web des données.

Ce document présente un résumé de mon travail de thèse. Dans Section 12.2, on présente le modèle DOREMUS pour décrire la musique, ainsi que des vocabulaires contrôlés spécifiques à la musique. Dans Section 12.3, on présente des outils de conversion de jeux de données de musique, en prenant comme exemple ceux provenant des riches archives musicales de l'institution partenaire de DOREMUS. On démontre l'expressivité du modèle en montrant comment il est possible de répondre à des requêtes complexes spécifiques à la musique. Enfin, nous décrivons les stratégies de visualisation et de recommandation de données dans Section 12.4. Des conclusions sont contenues dans Section 12.6.

12.2 Un modèle pour représenter les données musicales

Parmi les modèles RDF sur la musique, l'exemple le plus connu est la Music Ontology [161], qui fournit un ensemble de classes et de propriétés spécifiques à la musique pour décrire des œuvres musicales, des performances et des pistes, ainsi que des fragments de celles-ci. La nécessité d'exploiter davantage les connaissances musicales provenant des bibliothèques a conduit à la définition d'une nouvelle ontologie.

[8] http://www.doremus.org

12.2.1 The DOREMUS Ontology

Le modèle DOREMUS [9] est une extension de FRBRoo, permettant de décrire des objets culturels [55], appliquée au domaine spécifique de la musique. Il s'agit d'un modèle dynamique dans lequel l'intention abstraite de l'auteur (appelée œuvre) n'existe que par le presence d'un événement (c'est-à-dire la composition) qui le réalise dans une série distincte de choix appelée expression. Ce triplet œuvre-expression-événement peut également décrire différentes parties de la vie d'une œuvre, telles que la performance, la publication ou la création d'une œuvre dérivée, chacune d'elles incorporant l'expression dont elle provient.

En plus de les classes et propriétés originales de FRBRoo, des classes spécifiques ont été ajoutées afin de décrire les aspects spécifiquement liés à la musique, tels que la tonalité, le genre, le tempo, le moyen d'exécution (MoP, c'est a dire l'instrument), etc. [39].

Chaque triplet contient une information qui, dans le même temps, peut vivre de manière autonome et être liée aux autres entités. En pensant à une œuvre classique, nous aurons un triplet pour la composition, un pour tout événement de performance, un pour chaque manifestation (c'est-à-dire la partition), etc., tous reliés dans le graphe. Une improvisation jazz consistant en la création improvisée d'une nouvelle œuvre n'aura que le triplet liée à la performance, en l'absence du moment de la composition et de l'écriture de la partition qui est presque obligatoire pour la musique classique sans avoir besoin d'être rattaché à une autre entité. Il est considéré comme un travail en soi. Toutes les entités de travail de chaque triplet sont ensuite connectées à un travail complexe, une classe ayant pour objectif de rassembler toutes les représentations, conceptuelles et sensorielles (manifestations), de la même idée créative.

Le résultat est un modèle qui, si d'un côté est assez complexe et difficile à adopter, a d'autre part une expressivité très détaillée. Le graphe représenté dans Figure 12.2 montre un exemple réel provenant de nos données: la *Sonate pour piano et violoncelle n.1* de Beethoven[10].

12.2.2 Vocabulaires contrôlés pour les métadonnées musicales

Un grand nombre de propriétés impliquées dans la description de musique sont supposées contenir des valeurs partagées par différentes entités: différentes compositions peuvent avoir le genre “sonata", différents interprètes peuvent jouer un “basson", différents auteurs peuvent ont pour fonction “compositeur" ou “lyriciste". Ces étiquettes peuvent être exprimées dans plusieurs langues ou sous d'autres formes (par exemple, “sax" et“saxophone", ou les clés françaises “Do majeur" et “Ut majeur"), rendant la réconciliation difficile. Notre choix est d'utiliser des vocabulaires contrôlés pour ces concepts communs. Un vocabulaire contrôlé est

[9] http://data.doremus.org/ontology/

[10] http://data.doremus.org/expression/614925f2-1da7-39c1-8fb7-4866b1d39fc7

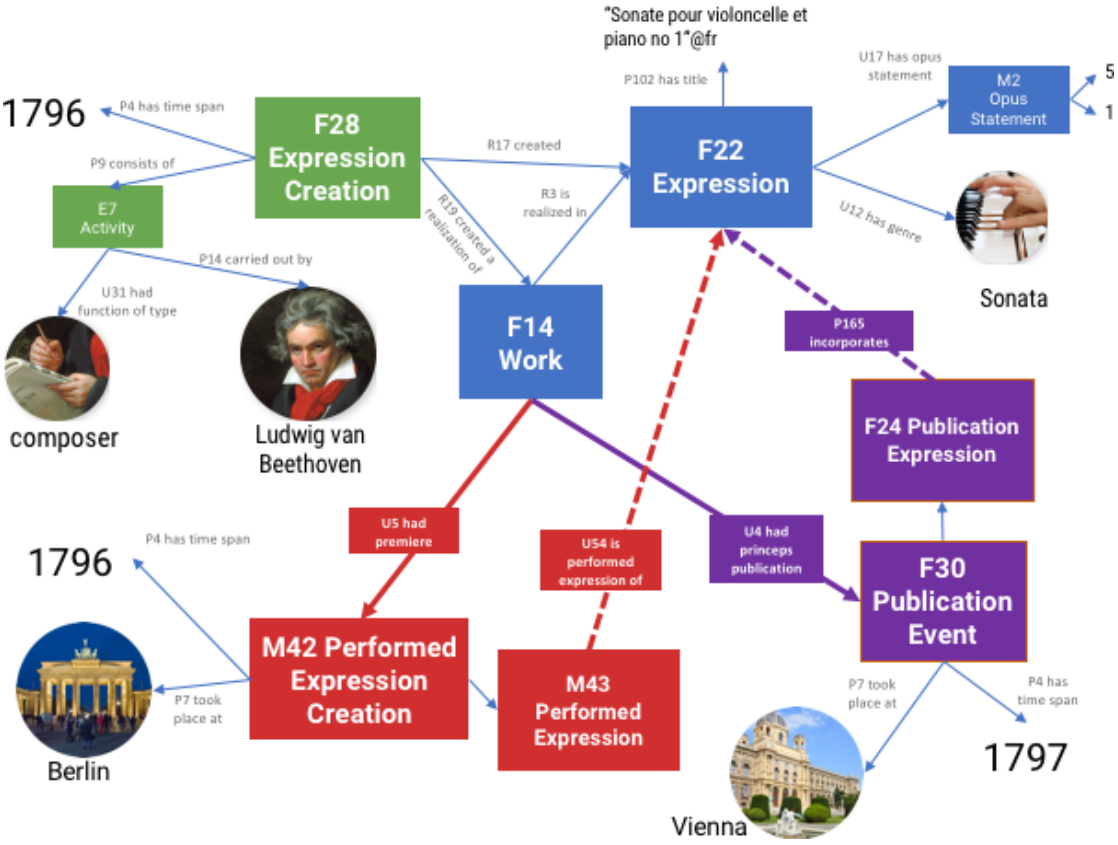


Figure 12.2 – Beethoven's *Sonata for piano and cello n.1* represented as a graph using the DOREMUS ontology

un thésaurus thématique d'entités, chacune étant à nouveau identifiée à un URI. On utilise SKOS [129] comme modèle de représentation, ce qui permet de spécifier pour chaque concept les libellés préférés et alternatifs dans plusieurs langues, et de définir une hiérarchie entre les concepts (de sorte que le "violon" est un concept appartenant à la notion plus large de "instrument à cordes"), et d'ajouter des commentaires et des notes pour décrire l'entité et aider l'activité d'annotation. Chaque concept devient un nœud commun dans le graphe musical qui peut connecter une œuvre musicale à une autre, un auteur à un interprète, etc.

