Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

ONTOLOGY-BASED DATA INTEGRATION: MEMUK, THE MELODIC MUSIC KNOWLEDGE GRAPH

Relatore:
Chiar.ma Prof.essa
VALENTINA PRESUTTI

Correlatore:
Dott.
ANDREA POLTRONIERI

Presentata da:
ESTHER GIULIANO

III Sessione
Anno Accademico 2023 - 2024

"Ce qui pour les uns est la
même note, pour les autres est
l'infini de la musique."

Honoré de Balzac,
Le Père Goriot

Abstract

La crescente digitalizzazione nel consumo e nella produzione musicale ha aumentato le richieste nell'industria musicale di sistemi automatizzati per l'analisi e le proposte musicali. Ne consegue l'aumento di *datasets* musicali disponibili, la maggior parte dei quali ha dimensioni ridotte, restrizioni di copyrigth e/o scarsa interoperabilità dovuta a formati di annotazione diversi; trattasi, quindi, di *datasets* inadeguati nel settore del *Music Information Retrieval*.

L'integrazione di dati musicali da diverse fonti richiede tempo, denaro, e la collaborazione tra esperti musicali e informatici per evitare perdite o errori durante la conversione. Choco - *Chord Corpus and Data Transformation Workflow for Musical Harmony Knowledge Graphs* (un corpus contente dati armonici e un workflow per la creazione di un grafo di conoscenza basato su tali dati) - fornisce un'integrazione semantica su larga scala di *datasets* contenenti dati armonici, ossia l'aspetto verticale della musica.

Al contrario, appare ancora limitata la ricerca sull'integrazione semantica di dati melodici, riferiti all'aspetto orizzontale della musica. Questo lavoro propone un *workflow* per semplificare l'integrazione di dati melodici, e una procedura automatizzata di validazione dei risultati, con l'uso della libreria Music21 di Python, che supporta vari formati musicali digitali.

Viene presentato anche un esempio di integrazione semantica di tre *datasets* distinti, con la costruzione del *Melodic Music Knowledge Graph* (MEMUK, un grafo di conoscenza con dati musicali melodici), utilizzando Polifonia Ontology Network, per garantire l'interoperabilità con altri *knowledge graph*

basati sulla stessa ontologia, come il Choco Chord Corpus.

Abstract

Since music consumption and production became more digitised, the demand from music industries for automated systems to organise analyse and recommend music continues to grow, as well as the number of available musical datasets. However, most of them are not suitable for Music Information Retrieval because of their limited size, non-openness, and poor interoperability caused by the use of many different annotations formats.

Therefore the integration process of data from different sources is both time-consuming and financially expensive, as it requires collaboration between musical experts and computer scientist to reduce the loss of important musical information, and check for errors during the conversion. Choco - a Chord Corpus and Data Transformation Workflow for Musical Harmony Knowledge Graphs - already provides a large-scale semantic integration for harmonic data, i.e focusing on the vertical aspect of music.

Meanwhile, there appears to be limited explorations into the semantic integration of melodic data, which refers to the horizontal aspect of music, and is described by the music notation elements traditionally found in a musical score. This work proposes a modular integration workflow to simplify the standardisation process of melodic data, incorporating also a draft of an automated technical validation procedure, enabled by the Music21 Python library, which can parse many different digital music formats.

Furthermore, an example of semantic integration of three distinct datasets is presented, building the Melodic Music Knowledge Graph (MEMUK) by using the Polifonia Ontology Network, to ensure interoperability with other

Polifonia-based knowledge Graphs like the Choco Chord Corpus.

Contents

List of Figures

List of Tables

Introduction

Whether your goal is to find the title of a musical motif that has been haunting you during moments of boredom or to write a song for your loved ones without needing any musical knowledge, Music Information Retrieval (MIR) researchers have been developing solutions to address these common needs for the past fifty years. MIR is an interdisciplinary science that actively pursues a variety of different tasks, aiming to retrieve information from music. For example MIR researchers have been trying to craft reliable query-by-humming systems (allowing song identification via hummed or sung input). Another example is the development of automatic music composition tools. Other MIR tasks include music structure analysis, music recommendation algorithms, and audio content analysis [1].

These tasks are not only topics of experimental research, but also highly valued products in the music industry, as "The music industry is also quickly and constantly growing, supported by the new digital technologies and the rise of streaming platforms and digital services [...]." [2]. In 2021 the music industry generated around $65 billion through its core and complementary businesses combined, and employed almost 4 million people worldwide, a value second to visual arts only [3]. In addition, the music industry has been recently enjoying a constant growth, which is mainly due to the widespread adoption of streaming services (worth $13.4 billion in revenue, with a 19.9% growth in 2020 compared to the previous year). As a result, the core business of the industry was generating, in 2020, more than $21 billion worldwide — its biggest value ever — largely offsetting the losses reported by the decline of

the revenues in the sales of physical formats and revenues from performance rights, as a result of the COVID-19 pandemic [4]. Moreover, the number of users using music streaming services has increased from 76.8 million premium subscribers in 2015 to around 400 million in 2021 [5].

As music consumption and production became more digitised, the demand for automated systems to organise, analyse, and recommend music continues to grow. At the heart of each of these MIR tasks lies one essential ingredient: data. Music data includes a wide range of elements: such as metadata (bibliographic and library records), or data relating to concert events and venue. Other kinds of data are audio and/or video recordings and digital encodings of performances. Furthermore, data may include analytical and structural insights into music pieces, or capture the emotional and effective impact of music. Finally, digital sheet music encoding provides information on the structure and on the elements that compose a music piece.