Différents types de vocabulaires sont nécessaires pour décrire la musique. Certains d'entre eux sont déjà disponibles sur le Web: c'est le cas de MIMO[11] pour décrire les instruments de musique ou RAMEAU[12] pour les genres musicaux, les groupes ethniques, etc. Certains autres ne sont pas publiés dans un format adapté au Web of Data, ou la version publiée n'est pas aussi complète que d'autres formats disponibles dans les bibliothèques ou en ligne: cela se produit avec les vocabulaires publiés par l'Association internationale des bibliothèques musicales

[11] http://www.mimo-db.eu/
[12] urlhttp://rameau.bnf.fr/

(IAML)[13], publiés après le début du projet et pour lesquels nous fournissons parfois plus de détails (libellés, langues, etc.). Enfin, il y a aussi le cas de vocabulaires qui n'existent pas du tout et que nous générons à partir de données réelles provenant des partenaires, enrichies par un processus éditorial impliquant également les bibliothécaires. En conséquence, nous avons collecté, mis en œuvre et publié 23 vocabulaires contrôlés appartenant à 18 catégories différentes [107].

12.3 Conversion de données

La BnF et la Philharmonie de Paris utilisent le format MARC pour représenter les métadonnées de la musique. La structure plate de MARC, qui est une succession de champs et de sous-champs, reflète l'objectif de la conversion des enregistrements imprimés ou manuscrits en un formulaire informatique. Bien que ce soit un standard, son adoption est restreinte au monde entier, rendant sa sérialisation à d'autres formats (généralement XML) nécessaire pour une utilisation réelle. Les champs MARC ne sont pas étiquetés explicitement, mais codés avec des nombres, avec pour conséquence l'utilisation d'un manuel pour déchiffrer le contenu. La sémantique de ces champs et sous-champs n'est pas triviale: un sous-champ peut changer de signification sur le champ sous lequel il se trouve et sur la variante particulière de MARC (UNIMARC et INTERMARC). Un champ ou un sous-champ peut contenir des informations sur différentes entités, telles que la première performance et la première publication combinée dans le champ des notes, sans séparation claire. Souvent, les informations sont représentées sous la forme de texte libre [188].

Les avantages de passer de MARC à une solution basée sur RDF consistent en l'intero- pérabilité et l'intégration entre bibliothèques et avec des acteurs tiers, avec la possibilité de réaliser une recherche fédérée intelligente [8, 27]. Pour atteindre ces objectifs, deux tâches sont nécessaires: la conversion des données et la liaison des données.

12.3.1 De MARC à RDF

Pour la tâche de conversion, on utilise `marc2rdf`[14], un prototype open source que nous avons développé pour la conversion automatique des notices bibliographiques MARC en RDF utilisant l'ontologie DOREMUS [104, 112]. Le processus de conversion repose sur des règles de transfert explicites définies par des experts (ou mappings) qui indiquent où dans le fichier MARC rechercher quel type d'informations, en fournissant le chemin de propriété correspondant dans le modèle, ainsi que des exemples utiles illustrant chaque règle de transfert. Le rôle de ces règles va au-delà d'une simple documentation pour les notices MARC, intégrant

[13] http://iflastandards.info/ns/unimarc/
[14] https://github.com/DOREMUS-ANR/marc2rdf

également des informations sur certaines pratiques des bibliothécaires dans la formalisation du contenu: format des dates, accords sur la syntaxe des champs textuels, valeurs par défaut en cas d'absence d'informations.

Le convertisseur est composé de différents modules, qui fonctionnent successivement. Tout d'abord, un *file parser* lit le fichier MARC et rend le contenu accessible par champ et numéro de sous-champ. On a implémenté un module de conversion pour les variantes INTERMARC et UNIMARC. Ensuite, il construit le graphe RDF en lisant les champs et en affectant leur contenu à la propriété DOREMUS suggérée dans les règles de transfert.

Ensuite, le *free-text interpreter* extrait des informations supplémentaires des champs de texte, y compris des notes éditoriales. Cela revient à effectuer une analyse basée sur les connaissances, car nous recherchons exactement les informations que nous voulons instancier dans chaque chaîne (par example le MoP dans la note du casting, ou la date et l'éditeur dans la note de la première publication). L'analyse est réalisée avec d'expressions régulières définies de manière empirique. Enfin, le module *string2vocabulary* effectue un mappage automatique des littéraux de chaîne sur les URI provenant de vocabulaires contrôlés. Toutes les variantes d'une libelle de concept sont considérées afin de traiter les différences potentielles dans les termes de dénomination. En tant que fonctionnalité supplémentaire, ce composant est capable de reconnaître et de corriger le bruit présent dans le fichier MARC source: c'est le cas de certains tonalité déclarées comme genre, ou des champs pour le numéro d'opus qui contiennent en réalité un numéro de catalogue et vice-versa. Ces cas ainsi que d'autres fautes de frappe et erreurs ont été identifiés grâce au processus de conversion et à la visualisation des données converties, aidant ainsi l'institution source à mettre à jour et à corriger en permanence ses données.

12.3.2 Traiter les formats hétérogènes

En dehors de MARC, nous convertissons d'autres bases sources (en XML), trop spécifiques pour être gérées par un seul convertisseur. Par conséquent, nous avons développé des logiciels *ad hoc* doté d'un workflow générique: analyser le fichier et collecter les informations requises, créer la structure du graphe dans RDF, exécuter le module *string2vocabulary* décrit précédemment. Cette procédure crée différents graphes, un pour chaque source. Ces bases de données sources sont complémentaires mais comportent également des chevauchements (par exemple, deux bases de données décrivant le même travail ou les mêmes performances avec des métadonnées complémentaires) et ont été interconnecté automatiquement afin que le graphe de connaissances résultant fournisse une description plus détaillée de chaque travail.

Catégorie	Questions	Supporté par le modèle	Résultats dans les données
A. Works	31	31	23
B. Artists	3	2	1
C. Performances	9	8	6
D. Recordings	11	9	7
E. Publications	5	5	3

Table 12.2 – Pour chaque catégorie de questions, nous fournissons le rapport entre le nombre de requêtes en langage humain intelligible, le nombre de requêtes ayant été converties avec succès dans une requête DOREMUS et le nombre de celles-ci produisant au moins un résultat lorsque la requête est soumise au DOREMUS endpoint.

12.3.3 Répondre à des requêtes complexes

Avant le début du projet, une liste de questions avait été collectée auprès des experts des institutions partenaires [15]. Ces questions reflètent les besoins réels des institutions et révèlent les problèmes auxquels elles sont confrontées chaque jour pour sélectionner des informations dans la base de données (organisation de concerts ou programmation radiodiffusée, par exemple) ou pour soutenir des études de bibliothécaire et de musicologue. Ils peuvent être liés à des cas d'utilisation pratiques (la recherche de toutes les partitions correspondant à une formation particulière), à des thèmes liés au musicologue (la musique d'une région donnée dans une période historique donnée), à des statistiques intéressantes (les œuvres généralement interprétées ou publiées ensemble), ou à de curieuses connexions entre œuvres, performances ou artistes. La plupart des questions étant très spécifiques et complexes, il est très difficile de trouver une réponse à ces questions en interrogeant simplement les moteurs de recherche actuellement disponibles sur le Web. Nous avons regroupé ces questions en catégories, en fonction des classes DOREMUS concernées.

Table 12.2 fournit une vue d'ensemble du nombre de requêtes que nous pouvons actuellement écrire pour chaque catégorie. Peu d'entre eux ne trouvent aucun résultat dans les données. D'autres sont difficiles à écrire au format SPARQL car ils impliquent des détails spécifiques qui sont hors de la portée du modèle (par example *Récupérez les œuvres d'artistes qui se sont mutuellement amoureux*). Le taux de conversion est de toute façon plus que positif.

12.4 Exploration et Recommendation

On considére l'exploration et la recommandation comme les deux faces d'une même médaille. Dans le premier cas, on permette à l'utilisateur de parcourir les jeux de données, de découvrir lui-même les connexions, de comprendre comment nous construisons les connaissances. Par

[15] https://github.com/DOREMUS-ANR/knowledge-base/tree/master/query-examples

recommandation, on retire cette responsabilité à l'utilisateur, dans le but de présenter ce dont il a besoin à un moment donné.