Nevertheless, datasets generally consist of raw data, without an accurate description of relationships between their elements, failing to provide semantic understanding, facilitate reasoning and advanced querying. For this reason Choco also introduces a novel ontology for modelling music annotations, which facilitates the identification of semantic relationships [6]. These relationships, for example, help the identification of different songs and musical pieces that contain the same sequence of notes in the same or a similar order. For this reason, a semantic integration helps ensure that information across different music-related data sources is aligned and interpreted correctly.

However, finding large datasets in public domain to train and test an innovative MIR system is far from simple. Often, those MIR datasets are very small, containing around 100 entries, such as the labROSA:APT dataset [7] or the Chopin22 dataset [8]. In addition, each of these datasets uses a different annotation format (for instance music encoded with different file formats), with limited or no standardisation at the metadata level, making it hard to seamlessly integrate data from various sources. This requires highly skilled researchers with strong musical and programming knowledge. As a

result, integrating datasets is a time-consuming and financially demanding process.

Access to interoperable datasets can support the music industry in analysing product innovation [2], and promotes the reproducibility and replicability of experiments. Choco - a Chord Corpus Data Transformation Workflow for Musical Harmony Knowledge Graphs - already [6] provides a large scale semantic integration of datasets containing harmonic data, i.e. focusing on the "vertical" aspect of music. On the contrary, there appears to be limited explorations into the semantic integration of melodic data, which refers to the "horizontal" aspect of music, and is described by the music notation elements traditionally found in sheet music 1.2. Therefore, this thesis proposes a workflow to simplify the standardisation process of melodic data. Furthermore, an example of semantic integration of three distinct datasets is presented.

Therefore, the proposed workflow includes the transformation of the integrated dataset into a Knowledge Graph (KG), the Melodic Music Knowledge Graph (MEMUK) using the POLIFONIA ontology, which is designed "to represent a wide range of music-related concepts and relations" [9], ensuring interoperability with other Polifonia-based KG like the Choco Chord Corpus. Finally, the proposed methodology may be leveraged to favour the establishment of collaboration among artists to exchange knowledge to innovate music products, in an open innovation fashion [5].

Chapter 1

The Music Representation challenge

In Western culture, the search for a music representation system arose from the need of handing down melodies of Christian devotional chants. The first musical notation systems involved the use of neumes[1], placed above the words of the chant to indicate, in a rough manner, ascending and descending tones, and rhythmic figures. To address the imprecision of this system, Guido d'Arezzo (991/992 - after 1033) developed the four-line staff [11], four lines on which square-shaped notes were written at different pitches [10]. With Ugolino di Francesco da Orvieto (1380 - 1457), notation evolved into the five-line staff that we use today [12].

The traditional written or printed sheet music (also called as musical score) is not the only possible way to represent music. Any kind of digital music format may be regarded as "symbolic" since it is based on a finite alphabet of letters and symbols [13]. Actually, according to Müller M. [13], when talking about **symbolic** representations of music, we refer to formats that explicitly describe musical entities (also called musical events, as each of them has a start and an end, see Section 1.1). However, symbolic repre-

[1] A neume transcribes a melodic and rhythmic formula applied to a single syllable of a chant. [10]

sentations and musical sheets, do not offer all the necessary information to reproduce an acoustic realisation of a piece of music. Only audio representations (sound recordings) offer this feature, as they are composed of acoustic waves. This classification of three different types of music representations is based on the Müller proposal [13]. Nevertheless, each of these types of representation expresses only certain aspects of music, but none of them individually is music, because, as the philosophers Wiggins et al. argue [14]: "Music is actually something abstract and intangible which does not have a real existence in itself."

This chapter begins by introducing the concept of musical notation. Then the most common digital symbolic representations used to encode musical notation are presented. Finally, a comparison of these digital formats is conducted, to identify the one that best represents all musical notation elements, while ensuring compatibility across the diverse repertoire of Western Music.

1.1 Musical Notation

Musical notation refers to the system used to visually represent music elements. Mastering musical notation is not an easy task, and may require even several years. However, the goal of this section is not to make the reader a fluent music reader, but to provide him with a basic understanding of the elements that compose sheet music, enabling a better understanding of how different symbolic digital music representation formats work.

Sheet music is represented on **five-line staff** paper. Like when writing text on lined paper, staves are the five lines on which music symbols are written. In fact, additional indications can be found on top or below the staff, such as the lyrics of a song. A **note** is the first element that comes to mind when we think about music. In fact, this term is often used in a loose way, both to refer to the graphical symbol (♩) and to the pitched sound (when talking about audio representations) [13]. The main attributes of notes are **duration** and **pitch**.

The pitch is the sound produced, for instance, when playing a specific key on a piano, while the duration expresses how long to hold that key. Actually, the pitch is a periodic sound of a certain fundamental frequency, and the majority of human beings, thanks to a perceptual property, can order the sounds that they hear on a frequency-related scale [13].

In order to describe music notes using a finite number of symbols, the space of all possible pitches is discretised. Western music uses a discretisation called as **equal-temperament**, which means that the space between two notes with fundamentals frequencies in a ratio equal to any power of two is subdivided into twelve parts, each called semitone[13].

However, in Western music only seven literal symbols are used to represent notes: A, B, C, D, E, F, G (or seven syllables: la, si, do, re, mi, fa, sol). These are the white keys on a piano. To represent the five additional pitches, **accidentals** (♯, sharp or ♭, flat) are added to the notes. These correspond to the black keys on a piano and can be notated either as C♯, D♯, E♯, F♯, G♯, A♯, B♯, or as D♭, E♭, F♭, G♭, A♭, B♭, C♭.

Notes on a staff are organised into **measures** (also called as bars). The end of a measure is delimited by a vertical line: |. The length of a measure, i.e. how many notes can be inside of a bar, is defined by a **meter indication**, usually a fraction such as 3/4 or 6/8, called **time signature**, where the numerator indicates the number of **beats** per measure, while the denominator specifies the note duration that represents one beat.