12.4.1 Visualiser la complexité

Nous avons développé OVERTURE[16] [104], un prototype Web de moteur de recherche exploratoire pour les données DOREMUS. L'application envoie des demandes directement à le SPARQL endpoint et fournit les resultat dans une interface utilisateur agreable.

En haut de l'interface utilisateur, la barre de navigation permet à l'utilisateur de naviguer entre les principaux concepts du modèle DOREMUS: expression, performance, partition, enregistrement, artiste. Le défi consiste à donner à l'utilisateur final une vision complète des données de chaque classe et à lui faire comprendre comment elles sont connectées les unes aux autres. On garde comme exemple *Sonata pour piano et violoncelle n.1*[17]. Outre les différentes versions du titre, le compositeur et une description textuelle, la page fournit des détails sur les informations dont nous disposons sur le travail, telles que la tonalité, les genres, le MoP prévu, le numéro d'opus. Lorsque ces valeurs proviennent d'un vocabulaire contrôlé, un lien est présenté afin de rechercher des expressions partageant la même valeur (par exemple, le même genre ou la même tonalité). Une chronologie montre les événements les plus importants liés au travail (la composition, la creation, la première publication). D'autres performances et publications peuvent être représentées ci-dessous. L'arrière-plan est un portrait du compositeur qui vient de DBpedia. Il est récupéré grâce à la présence dans la base de données DOREMUS de liens `owl: sameAs`. Ces liens proviennent en partie du service ISNI (International Standard Name Identifier) [18], et en partie depuis l'interconnexion réalisée en faisant correspondre le nom de l'artiste, la date de naissance et la date de décès dans l'année. les différents jeux de données.

12.4.2 Plongements de graphe pour le calcul de similarité

Que devrions-nous suggérer à un utilisateur écoutant Beethoven? Des musiciens similaires devraient partager certaines caractéristiques avec le compositeur allemand: la période, des propriétés similaires sur les compositions (genre, tonalité, casting) ou un instrument similaire joué (le piano lui-même ou le clavecin de la même famille). Mais comment définir une mesure de similarité prenant en compte ces concepts? On propose une solution [107, 109] basée sur des plongements de graphes générés à différents niveaux:

[16] http://overture.doremus.org

[17] http://overture.doremus.org/expression/614925f2-1da7-39c1-8fb7-4866b1d39fc7

[18] La base de données ISNI contient des informations sur les personnes impliquées dans les processus de création (artistes, par exemple). Il est géré par l'équipe qualité ISNI, dont la BnF est membre, et les artistes enregistrés dans la base de données BnF contiennent généralement une référence ISNI.

1. Pour des caractéristiques simples (genre, tonalité, instrument, par exemple), on calcule pour chaque terme une imbrication intégrant *node2vec* [70] sur deux sous-graphes: celui des vocabulaires contrôlés et celui correspondant à l'utilisation de leurs valeurs dans le jeu de données DOREMUS;

2. Pour les caractéristiques complexes (par exemple, l'artiste), on génére les incorporations en combinant les entités correspondantes. Dans le cas des artistes, on génére un vecteur composé de la date de naissance et de décès, du lieu de naissance et du lieu de la mort, du genre, de la tonalité et du casting (MoP) de sa composition, ainsi que de l'instrument joué;

3. Enfin, pour les œuvres, on combine encore une fois des fonctionnalités simples et complexes, en suivant les mêmes règles. Nous prenons en compte la date de composition, le genre, le casting, l'instrument soliste, la tonalité, le compositeur.

L'utilisation des plongements de graphe réduit le problème de similarité à l'inverse d'une distance euclidienne. Si certaines propriétés manquent, on applique une pénalisation calculée en pourcentage de le caractéristiques manquante dans le vecteur cible par rapport à le seed.

Le principal avantage de cette méthode est que le calcul des plongements n'est requis que pour les caractéristiques simples: chaque plongements peut être réutilisée dans des combinaisons différente. En utilisant la distance euclidienne pondérée, différents poids peuvent être attribués à chaque propriété afin d'ajuster la recommandation:

$$d(s,t,w) = \sqrt{w(s-t)^2} = \sqrt{\frac{1}{N}\sum_x w_x(s_x - t_x)^2} \tag{12.1}$$

produisant la fonction de similarité suivante:

$$similarity(s,t) = \frac{d_{max} - d(s,t,w)}{\mid d_{max} \mid} * (1 - penalty(s,t)) \tag{12.2}$$

12.5 Analyser les playlists de musique classique

Les experts humains ont toujours joué un rôle central dans l'établissement de listes d'œuvres musicales pouvant servir à différentes fins, telles que la programmation de concerts, la radiodiffusion ou la production de playlists éditoriales. Notre intuition est qu'il existe certaines règles cachées qui sont suivies lors de la création d'une playlist et qui déterminent quel artiste

ou quelle œuvre doit en suivre un autre. Ces règles découlent directement des connaissances des experts eux-mêmes, qui peuvent les appliquer consciemment ou non, et qui peuvent ne pas être en mesure de les décrire. Nous pensons que ces règles peuvent être extraites en étudiant le contenu des playlists.

Nous avons collecté quatre jeux de données contenant une liste d'œuvres: 2 jeux de données de concerts de PP et RF, un programme de radio Web (de RF) et une de playlists éditorials (de Spotify). On analyse la différence de variance *within* (dans) σ_W^2 et *between* (entre) σ_B^2 les playlists, conformément à la définition de ANalysis Of VAriance (ANOVA)[19]. On définit les poids de Equation (12.1) proportionnellement au **rapport de variance**:

$$\sigma_{ratio}^2 = \frac{\sigma_B^2}{\sigma_W^2} \tag{12.3}$$

La similarité euclidienne est utilisée pour classer les éléments dans un système de génération de listes de lecture basé sur le contenu simple dans lequel, dans le cas d'un œuvre seed s, les éléments les plus similaires ont plus de chances de figurer en haut de la liste. Les candidats sont choisis parmi un groupe d'œuvres cibles T, en les classant avec la fonction de similarité et en sélectionnant les premiers résultats.

Le système de classement a été évalué par deux groupes, composés d'experts provenant respectivement du monde de la radiodiffusion (Radio France, 4 membres) et des salles de concert (Philharmonie de Paris, 3 membres). Nous avons préparé une interface d'evaluation[20] composée de 10 étapes. Chaque étape montre un élément de départ (artiste ou œuvre) et les 10 premiers éléments cibles, classés par score de similarité. Certaines étapes ont utilisé la mesure de similarité non pondérée, afin de comparer avec la version pondérée. Les évaluateurs sont invités 1. à supprimer les éléments erronés en les faisant glisser dans une corbeille et 2. à trier les éléments restants par ordre de pertinence. Les résultats montrent une préférence générale pour la version pondérée du classement.

12.6 Conclusion

Représenter l'information sur la musique classique est une activité complexe, impliquant différentes sous-tâches. Nous avons proposé un flux de travail complet pour la gestion des métadonnées de musique utilisant les technologies du Web sémantique. Nous avons développé une ontologie spécialisée et un ensemble de vocabulaires contrôlés pour les différents concepts spécifiques à la musique. Ensuite, nous avons proposé une approche de conversion des données, afin d'aller au-delà de la pratique bibliothécaire actuellement utilisée. Enfin,

[19] https://people.richland.edu/james/lecture/m170/ch13-1wy.html
[20] http://overture.doremus.org/evaluation

nous montrons comment ces données peuvent être exploitées, permettant à l'utilisateur final d'explorer les données et d'obtenir une recommandation musicale.

La voie pour faire de la musique classique un citoyen de première classe dans MIR ne fait que commencer. Nous sommes convaincus que les études contenues dans ce manuscrit inspireront des recherches plus poussées.

Bibliography

[1] Sunitha Abburu and G Suresh Babu. Format SPARQL Query Results into HTML Report. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(6):144–148, 2013.

[2] Manel Achichi, Rodolphe Bailly, Cécile Cecconi, Marie Destandau, Konstantin Todorov, and Raphaël Troncy. DOREMUS: Doing Reusable Musical Data. In *14th International Semantic Web Conference (ISWC)*, Bethlehem, PA, USA, 2015.

[3] Manel Achichi, Zora Bellahsene, Mohamed Ben Ellefi, and Konstantin Todorov. Linking and disambiguating entities across heterogeneous RDF graphs. *Journal of Web Semantics*, 55:108–121, 2019.

[4] Manel Achichi, Mohamed Ben Ellefi, Danai Symeonidou, and Konstantin Todorov. Automatic Key Selection for Data Linking. In *20th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 3–18, Bologna, Italy, 2016.

[5] Manel Achichi, Pasquale Lisena, Konstantin Todorov, Raphaël Troncy, and Jean Delahousse. DOREMUS: A Graph of Linked Musical Works. In *17th International Semantic Web Conference (ISWC)*, Monterey, CA, USA, 2018.

[6] Alessandro Adamou, Mathieu d'Aquin, Helen Barlow, and Simon Brown. LED: curated and crowdsourced linked data on music listening experiences. In *13th International Semantic Web Conference (ISWC), Posters & Demos Track*, volume 1272, pages 93–96, Riva del Garda, Italy, 2014. CEUR-WS.org.

[7] Gediminas Adomavicius and Alexander Tuzhilin. *Context-Aware Recommender Systems*, pages 217–253. Springer, 2011.

[8] Getaneh Alemu, Brett Stevens, Penny Ross, and Jane Chandler. Linked Data for libraries: Benefits of a conceptual shift from library-specific record structures to RDF-based data models. *New Library World*, 113(11/12):549–570, 2012.

[9] Alo Allik, György Fazekas, and Mark B Sandler. An Ontology for Audio Features. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, NY, USA, 2016.

[10] Alo Allik, Florian Thalmann, and Mark Sandler. MusicLynx: Exploring Music Through Artist Similarity Graphs. In *The Web Conference (WWW)*, pages 167–170, Lyon, France, 2018.

[11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *6th International Semantic Web Conference (ISWC) and 2nd Asian Semantic Web Conference (ASWC)*, pages 722–735, Busan, Korea, 2007. Springer-Verlag.

[12] David Bainbridge, Xiao Hu, and J. Stephen Downie. A Musical Progression with Greenstone: How Music Content Analysis and Linked Data is Helping Redefine the Boundaries to a Music Digital Library. In *1st International Workshop on Digital Libraries for Musicology (DLfM)*, pages 1–8. ACM, 2014.

[13] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. InCarMusic: Context-Aware Music Recommendations in a Car. In *12th International Conference on Electronic Commerce and Web Technologies (EC-Web)*, pages 89–100, Toulouse, France, 2011. Springer.

[14] Judit Bar-Ilan, Mazlita Mat-Hassan, and Mark Levene. Methods for comparing rankings of search engine results. *Computer Networks*, 50(10):1448–1463, 2006.

[15] Zohra Bellahsene, Vincent Emonet, Duyhoa Ngo, and Konstantin Todorov. YAM++ Online: A Web Platform for Ontology and Thesaurus Matching and Mapping Validation. In *14th Extended Semantic Web Conference (ESWC), Poster & Demo Track*, pages 137–142, Portoroz, Slovenia, 2017. Springer.

[16] Thomas Bergwinkl, Michael Luggen, elf Pavlik, Blake Regalia, Piero Savastano, and Ruben Verborgh. Interface Specification: RDF Representation, Draft Report. Technical report, W3C, 2017.

[17] Tim Berners-Lee. Linked Data - Design Issues, 2010. http://www.w3.org/DesignIssues/LinkedData.html.

[18] Federico Bianchi, Matteo Palmonari, and Debora Nozza. Towards Encoding Time in Text-Based Entity Embeddings. In *17th International Semantic Web Conference (ISWC)*, Monterey, CA, USA, 2018.

[19] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227, 2009.

[20] Christian Bizer and Andy Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *3rd International Semantic Web Conference (ISWC), Posters & Demos Track*, volume 2004, Hiroshima, Japan, 2004. Springer.

[21] Dmitry Bogdanov and Perfecto Herrera. Taking advantage of editorial metadata to recommend music. In *9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, London, UK, 2012.

[22] Geoffray Bonnin and Dietmar Jannach. Automated Generation of Music Playlists: Survey and Experiments. *ACM Computing Surveys (CSUR)*, 47(2):26:1–26:35, 2014.

[23] David Booth, Christopher G. Chute, Hugh Glaser, and Harold Solbrig. Toward Easier RDF. In *W3C Workshop on Web Standardization for Graph Data*, Berlin, Germany, 2019.

[24] Sarah Bouraga, Ivan Jureta, Stéphane Faulkner, and Caroline Herssens. Knowledge-Based Recommendation Systems: A Survey. *International Journal of Intelligent Information Technologies (IJIIT)*, 10(2):1–19, 2014.

[25] Matthias Braunhofer, Marius Kaminskas, and Francesco Ricci. Location-aware music recommendation. *International Journal of Multimedia Information Retrieval*, 2(1):31–44, 2013.

[26] David Bretherton, Daniel Alexander Smith, Richard Polfreman, Mark Everist, Jeanice Brooks, Joe Lambert, et al. Integrating musicology's heterogeneous data sources for better exploration. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 27–32, 2009.

[27] Gillian Byrne and Lisa Goddard. The strongest link: Libraries and linked data. *D-Lib magazine*, 16(11):5, 2010.

[28] Rui Cai, Chao Zhang, Chong Wang, Lei Zhang, and Wei-Ying Ma. MusicSense: Contextual Music Recommendation using Emotional Allocation Modeling. In *15th ACM International Conference on Multimedia (MM)*, pages 553–556, Augsburg, Germany, 2007. ACM.

[29] Erion Çano and Maurizio Morisio. Music mood dataset creation based on last.fm tags. In *International Conference on Artificial Intelligence and Applications (AIAP)*, Vienna, Austria, 2017.

[30] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[31] Zehra Cataltepe, Yusuf Yaslan, and Abdullah Sonmez. Music Genre Classification Using MIDI and Audio Features. *EURASIP Journal on Advances in Signal Processing*, 2007(1):036409, 2007.

[32] Òscar Celma. *Music recommendation and discovery in the long tail*. PhD thesis, Universitat Pompeu Fabra, 2009.

[33] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *12^{th} ACM Conference on Recommender Systems (RecSys), Challenge Track*, pages 527–528, Vancouver, British Columbia, Canada, 2018. ACM.

[34] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *22^{nd} ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016.

[35] Zhiyong Cheng and Jialie Shen. On Effective Location-Aware Music Recommendation. *ACM Transactions on Information Systems (TOIS)*, 34(2):13:1–13:32, April 2016.

[36] Samira Si-said Cherfi, Christophe Guillotel, Fayçal Hamdi, Philippe Rigaux, and Nicolas Travers. Ontology-based annotation of music scores. In *Knowledge Capture Conference (K-CAP)*, pages 10:1–10:4, Austin, TX, USA, 2017. ACM.

[37] Pierre Choffé. Introducing DOREMUS, an FRBRoo Extension for Music. In *24^{th} General Conference of the International Council of Museums (ICOM)*, Milan, Italy, 2016.

[38] Pierre Choffé and Marie Destandau. Introducing DOREMUS, a Rich Ontology for Music. In *65^{th} international IAML meeting*, Rome, Italy, 2016.

[39] Pierre Choffé and Françoise Leresche. DOREMUS: Connecting Sources, Enriching Catalogues and User Experience. In *24^{th} IFLA World Library and Information Congress*, Columbus, OH, USA, 2016.

[40] Michael Cochez, Martina Garofalo, Jérôme Lenßen, and Maria Angela Pellegrino. A First Experiment on Including Text Literals in KGloVe. In *4^{th} Workshop on Semantic Deep Learning (SemDeep)*, Monterey, CA, USA, 2018.