Actually note symbols (𝅗𝅥) have no meaning if there's no clef at the beginning of each staff. Three kind of clefs are used in classical, commercial and contemporary Western music: G clef (𝄞), F clef (𝄢) and C clef (𝄡). Clefs assign specific pitches to lines on the staff, using a semicolon (:) to mark a reference line: the G clef marks G, the F clef marks F, and the C clef marks C. Clef changes might also occur to represent notes in a way that makes them easier to be read by a musician.

Next to a clef, flats or sharps may be found: these indicate the **key signature**, i.e. which sharps or flats are taken for granted in the whole

piece without the need to repeat them before each note. Other important indications on a score are:

- **tempo indications**: expressed as beats per minute (BPM), and refer to the speed of a music piece;
- **dynamics**: tell the player how loud they should play, by using terms such as *forte* for loud or *piano* for quiet;
- **expression markings**: another way of telling the speed of a piece, by using italian words such as *Grave*, *Presto*, *Affettuoso*, *Appassionato*, *Cantabile*, *Dolce*;
- **articulations**: tell the musician how to play the note like *slurs* (⌒), for smooth notes, or *staccato* (.), for short separated notes.
- **rests**: indicate to the musician when not to play and how long to remain silent, such as the symbol 𝄽

In digital formats, to refer to the occurrences of all of the musical notation elements presented in this Section, the term **musical events** is often used.

1.2 Sheet Music

This type of representation is the most easy for humans to read, while it's the most complex to analyse for a computer; for example, through optical music recognition (OMR[2]). Sheet music provides information related to systems of staves, clefs, time signatures, expression markings, pitch of notes, rests, accidentals, and dynamics. However, during a performance, the musician adjusts the suggested tempo and emphasises certain dynamics over others. As a consequence, sheet music is far away from an exhaustive

[2]This algorithm is the equivalent of optical character recognition (OCR), used for images of written text.

representation of music: it describe the notes to be played, leaving the performer with artistic freedom to create a new interpretation different from the previous ones [13].

1.3 Symbolic Representations

Symbolic representations describe music in a way that is easily readable by a machine. An example of an analog symbolic representation is the piano roll (see Section 1.3.1). Today, when referring to symbolic representations, we generally mean various digital formats. Among the most common are the **MusicXML** format (see Section 1.3.2), the **MEI** format (see Section 1.3.3), the **MIDI** format (see Section 1.3.4) and the ****kern** format (see Section 1.3.5).

1.3.1 The Piano Roll

This analogue musical representation developed between the late 19th and early 20th centuries, peaking in popularity in 1924, before being replaced by phonographic recordings. Known as Piano-Roll, it is a type of symbolic representation on punched rolls of paper, where each perforation encodes a note. These rolls were developed for self-playing keyboards that would play music when activated by a crank. This way, music recorded by great pianists such as Gustav Mahler, Edvard Grieg, Scott Joplin, and George Gershwin could be reproduced in private homes. The rolls were created using a special piano that allowed the paper to be perforated to mark which note was played, as well as its duration, dynamics, and use of pedals. Today, this type of representation is used in digital form, as a graphical representation derived from other formats like MIDI, MusicXML, or **kern, to quickly visualise semantic relationships. For example, compared to the original score fragment of the Fugue BWV 846 in C Major by J. S. Bach (Figure 1.1), in Figure 1.2 it is possible to immediately observe the entries of the different voices (soprano in green, alto in orange, tenor in yellow, and bass in azure).

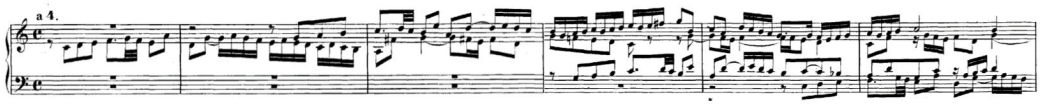


Figure 1.1: Fugue BWV 846 in C major by J. S. Bach, measures 1-6. Source: [15].

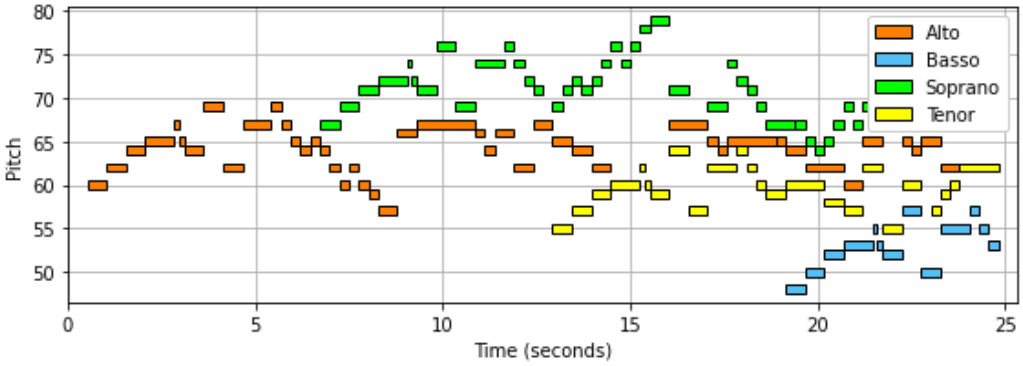
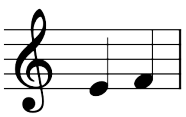


Figure 1.2: Piano-roll of the Fugue BWV 846 in C major by J. S. Bach.