[41] Florian Colombo, Johanni Brea, and Wulfram Gerstner. Learning to Generate Music with BachProp. In *16^{th} Sound and Music Computing Conference (SMC)*, pages 380–386, Malaga, Spain, 2019.

[42] Olivier Corby, Catherine Faron-Zucker, and Fabien Gandon. A Generic RDF Transformation Software and Its Application to an Online Translation Service for Common Languages of Linked Data. In *14^{th} International Semantic Web Conference (ISWC)*, pages 150–165, Bethlehem, PA, USA, 2015.

[43] Olivier Corby, Catherine Faron-Zucker, and Fabien Gandon. LDScript: a Linked Data Script Language. In *16th International Semantic Web Conference (ISWC)*, pages 208–224, Vienna, Austria, 2017.

[44] Débora Cristina Corrêa and Francisco Aparecido Rodrigues. A survey on symbolic data-based music genre classification. *Expert Systems with Applications*, 60:190 – 210, 2016.

[45] Tim Crawford, Golnaz Badkobeh, and David Lewis. Searching Page-Images of Early Music Scanned with OMR: A Scalable Solution Using Minimal Absent Words. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[46] Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. Location Embeddings for Next Trip Recommendation. In *The Web Conference Companion (WWW Companion)*, pages 896–903, San Francisco, CA, USA, 2019. ACM.

[47] Enrico Daga, Luca Panziera, and Carlos Pedrinaci. A BASILar Approach for Building Web APIs on top of SPARQL Endpoints. In *International Workhop on Services and Applications over Linked APIs and Data (SALAD)*, volume 1359, Bethlehem, Pennsylvania, USA, 2015. CEUR Workshop Proceedings.

[48] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[49] Marilena Daquino, Enrico Daga, Mathieu d'Aquin, Aldo Gangemi, Simon Holland, Robin Laney, Albert Meroño-Peñuela, and Paul Mulholland. Characterizing the Landscape of Musical Data on the Web: State of the Art and Challenges. In *2nd Workshop on Humanities in the Semantic Web (WHiSe)*, Vienna, Austria, 2017.

[50] Marie Després-Lonnet, Béatrice Micheau, and Marie Destandau. Interoperability and organizational logics : What opening data really means. *Revue COSSI : communication, organisation, société du savoir et information*, 2017.

[51] Marie Destandau, Raphaël Troncy, Konstantin Todorov, Cécile Cecconi, Martine Voisin, Isabelle Canno, and Francoise Leresche. Linked Data Approach for Structuring and Interlinking Musical Catalogs. In *IFLA satellite Event "Data In Libraries: The Big Picture"*, Chicago, IL, USA, 2016.

[52] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked Open Data to Support Content-based Recommender Systems. In *8th International Conference on Semantic Systems (I-SEMANTICS)*, pages 1–8, Graz, Austria, 2012. ACM.

[53] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2005.

[54] Martin Doerr. The CIDOC Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata. *AI magazine*, 24(3):75–75, 2003.

[55] Martin Doerr, Chryssoula Bekiari, and Patrick LeBoeuf. FRBRoo: a conceptual model for performing arts. In *CIDOC Annual Conference*, pages 6–18, 2008.

[56] J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

[57] Ecma International. ECMAScript 2015 Language Specification. 6^{th} Edition. ECMA-262. Technical report, Ecma International, 2015.

[58] Luis Espinosa-Anke, Sergio Oramas, Horacio Saggion, and Xavier Serra. ELMDist: A Vector Space Model with Words and MusicBrainz Entities. In *The Semantic Web: ESWC 2017 Satellite Events*, pages 355–366, Portoroz, Slovenia, 2017.

[59] Beat Estermann. “OpenGLAM" in Practice–How Heritage Institutions Appropriate the Notion of Openness. In *20^{th} International Research Society for Public Management Conference (IRSPM)*, pages 13–15, Hong Kong, China, 2016.

[60] Michael Fell and Caroline Sporleder. Lyrics-based Analysis and Classification of Music. In *25^{th} International Conference on Computational Linguistics (COLING)*, Dublin, Ireland, 2014.

[61] Roy Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext transfer protocol (HTTP/1.1): Header Field Definitions. RFC 2616. Technical report, Internet Engineering Task Force, 2014.

[62] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures, 2000. PhD Thesis.

[63] Cristhian Figueroa, Iacopo Vagliano, Oscar Rodríguez Rocha, and Maurizio Morisio. A systematic literature review of Linked Data-based recommender systems. *Concurrency and Computation: Practice and Experience*, 27(17):4659–4684, 2015.

[64] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A Survey of Audio-Based Music Classification and Annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, 2011.

[65] Fabien Gandon, Franck Michel, Olivier Corby, Michel Buffa, Andrea Tettamanzi, Catherine Faron Zucker, Alain Giboin, Elena Cabrio, and Serena Villata. Graph Data on the Web: extend the pivot don’t reinvent the wheel. In *W3C Workshop on Web Standardization for Graph Data*, Berlin, Germany, 2019.

[66] Sara Giammusso, Mario Guerriero, Pasquale Lisena, Enrico Palumbo, and Raphaël Troncy. Predicting The Emotion of Playlists Using Track Lyrics. In *19th International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Demo Track*, Paris, France, 2018.

[67] Carol Jean Godby. *The Relationship between BIBFRAME and OCLC's Linked-Data Model of Bibliographic Description: A Working Paper*. OCLC Research, 2013.

[68] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78 – 94, 2018.

[69] Paul Groth, Antonis Loizou, Alasdair J.G. Gray, Carole Goble, Lee Harland, and Steve Pettifer. API-centric Linked Data integration: The Open PHACTS Discovery Platform case study. *Web Semantics: Science, Services and Agents on the World Wide Web*, 29(0):12 – 18, 2014.

[70] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016.

[71] R. V. Guha, Dan Brickley, and Steve MacBeth. Schema.Org: Evolution of Structured Data on the Web. *Queue - Structured Data*, 13(9):10:10–10:37, 2015.

[72] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[73] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: A Steerable Model for Bach Chorales Generation. In *34th International Conference on Machine Learning (ICML)*, pages 1362–1371. JMLR.org, 2017.

[74] Kelli Ham. OpenRefine (version 2.5). http://openrefine. org. Free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA*, 101(3):233, 2013.

[75] Steve Harris and Andy Seaborne. SPARQL 1.1 query language – W3C recommendation. Technical report, W3C, 2013.

[76] Peter Harrison and Marcus T Pearce. An energy-based generative sequence model for testing sensory theories of Western harmony. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[77] Olaf Hartig. The RDF* and SPARQL* Approach to Annotate Statements in RDF and to Reconcile RDF and Property Graphs. In *W3C Workshop on Web Standardization for Graph Data*, Berlin, Germany, 2019.

[78] Bernhard Haslhofer and Antoine Isaac. data.europeana.eu: The Europeana Linked Open Data Pilot. In *International Conference on Dublin Core and Metadata Applications (DC)*, volume 0, pages 94–104, The Hague, Netherlands, 2011. Dublin Core Metadata Initiative.

[79] Yang-Hui He. A Visualization of the Classical Musical Tradition. *ArXiv e-prints*, 2017.

[80] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. RelFinder: Revealing Relationships in RDF Knowledge Bases. In 4^{th} *International Conference on Semantic and Digital Media Technologies (SAMT)*, Graz, Austria, 2009.

[81] Patrick Helmholz, Sebastian Vetter, and Susanne Robra-Bissantz. AmbiTune: Bringing Context-Awareness to Music Playlists while Driving. In *International Conference on Design Science Research in Information Systems and Technology (DESRIST)*, pages 393–397, Miami, FL, USA, 2014. Springer.

[82] Allen Huang and Raymond Wu. Deep Learning for Music. *Computing Research Repository (CoRR)*, abs/1606.04930, 2016.

[83] Jochen Huelss and Heiko Paulheim. What SPARQL Query Logs Tell and Do Not Tell About Semantic Relatedness in LOD. In *Workshop on Negative or Inconclusive Results in Semantic Web (NoISE)*, pages 297–308, Portoroz, Slovenia, 2015.