1.3.2 MusicXML

While in the past musical scores were first handwritten by amanuensis and then printed by engraving metal plates, today various software applications are used to write, modify, and print music, such as MuseScore, Finale, or Sibelius. The universal format used to store and share musical files among these different software applications is called **MusicXML**. This format defines a set of rules for encoding musical scores, in order to make them both machine and human readable. As shown in Figure 1.3, the elements that make up the score can be easily identified in the file structure: the measure, indicated by the tag `<measure>`, and the individual notes, indicated by the tag `<note>`. For each of these notes, the pitch, `<pitch>`, and the note duration, `<duration>`, must be specified.

As suggested by David Huron, the inventor of the **kern format (see Section 1.3.5), MusicXML takes into account both the vertical and horizontal organisation of the music. Therefore, the file can have a `<score-partwise>` structure (as shown in Figure 1.3) or a `<score-timewise>` structure. In the first case, the horizontal organisation of the music is emphasised, with a clear separation between different parts, each assigned to its corresponding measures. In the second case, the vertical dimension of the music is highlighted, with the file structured by measures, and for each measure, the different parts are defined. If needed, it is possible to switch from one structure to another automatically using XSLT style sheets, parttime.xsl and timepart.xsl. In Figure 1.3 is shown how the quarter notes E and F are expressed in a MusicXML file. Each `<note>` element is defined by the following components:



```
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Melody</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <note>
        <pitch>
          <step>E</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>quarter</type>
      </note>
      <note>
        <pitch>
          <step>F</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>quarter</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Figure 1.3: Example of the structure of a MusicXML file and its graphical output.

- a `<pitch>` element: this specifies the exact note, including both the note itself (E or F) and the octave in which it occurs;
- a `<duration>` element, which defines the length of the note;
- a `<type>` element: in this case both notes are quarter notes

1.3.3 MEI

Music Encoding Initiative (MEI) is a markup language[3] similar to MusicXML, used to write music in digital format. It is an open-source, platform-independent standard, compatible with various operating systems.

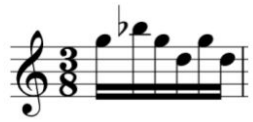
Compared to MusicXML, MEI allows for the creation of critical editions[4] and supports all types of Western musical notation, including mensural and neumatic notation. Furthermore, MEI can be used to mark musical analysis elements on the score, such as melodic and harmonic intervals, scale degrees, and harmonic notations. Due to the numerous features it offers, the MEI format is not currently supported by any popular music notation software (like Finale or Sibelius). Therefore, a reduced version, *MEI Basic*, with limited functionalities, has been created, which is supported by Musescore, but is not compatible with the MEI files generated in this project. However. online digital tools like https://mei-friend.mdw.ac.at/ or https://www.verovio.org/musicxml.html provide viewing and editing features for MEI files.

A MEI file consists of two types of elements: *Events* and *ControlEvents*. The *Events* define what should be played, i.e., the notes, chords, and rests in a piece. The *ControlEvents*, on the other hand, are those elements that describe how the music is to be performed, such as ties, dynamics, and tempo. In Figure 1.4, a segment of a MEI file is shown, where XML elements define the same notes as represented in the staff above. The `<beam>` element groups the eighth notes into a sextuplet. Each note includes the following attributes:

- **dur**: specifies the note's duration;
- **pname**: indicates the note's pitch name;
- **octave**: defines the note's octave;

[3] A *markup* language is a programming language with markers, called tags, that indicate to the program interpreting the document its logical and "hierarchical" structure [12]

[4] An edition of a musical work that contains notes about the notation, explanations of difficult parts, information about the writer, etc. [16]



```
<measure n="1">
  <staff n="1">
    <layer n="">
      <beam>
        <note dur="16" pname="g" oct="5" stem.dir="down" />
        <note dur="16" pname="b" oct="5" stem.dir="down" accid="f" />
        <note dur="16" pname="g" oct="5" stem.dir="down" />
        <note dur="16" pname="d" oct="5" stem.dir="down" />
        <note dur="16" pname="g" oct="5" stem.dir="down" />
        <note dur="16" pname="d" oct="5" stem.dir="down" />
      </beam>
    </layer>
  </staff>
  <tempo staff="1" tstamp="1.0" place="above" midi.bpm="130.0">
    <rend fontfam="smufl"> = 130</rend>
  </tempo>
</measure>
```

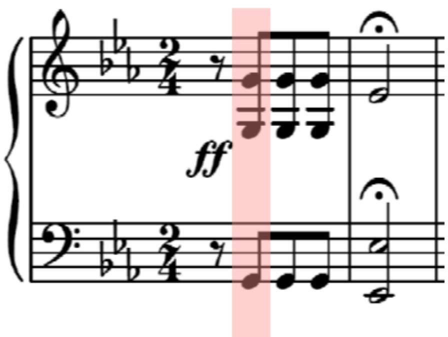
Figure 1.4: Fragment of an MEI file and its graphical output.

- **stem.dir**: determines the stem direction (up or down);
- **accid**: denotes any accidental (flat or sharp).

1.3.4 MIDI

Musical Instrument Digital Interface (MIDI) is the most widely used symbolic format for music representation. It was developed in the 1980s to facilitate communication between electronic musical instruments and computers. MIDI encodes information about the pitch, velocity, and duration of each note in a piece, as well as other performance details such as articulation, tempo, and instrument timbre. Unlike a digital audio file, MIDI does not store the actual sound of the music, but rather stores a series of control commands that can be used to synthesise the music when played back through a compatible instrument or software synthesiser. A MIDI file contains a series of messages note-on and note-off as shown in Figure 1.5. These messages tell to a computer when a note starts and ends.

Each of these messages also contains the following.



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

Figure 1.5: Comparison between a sheet music representation and a MIDI representation (in simplified table form) of the first twelve notes of Beethoven's Fifth Symphony. Source: [15].

- *note number*: an integer between 0 and 127 that encodes the pitch of the note, according to the equal-tempered system[5]. This format is therefore suitable only for representing Western classical, commercial and contemporary music.

- *key velocity*: an integer between 0 and 127, controlling the intensity of the sound. In the case of a note-on message, it controls the volume of the note, while in the note-off message, it controls the reduction in sound intensity, i.e. how the note fades away.

[5]This is the tuning system of Western classical music, in which all semitone intervals between consecutive notes are equal [10]

- *channel specification*: an integer between 0 and 15, indicating the channel assigned to a particular instrument. Each channel supports polyphony, meaning multiple simultaneous notes.

- *timestamp*: an integer that represents how many clock pulses or ticks[6] must be waited before executing a message. As shown in Figure 1.5, 60 ticks must be waited to execute the first message, corresponding to the crotchet rest. To express the simultaneity of the first three notes, the following two messages have a wait time of 0, meaning they will be executed simultaneously with the first message, once 60 ticks have passed to trigger the first message.

In contrast to a digital musical score, MIDI files enable the encoding and preservation of absolute-time information at a much finer and more flexible level. Although PPQN is a fixed number, absolute time information can be modified within the file by inserting a tempo message between two note-on messages. This facilitates the recording of pauses or changes in expression markings, such as *accelerandi*, *ritardandi*. Through these types of messages, it is also possible to calculate the number of quarter notes played in one minute, i.e., the measurement of time in BPM (Beats Per Minute). For example, if one quarter note corresponds to $6,000\,\mu$s, then the tempo is 100 BPM.

1.3.5 **kern

This symbolic format was originally created by David Huron in the 1980s as part of the Humdrum Toolkit. Its aim was to create a set of resources that are useful for computational music analysis. ****kern** is a musical notation

[6] A unit of temporal measurement determining the subdivision of time within a piece. In a MIDI file, each quarter note is subdivided into a predefined number of clock pulses or ticks. This number specifies the resolution of a MIDI file, called PPQN (Pulses Per Quarter Note), which is specified in the file's header. For example, if a MIDI file has a resolution of 480 PPQN, that means that there are 480 pulses per quarter note (1/4 of a beat).

format where the focus is on encoding the functional information underlying a musical score, rather than the orthographic and visual details found in printed versions. Among the numerous questions that can be addressed using this toolkit for analysing kern files, one might, for example, search for instances of a motif, calculate harmonic intervals between two parts, identify Italian, French, or German sixth chords, or determine how frequently augmented intervals are used within a piece.

In a **kern file, the staff is represented vertically. It is possible to have multiple columns, with each column representing a different voice and/or instrument (as seen in Figure 1.7, where the second row indicates which staff the voice is in). Within each column, as shown in Figure 1.7, the measures are separated by an "=" sign followed by the measure number. To aid in the analysis of the piece, each note is encoded with the appropriate accidental, even if this has already been specified in the key signature or has appeared earlier in the same measure. For example, in Figure 1.7, the C

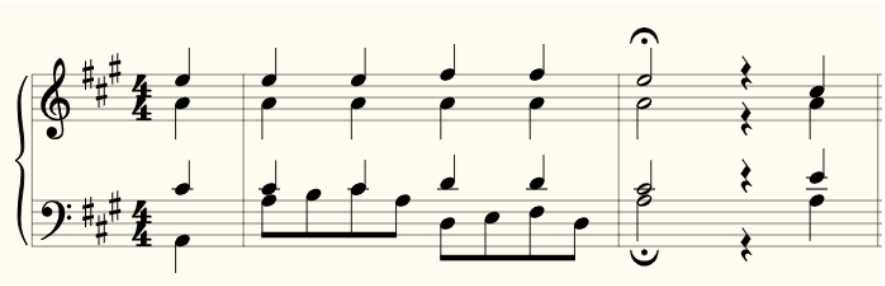


Figure 1.6: Sheet music: "*Nun danket alle Gott*", J.S. Bach, measures 1-2. Source: [17].

played by the tenor voice in the lower staff is always sharp due to the key signature (F♯, C♯ and G♯, indicating A major). However, in the kern file, each occurrence of this C is written explicitly as *4c♯*, where *4* denotes the note duration, *c* specifies the pitch, and ♯ indicates the sharp. Like MIDI files, only the notes of the equal-tempered system can be encoded, meaning this format is suitable only for representing Western music. Finally, this format provides a standardised way to encode bibliographic information through the