[84] Kurt Jacobson, Simon Dixon, and Mark Sandler. LinkedBrainz: Providing the MusicBrainz Next Generation Schema as Linked Data. In 11^{th} *International Society for Music Information Retrieval Conference (ISMIR), Demo Session*, Utrecht, The Netherlands, 2010.

[85] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[86] Marius Kaminskas and Derek Bridge. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(1):2:1–2:42, 2016.

[87] Marius Kaminskas, Ignacio Fernández-Tobías, Francesco Ricci, and Iván Cantador. Knowledge-based music retrieval for places of interest. In 2^{nd} *International ACM Workshop on Music information retrieval with user-centered and multimodal strategies*, pages 19–24, 2012.

[88] Marius Kaminskas and Francesco Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2):89–119, 2012.

[89] Mayank Kejriwal and Pedro Szekely. Neural Embeddings for Populated Geonames Locations. In *16th International Semantic Web Conference (ISWC)*, pages 139–146, Vienna, Austria, 2017. Springer International Publishing.

[90] Houda Khrouf and Raphaël Troncy. Hybrid Event Recommendation Using Linked Data and User Diversity. In *7th ACM Conference on Recommender Systems (RecSys)*, pages 185–192, Hong Kong, China, 2013. ACM.

[91] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 255–266, Utrecht, The Netherlands, 2010.

[92] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. In *ACM SIGCHI Conference on Designing Interactive Systems (DIS)*, pages 555–565, Edinburgh, United Kingdom, 2017. ACM.

[93] Peter Knees and Markus Schedl. A Survey of Music Similarity and Recommendation from Music Context Data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(1):2:1–2:21, December 2013.

[94] Craig A. Knoblock, Pedro Szekely, Jose Luis Ambite, Shubham Gupta, Aman Goel, Maria Muslea, Kristina Lerman, Mohsen Taheriyan, and Parag Mallick. Semi-Automatically Mapping Structured Sources into the Semantic Web. In *9th Extended Semantic Web Conference*, Crete, Greece, 2012.

[95] Donald E. Knuth. Two Notes on Notation. *The American Mathematical Monthly*, 99(5):403–422, 1992.

[96] Sefki Kolozali, Mathieu Barthet, György Fazekas, and Mark B Sandler. Knowledge Representation Issues in Musical Instrument Ontology Design. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 465–470, 2011.

[97] Filip Korzeniowski and Gerhard Widmer. Genre-Agnostic Key Classification With Convolutional Neural Networks. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[98] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating Literals into Knowledge Graph Embeddings. *Computing Research Repository (CoRR)*, abs/1802.00934, 2018.

[99] Angela Kroeger. The Road to BIBFRAME: The Evolution of the Idea of Bibliographic Transition into a Post-MARC Future. *Cataloging & Classification Quarterly*, 51(8):873–890, 2013.

[100] Sebastian Ryszard Kruk and Bill McDaniel. *Goals of Semantic Digital Libraries*, pages 71–76. Springer, Berlin, Germany, 2009.

[101] Catherine Lai, Ichiro Fujinaga, David Descheneau, Michael Frishkopf, Jenn Riley, Joseph Hafner, and Brian McMillan. Metadata Infrastructure for Sound Recordings. In 8^{th} *International Society for Music Information Retrieval Conference (ISMIR)*, pages 157–158, 2007.

[102] Matthew Lasar. Digging into Pandora's Music Genome with musicologist Nolan Gasser. *Ars Technica*, 2011. https://arstechnica.com/tech-policy/2011/01/digging-into-pandoras-music-genome-with-musicologist-nolan-gasser/.

[103] Maxime Lathuilière. `Wikidata SDK`. https://github.com/maxlath/wikidata-sdk, 2015.

[104] Pasquale Lisena, Manel Achichi, Eva Fernandez, Konstantin Todorov, and Raphaël Troncy. Exploring Linked Classical Music Catalogs with OVERTURE. In 15^{th} *International Semantic Web Conference (ISWC)*, Kobe, Japan, 2016.

[105] Pasquale Lisena, Lorenzo Canale, Fabio Ellena, and Raphaël Troncy. Citymus: Music recommendation when exploring a city. In 16^{th} *International Semantic Web Conference (ISWC), Posters & Demos Track*, 2017.

[106] Pasquale Lisena, Albert Meroño-Peñuela, Tobias Kuhn, and Raphaël Troncy. Easy Web API Development with SPARQL Transformer. In 18^{th} *International Semantic Web Conference (ISWC), In-Use Track*, Auckland, New Zealand, 2019.

[107] Pasquale Lisena, Konstantin Todorov, Cécile Cecconi, Françoise Leresche, Isabelle Canno, Frédéric Puyrenier, Martine Voisin, and Raphaël Troncy. Controlled Vocabularies for Music Metadata. In 19^{th} *International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[108] Pasquale Lisena and Raphaël Troncy. DOREMUS to Schema.org: Mapping a Complex Vocabulary to a Simpler One. In 20^{th} *International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2016. Posters & Demos Track.

[109] Pasquale Lisena and Raphaël Troncy. Combining Music Specific Embeddings for Computing Artist Similarity. In 18^{th} *International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Demo Track*, Suzhou, China, 2017.

[110] Pasquale Lisena and Raphaël Troncy. DOing REusable MUSical Data (DOREMUS). In 9^{th} *Knowledge Capture Conference (K-CAP), Satellites: Workshops and Tutorials*, pages 1e:1–1e:6, Austin, TX, USA, 2017. ACM.

[111] Pasquale Lisena and Raphaël Troncy. Transforming the JSON Output of SPARQL Queries for Linked Data Clients. In *International Conference Companion on World Wide Web (WWW Companion)*, pages 775–780, Lyon, France, 2018. International World Wide Web Conferences Steering Committee.

[112] Pasquale Lisena, Raphaël Troncy, Konstantin Todorov, and Manel Achichi. Modeling the Complexity of Music Metadata in Semantic Graphs for Exploration and Discovery. In *4th International Workshop on Digital Libraries for Musicology (DLfM)*, pages 17–24, Shanghai, China, 2017.

[113] Laurie Lopatin. Metadata Practices in Academic and Non-Academic Libraries for Digital Projects: A Survey. *Cataloging & Classification Quarterly*, 48(8):716–742, 2010.

[114] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. A Demonstration of the Solid Platform for Social Web Applications. In *25th International Conference Companion on World Wide Web (WWW Companion)*, pages 223–226. International World Wide Web Conferences Steering Committee, 2016.

[115] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis. Conditioning Deep Generative Raw Audio Models for Structured Automatic Music. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[116] Brian McFee and Gert Lanckriet. Heterogeneous embedding for subjective artist similarity. In *10th International Symposium for Music Information Retrieval (ISMIR)*, Kobe, Japan, 2009.

[117] Deborah L McGuinness and Frank Van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), 2004. W3C recommendation.

[118] Cory McKay, John Burgoyne, Jason Hockman, Jordan B. L. Smith, Gabriel Vigliensoni, and Ichiro Fujinaga. Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010.

[119] Cory McKay and Ichiro Fujinaga. Automatic Genre Classification Using Large High-Level Musical Feature Sets. In 5th *International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004.

[120] Cory Mckay and Ichiro Fujinaga. jSymbolic: A Feature Extractor for MIDI Files. In *International Computer Music Conference*, pages 302–305, 2006.

[121] Albert Meroño-Peñuela, Marilena Daquino, and Enrico Daga. A Large-Scale Semantic Library of MIDI Linked Data. In *5th International Conference on Digital Libraries for Musicology (DLfM)*, Paris, France, 2018.

[122] Albert Meroño-Peñuela and Rinke Hoekstra. grlc Makes GitHub Taste Like Linked Data APIs. In *The Semantic Web – ESWC 2016 Satellite Events*, pages 342–353, Heraklion, Greece, 2016.