```
**kern      **kern      **kern      **kern
*staff2     *staff2     *staff1     *staff1
*cleff4     *cleff4     *clefG2     *clefG2
*k[f#c#g#]  *k[f#c#g#]  *k[f#c#g#]  *k[f#c#g#]
*M4/4       *M4/4       *M4/4       *M4/4
4AA         4c#         4a          4ee
=1          =1          =1          =1
8AL         4c#         4a          4ee
8B          .           .           .
8c#         4c#         4a          4ee
8AJ         .           .           .
8DL         4d          4a          4ff#
8E          .           .           .
8F#         4d          4a          4ff#
8DJ         .           .           .
=2          =2          =2          =2
2A;         2c#;        2a;         2ee;
4r          4r          4r          4r
4A          4e          4a          4cc#
=3          =3          =3          =3
```

Figure 1.7: Lines from a **kern file containing "*Nun danket alle Gott*" by J.S. Bach, measures 1-2. Source: [17].

so-called *reference record*. This is a type of global comment that facilitates computer access to bibliographic information. Figures 1.6 and 1.7 show the same excerpt from the chorale "*Nun danket alle Gott*" by J.S. Bach, both in sheet music notation and as a **kern file.

1.4 Comparison of the formats

To determine the optimal format for data standardisation and the creation of MEMUK, the table 1.1 outlines the advantages and disadvantages of these various formats. It evaluates key factors, such as the ability to encode performance data, machine and human readability, and compatibility with the Music21[7] Python library.

Since performance data is influenced by the interpretation of a particular musician, it is highly variable. For instance, MIDI is a digital format that

[7]Documentation is available at: `https://www.music21.org/music21docs/#`

can encode a musician's performance. However, converting a performance encoded in MIDI is a difficult and error-prone task because, during a performance, the duration of the notes is not very precise and is expressed in milliseconds. As a result, transcriptions of note durations may be incorrect. This problem can be avoided when the MIDI file is generated from a score rather than from a performance.

The Music21 library was, then, chosen for converting files to a common and standard format (see Chapter 4). For this reason, the target format needed to be compatible with the library.

Furthermore, selecting a format that is readable by both human beings and machines simplifies error checking during the conversion process, also allowing for quick manual verification. For example, sheet music can be visually compared to the lines of code in the MEI format to spot errors.

To summarise, based on the previous considerations, the goal was to find a formats that excludes performance data, is compatible with Music21, and is human-readable.

MEI and MusicXML stood out among the options because they both rely on XML structure. This structure is particularly useful for storing musical data as it is readable and editable by both humans and machines [13]. In addition it is highly compatible with the SPARQL Anything tool, that allows the creation of Knowledge Graphs (see Chapter 6).

Ultimately, MEI was selected over MusicXML because it provides a standardised structure and superior support for complex musical notations, including those used in Ancient music. This flexibility ensures that both the knowledge graph and the dataset can be easily extended and enriched in the future, even with Ancient Music or critical editions.

Format	MusicXML	MEI	**kern	MIDI
Representation type	music score	music score	symbolic	symbolic
Pros	Universal format to share files across different music notation software	Supports all western music notation types (even from the Middle Ages and the Renaissance), by encoding not only the visual aspects but also the semantic structure of these notation	Compatible with the Humdrum Toolkit for musical pieces' comparison (for Unix-based operating systems only)	Compatible with the different operative systems
Performance	No	No	No	Yes
Machine readable	Yes	Yes	Yes	Yes
Human readable	Yes	Yes	Yes	No
Compatible with music21 python library	Yes	Yes	Yes	Yes

Table 1.1: Comparison of the music formats MusicXML, MEI, kern and MIDI

Chapter 2

State of the art

Beyond the need for data diversification to reduce cultural bias, traditional datasets pose many other challenges. First of all, they are often focused on a particular aspect of music, such as metadata, audio files or the emotional content of music. Therefore, it becomes harder to gain a deeper understanding of music's structure and its cultural context. Furthermore, datasets often fail to capture the complex relationships between a musical piece and its audio or video representation, as well as its connections to other fields like ethnomusicology or philosophy.

A KG is one of the most suitable data structure to represent all these relationships with knowledge coming from different disciplines. The core component of KGs are RDF[1] triples. Each of them consists of three components:

1. **Subject**: the entity being described, for instance Violin Concerto No. 1;

2. **Predicate**: the relationship with the object or a property, like *composed by*;

3. **Object**: the value of a property or the related entity, such as Johann

[1]RDF stands for Resource Description Framework, a data model for the unique description of resources through URIs (Uniform Resource Identifiers) [12].

Sebastian Bach;

Thanks to this structure many different types of information can be linked together, such as metadata, melodic and harmonic annotations, lyrics and the cultural context of a musical piece, allowing for an interconnected representation of musical data. The structure of a knowledge graph is defined by an **ontology** which specifies all the possible types of entities, their attributes and how entities relate to each other. Ontologies are formalised models used to represent a certain domain of knowledge, such as music. They enable automatic reasoning, thanks to the use of languages based on logic.

In the last two decades, significant contributions have been made to the creation of musical ontologies. Most of them have a specific focus, much like traditional datasets do. Many of them have been developed as standalone projects "with little or no alignment to other relevant ontologies within the same domain"[9]. A comprehensive list of these different ontologies is reported in Figure 2.1 on a timeline. These ontologies can be grouped into six main categories:

1. **Metadata**: the Music Ontology [18], the Listening Habits and Music Tastes ontology [19]and the DOREMUS Ontology [20]. These ontologies have been developed not only to represent high-level music metadata information, such as composer, title and other editorial information, but also contextual and historical information;

2. **Theoretical concepts**: the Tonality Ontology [21] and the Temperament Ontology [22]. These ontologies represent musical concepts related to music theory, like tonalities and temperamets;

3. **Music notation**: the Music Theory Ontology [23], the Music Notation Ontology [24], the Music Score Ontology [25]and the MIDI Linked Data Cloud [26]. These ontologies describe musical notation, including both score-based formats and symbolic representations.

4. **Recording studio environment**: the Studio Ontology [27], the Audio Effects Ontology [28]. These ontologies define elements from music

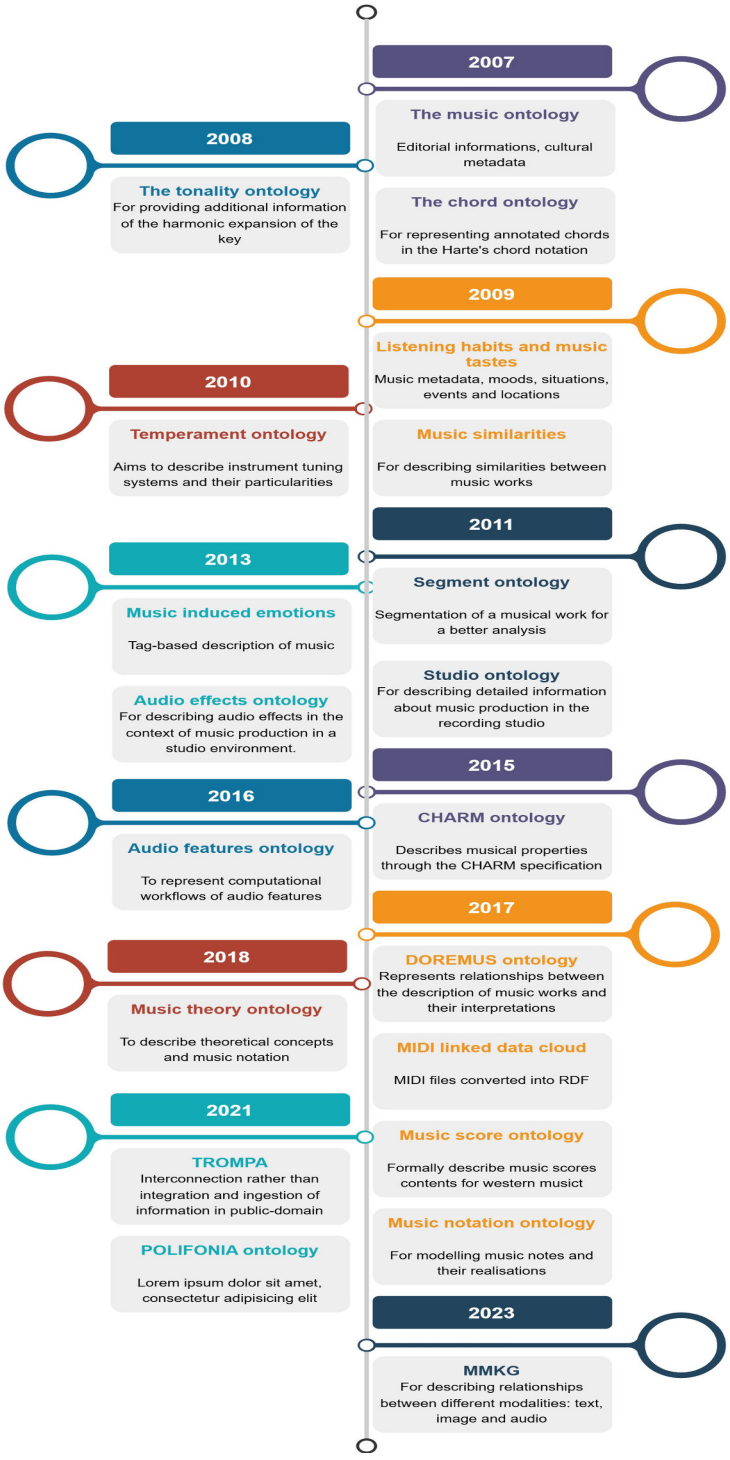


Figure 2.1: Timeline of the ontologies.

production and recording studios.

5. **Music recommendation**: the Music-Induced Emotions ontology [29], the Music Similarities ontology [30] and MMKG [31]. These ontologies were created with the goal of enhancing the accuracy and efficiency of music recommendation systems.

6. **Others**:

   - the TROMPA ontology [32] aims to represent the interconnection between musical information already in the public domain, rather than offerring integration and ingestion of these data.
   - the Segment Ontology [33], the Audio Features ontology [34] and the Chord ontology [35], which all have a very specific focus;
   - the CHARM Ontology [36], which describes musical structure through the CHARM specification.

None of these ontologies proposes a complete framework that can cover at the same time data about musical features, instruments, emotions and performance. Recently the Polifonia Ontology Network (PON) has been released. PON aligns most of the existing ontologies in the music domain and extends significantly their coverage.

2.1 The polifonia ontology network

The Polifonia Ontology Network (PON) is a modular ontological ecosystem designed to support cultural, contextual, and music-related queries. Unlike other ontologies, PON is designed as a network of interconnected modules that work cohesively to represent both the cultural context and the technical details of music. This ontology is made up of 15 modules that are organised both thematically (colours, horizontal view) and hierarchically to highlight their dependencies (vertical view), as shown in Figure 2.2. Thanks to this modular structure, the PON ontology is easier to maintain and extend. In

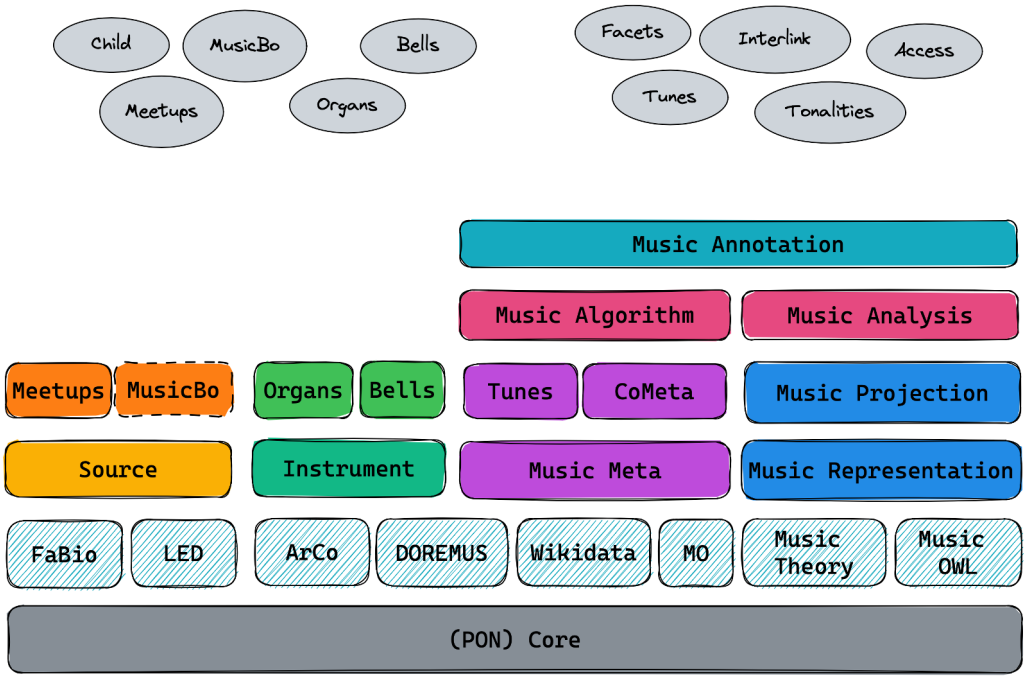


Figure 2.2: Polifonia Ontology Network architecture [9].