[123] Albert Meroño-Peñuela, Rinke Hoekstra, Aldo Gangemi, Peter Bloem, Reinier de Valk, Bas Stringer, Berit Janssen, Victor de Boer, Alo Allik, Stefan Schlobach, et al. The MIDI Linked Data Cloud. In *16th International Semantic Web Conference (ISWC)*, pages 156–164, Vienna, Austria, 2017. Springer.

[124] Béatrice Micheau, Marie Després-Lonnet, and Dominique Cotte. La recommandation musicale entre inscriptions documentaires, pratiques sociales, et dispositifs d'écoute. *Etudes de communication*, 49:33–56, 2017.

[125] MIDI Manufacturers Association. The Complete MIDI 1.0 Detailed Specification. Technical report, MIDI Manufacturers Association, Los Angeles, CA, USA, 1996-2014. https://www.midi.org/specifications/item/the-midi-1-0-specification.

[126] Nandana Mihindukulasooriya, Marıa Poveda-Villalón, Raúl Garcıa-Castro, and Asunción Gómez-Pérez. Loupe: An Online Tool for Inspecting Datasets in the Linked Data Cloud. In *14th International Semantic Web Conference (ISWC)*, Bethlehem, PA, USA, 2015.

[127] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[128] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *26th International Conference on Neural Information Processing Systems (NIPS)*, volume 2, pages 3111–3119, Lake Tahoe, NV, USA, 2013. Curran Associates Inc.

[129] Alistair Miles and José R Pérez-Agüera. Skos: Simple knowledge organisation for the web. *Cataloging & Classification Quarterly*, 43(3-4):69–83, 2007.

[130] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. An Ensemble Approach of Recurrent Neural Networks Using Pre-Trained Embeddings for Playlist Completion. In *12th ACM Conference on Recommender Systems (RecSys), Challenge Track*, pages 13:1–13:6, Vancouver, BC, Canada, 2018. ACM.

[131] Joshua L. Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 349–354, Porto, Portugal, 2012.

[132] Efthymia Moraitou, John Aliprantis, Yannis Christodoulou, Alexandros Teneketzis, and George Caridakis. Semantic Bridging of Cultural Heritage Disciplines and Tasks. *Heritage*, 2(1):611–630, 2019.

[133] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Rajasekar Venkatesan, Yang Liu Chen, and Shantanu Jaiswal. graph2vec: Learning Distributed Representations of Graphs. In *13th International Workshop on Mining and Learning with Graphs (MLG)*, Halifax, NS, Canada, 2017.

[134] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[135] Alberto Nogales, Miguel-Angel Sicilia, Elena García-Barriocanal, and Salvador Sánchez-Alonso. Exploring the Potential for Mapping Schema. org Microdata and the Web of Linked Data. In *Metadata and Semantics Research*, pages 266–276. Springer, 2013.

[136] Alberto Nogales, Miguel-Angel Sicilia, Salvador Sánchez-Alonso, and Elena Garcia-Barriocanal. Linking from Schema. org microdata to the Web of Linked Data: An empirical assessment. *Computer Standards & Interfaces*, 45:90–99, 2016.

[137] Terhi Nurmikko-Fuller, Daniel Bangert, and Alfie Abdul-Rahman. All the Things You Are: Accessing An Enriched Musicological Prosopography Through JazzCats. In *Digital Humanities*, Montreal, Canada, 2017.

[138] Terhi Nurmikko-Fuller, Daniel Bangert, Alan Dix, David Weigl, and Kevin Page. Building prototypes aggregating musicological datasets on the Semantic Web. *Bibliothek Forschung und Praxis*, 42(2):206–221, 2018.

[139] Terhi Nurmikko-Fuller, Daniel Bangert, Yun Hao, and J. Stephen Downie. Swinging Triples: Bridging Jazz Performance Datasets Using Linked Data. In *1st International Workshop on Semantic Applications for Audio and Music (SAAM)*, pages 42–45, Monterey, CA, USA, 2018. ACM.

[140] Terhi Nurmikko-Fuller, Alan Dix, David M. Weigl, and Kevin R. Page. In Collaboration with In Concert: Reflecting a Digital Library As Linked Data for Performance Ephemera. In *3rd International Workshop on Digital Libraries for Musicology (DLfM)*, pages 17–24, New York, NY, USA, 2016. ACM.

[141] Niels Ockeloen, Victor de Boer, and Lora Aroyo. LDtogo: A Data Querying and Mapping Framework for Linked Data Applications. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 199–203, Montpellier, France, 2013.

[142] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and Music Recommendation with Knowledge Graphs. *ACM Transactions on Intelligent Systems and Technology (TIST) - Survey Paper, Special Issue: Intelligent Music Systems and Applications and Regular Papers*, 8(2):21:1–21:21, 2016.

[143] Sergio Oramas, Mohamed Sordo, Luis Espinosa-Anke, and Xavier Serra. A Semantic-based Approach for Artist Similarity. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 100–106, Málaga, Spain, 2015.

[144] Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, Davide Romito, and Eugenio Di Sciascio. Cinemappy: a context-aware mobile app for movie recommendations boosted by dbpedia. In *ISWC Workshop Semantic Technologies meet Recommender Systems & Big Data (SeRSy)*, volume 919, pages 37–48, Boston, MA, USA, 2012.

[145] Alexander Pacha and Jorge Calvo-Zaragoza. Optical Music Recognition in Mensural Notation with Region-based Convolutional Neural Networks. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[146] Emilie Palagi, Fabien Gandon, Alain Giboin, and Raphaël Troncy. A Survey of Definitions and Models of Exploratory Search. In *ACM Workshop on Exploratory Search and Interactive Data Analytics (ESIDA)*, pages 3–8, Limassol, Cyprus, 2017. ACM.

[147] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *11th ACM Conference on Recommender Systems (RecSys)*, pages 32–36, Como, Italy, 2017. ACM.

[148] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. An Empirical Comparison of Knowledge Graph Embeddings for Item Recommendation. In *1st Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS)*, Heraklion, Greece, 2018.

[149] Jeff Z. Pan. Resource Description Framework. In *Handbook on Ontologies*, pages 71–90. Springer, 2009.

[150] Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks. In *25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Anchorage, AK, USA, 2019.

[151] Carlos Pedrinaci and John Domingue. Toward the Next Wave of Services: Linked Services for the Web of Data. *Journal of Universal Computer Science*, 16(13):1694—1719, 2010.

[152] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[153] Alfonso Perez-Carrillo, Florian Thalmann, György Fazekas, and Mark Sandler. Geolocation-Adaptive Music Player. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain, 2015.

[154] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710, New York, NY, USA, 2014. ACM.

[155] Cristina Portalés, Jorge Sebastián, Ester Alba, Javier Sevilla, Mar Gaitán, Paz Ruiz, and Marcos Fernández. Interactive Tools for the Preservation, Dissemination and Study of Silk Heritage—An Introduction to the SILKNOW Project. *Multimodal Technologies and Interaction*, 2(2), 2018.

[156] Jamie Powell. Classical music has a metadata problem. *Financial Times - Alphaville*, June 2019. Available at https://ftalphaville.ft.com/2019/06/25/1561435241000/Classical-music-has-a-metadata-problem/.

[157] Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1751–1756. Association for Computational Linguistics, 2017.

[158] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[159] Yves Raimond and Samer Abdallah. The Event Ontology. Technical report, Queen Mary University of London, 2007. http://motools.sourceforge.net/event.

[160] Yves Raimond and Samer Abdallah. The Timeline Ontology. Technical report, Queen Mary University of London, 2007. http://purl.org/NET/c4dm/timeline.owl.

[161] Yves Raimond, Samer A. Abdallah, Mark B. Sandler, and Frederick Giasson. The Music Ontology. In *15th International Conference on Music Information Retrieval (ISMIR)*, pages 417–422, Vienna, Austria, 2007.

[162] Yves Raimond and Mark Brian Sandler. A Web of Musical Information. In *9th International Conference of the Society for Music Information Retrieval (ISMIR)*, pages 263–268, 2008.

[163] Sabbir M. Rashid, David De Roure, and Deborah L. McGuinness. A Music Theory Ontology. In *1st International Workshop on Semantic Applications for Audio and Music (SAAM)*, pages 6–14, Monterey, CA, USA, 2018. ACM.

[164] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[165] Peter J. Rentfrow and Samuel D. Gosling. The Do Re Mi's of Everyday Life: The Structure and Personality Correlates of Music Preferences. *Journal of Personality and Social Psychology*, 84(6):1236–1256, 2003.

[166] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[167] Laurens Rietveld and Rinke Hoekstra. Man vs. Machine: Differences in SPARQL Queries. In *4^{t}h Workshop on Usage Analysis and the Web of Data (USEWOD)*, Anissaras, Greece, 2014.

[168] Laurens Rietveld and Rinke Hoekstra. The YASGUI family of SPARQL clients. *Semantic Web*, 8(3):373–383, 2017.

[169] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. RDF2Vec: RDF Graph Embeddings and Their Applications. *Semantic Web Journal*, 10(4):721–752, 2019.

[170] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *35 th International Conference on Machine Learning (ICML)*, Stockholmsmässan, Sweden, 2018.

[171] Jessica Rosati, Petar Ristoski, Tommaso Di Noia, Renato de Leone, and Heiko Paulheim. RDF Graph Embeddings for Content-based Recommender Systems. In *3rd Workshop on New Trends in Content-based Recommender Systems (CBRecSys)*, Boston, MA, USA, 2016.

[172] Charles Rosen. *The Classical Style: Haydn, Mozart, Beethoven*. WW Norton & Company, 1997.

[173] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[174] Muhammad Saleem, Muhammad Intizar Ali, Qaiser Mehmood, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. LSQ: Linked SPARQL Queries Dataset. In *14th International Semantic Web Conference (ISWC)*, pages 261–269, Bethlehem, Pennsylvania, USA, 2015.

[175] François Scharffe, Ghislain Atemezing, Raphaël Troncy, Fabien Gandon, Serena Villata, Bénédicte Bucher, Fayçal Hamdi, Laurent Bihanic, Gabriel Képéklian, Franck Cotton, et al. Enabling linked-data publication with the datalift platform. In *AAAI workshop on semantic cities*, 2012.

[176] Markus Schedl. Towards Personalizing Classical Music Recommendations. In *2nd International Workshop on Social Media Retrieval and Analysis (SoMeRA)*, Atlantic City, NJ, USA, 2015.

[177] Markus Schedl, Peter Knees, and Fabien Gouyon. New Paths in Music Recommender Systems Research. In *11th ACM Conference on Recommender Systems (RecSys)*, pages 392–393, Como, Italy, 2017. ACM.

[178] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. *Music Recommender Systems*, pages 453–492. Springer, 2015.

[179] Philip E. Schreur and Nancy Lorimer. Linked Data in Libraries' Technical Services Workflows. In *Metadata and Semantic Research*, pages 224–229, Cham, 2017. Springer International Publishing.

[180] Andy Seaborne. SPARQL 1.1 query results JSON format – W3C recommendation. Technical report, W3C, 2013.

[181] Juan F. Sequeda and Daniel P. Miranker. A pay-as-you-go methodology for ontology-based data access. *IEEE Internet Computing*, 21(2):92–96, Mar 2017.

[182] Seheon Song, Minkoo Kim, Seungmin Rho, and Eenjun Hwang. Music Ontology for Mood and Situation Reasoning to Support Music Retrieval and Recommendation. In *3rd International Conference on Digital Society (ICDS)*, pages 304–309, Cancun, Mexico, 2009. IEEE.

[183] Yading Song, Simon Dixon, Marcus Pearce, and Geraint A. Wiggins. A Survey of Music Recommendation Systems and Future Perspectives. In *9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, London, UK, 2012.

[184] Sebastian Speiser and Andreas Harth. Integrating Linked Data and Services with Linked Data Services. In *8th Extended Semantic Web Conference (ESWC)*, pages 170—184, Heraklion, Greece, 2011.

[185] Aaron Swartz. MusicBrainz: a Semantic Web Service. *IEEE Intelligent Systems*, 17(1):76–77, 2002.

[186] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. GraphQLLD: Linked Data Querying with GraphQL. In *17th International Semantic Web Conference (ISWC), Poster & Demo Track*, Monterey, California, USA, 2018.

[187] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. Bridges between GraphQL and RDF. In *W3C Workshop on Web Standardization for Graph Data*, Berlin, Germany, 2019.

[188] Roy Tennant. MARC must die. *Library Journal*, 127(17):26–27, 2002.

[189] Baker Thomas, Emmanuelle Bermes, Karen Coyle, Gordon Dunsire, Antoine Isaac, Peter Murray, Michael Panzer, Jodi Schneider, Ross Singer, Ed Summers, William Waites, Jeff

Young, and Marcia Zeng. W3C Library Linked Data Incubator Final Report. Technical report, World Wide Web Consortium (W3C), 2011.

[190] Abdel Nasser Tigrine, Zohra Bellahsene, and Konstantin Todorov. Light-Weight Cross-Lingual Ontology Matching with LYAM++. In *On the Move to Meaningful Internet Systems Conferences (OTM)*, pages 527–544, Rhodes, Greece, 2015. Springer International Publishing.

[191] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46-47:2 – 13, 2017.

[192] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building Comprehensive Knowledge Bases For City Exploration. *Journal of Web Semantics*, 2017.

[193] Christina Unger. Multilingual Question Answering over Linked Data: QALD-3 Dataset, 2013.

[194] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[195] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web*, 8(3):437–452, 2017.

[196] Ruben Verborgh. Decentralizing the semantic web through incentivized collaboration. In *17th International Semantic Web Conference (ISWC), Blue Sky Track*, volume 2189, October 2018.

[197] Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. Triple Pattern Fragments: a low-cost knowledge graph interface for the Web. *Journal of Web Semantics*, 37–38:184–206, 2016.

[198] Daniel Vila-Suero and Asunción Gómez-Pérez. datos.bne.es and MARiMbA: an insight into library linked data. *Library Hi Tech*, 31(4):575–601, 2013.

[199] Melanie Wacker, Jan Ashton, Ann Caldwell, Ray Denenberg, Angela Di Iorio, Rebecca Guenther, Myung-Ja Han, Sally McCallum, Tracy Meehleib, Stefanie Rühle, and Robin

Wendler. Metadata Object Description Schema (MODS). Specification, Library of Congress' Network Development and MARC Standards Office, 2013. http://www.loc.gov/standards/mods/modsrdf-primer.html.

[200] Mark Wick and Bernard Vatant. The GeoNames geographical database, 2012. http://geonames.org.

[201] Thomas Wilmering, György Fazekas, and Mark B. Sandler. AUFX-O: Novel Methods for the Representation of Audio Processing Workflows. In *15th International Semantic Web Conference (ISWC)*, pages 229–237, Kobe, Japan, 2016. Springer International Publishing.

[202] Austin Wright and Henry Andrews. JSON Schema: A Media Type for Describing JSON Documents. Technical report, Internet Engineering Task Force, 2017.

[203] Yujia Yan, Ethan Lustig, Joseph VanderStel, and Zhiyao Duan. Part-invariant Model for Music Generation and Harmonization. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[204] Mark Zadel and Ichiro Fujinaga. Web Services for Music Information Retrieval. In 5th *International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004.

[205] Eva Zangerle, Wolfgang Gassler, Martin Pichl, Stefan Steinhauser, and Günther Specht. An Empirical Evaluation of Property Recommender Systems for Wikidata and Collaborative Knowledge Bases. In *12th International Symposium on Open Collaboration (OpenSym)*, pages 18:1–18:8, Berlin, Germany, 2016. ACM.

[206] Amrapali Zaveri, Shima Dastgheib, Trish Whetzel, Ruben Verborgh, Paul Avillach, Gabor Korodi, Raymond Terryn, Kathleen Jagodnik, Pedro Assis, Chunlei Wu, and Michel Dumontier. smartAPI: Towards a more intelligent network of Web APIs. In *14th Extended Semantic Web Conference (ESWC)*, Portoroz, Slovenia, 2017.