addition, modularity ensures flexibility, as users can select only the necessary modules for specific applications. At the base of this ontology lies the **core** module. The interoperability across PON is provided by four foundational models: **Source**, **Instrument**, **Music Meta**, and **Music Representation**. By harmonising all the different information associated with a musical piece, users can explore the derived knowledge graphs with interdisciplinary and cross-domain queries.

Part of the PON ontology is also the ontology module developed to model chord music annotations. This module was implemented as part of the Choco project [6], which serves as a primary source of guidance for the implementation of the dataset integration process used in this thesis. Choco proposes the creation of a large-scale dataset that semantically integrates harmonic data from 18 different sources using heterogeneous representations and formats, such as Harte, Lead-sheet, Roman Numerals, and ABC. These chord

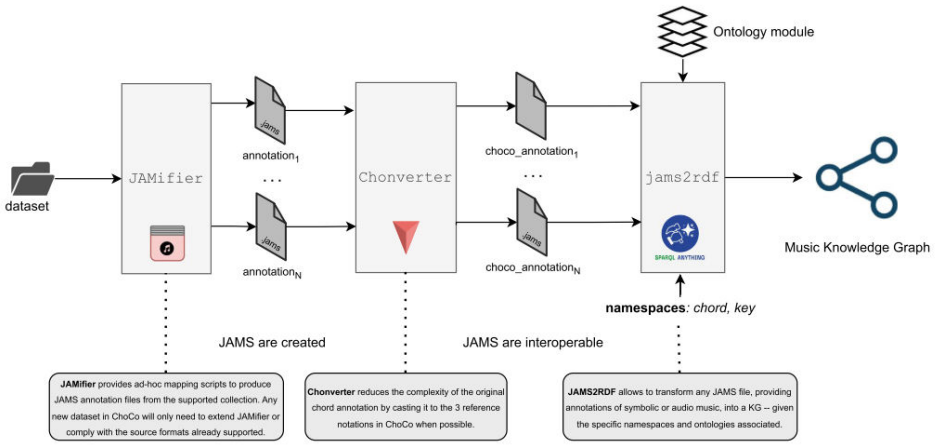


Figure 2.3: Integration workflow used in Choco [6].

annotations are then converted and aligned into a common format: JAMS. From this large-scale dataset a KG was generated, with the aim of answering questions regarding who the composer/performer of a musical object is, what the key of a composition/performance is, and what the value of an annotation is. Figure 2.3 shows the integration workflow used in Choco: first, the JAMifier ingests chord collections (where metadata and music annotations follow collection-specific conventions and formats) to generate a JAMS dataset [6]. This approach has multiple levels of integration. First, all metadata are systematically reorganised. Meanwhile, for each track or score, music annotations (such as chord progressions) are encoded and stored into individual JAMS files. The Chonverter achieves notational interoperability among collections by converting the original annotations to the same notational families [6]. Finally, jams2rdf uses notation-specific ontologies to produce RDF triples, thus creating a Music Knowledge Graph.

This thesis adopts a similar approach for data integration, but with a focus on musical scores and symbolic music representations, like MIDI, *kern and MusicXML. Instead of the JAMS format, which was used in Choco, this work uses the MEI format (see Section 1.3.3). A similar approach to the JAMifier has been implemented: for each new dataset a method to align

title and composer's metadata needs to be implemented. Each original file is then transformed into a MEI file using a converter as described in Chapter 4. This converter adopts a modular approach to facilitate future suppor of a wider variety of music formats.

A KG is then generated relying on the music projection module for describing music score elements, instead of the JAMS module used in Choco to represent chord annotations. This music projection module helps generate a KG that provides a robust framework for describing music score elements and to answer not only questions concerning who the composer of a musical piece is or for which instrument the score was written, but also questions related to finding similar pitches and notes in a score.

Chapter 3

Data collection

The effectiveness of any MIR application is heavily dependent on the quality and diversity of the used datasets. This chapter outlines the process of selecting and analysing datasets necessary for constructing a knowledge graph focused on melodic annotations. The process begins by exploring publicly available datasets. Furthermore, a systematic approach is for filtering and classifying datasets containing relevant melodic data. Python scripts were employed as an automated way to identify datasets containing specific keywords, while manual evaluation ensured the relevance of the data. Additional factors, including licensing, musical genres, and annotation formats, were also considered to ensure the datasets contained a diverse range of musical genres and annotation types.

3.1 Selection of datasets containing melodic annotations

ISMIR, a non-profit organisation and leading authority in the field of Music Information Retrieval, provides a comprehensive YAML-formatted list of datasets to be used for MIR tasks, available at `https://github.com/ismir/mir-datasets/blob/master/mir-datasets.yaml` and manteined by

Alexander Lerch[1].

Due to the limited time available for the development of this thesis and the extensive list of over 200 datasets, an initial automated selection was carried out using a Python script to remove all entries that did not contain the keywords of melodic annotations format. The most suitable formats for melodic information are the digital symbolic formats explained in Chapter 1, as they encode all music events, including notes. The keywords used were the most common digital symbolic file extesions: "MusicXML," "xml," "MIDI," "MEI," "Kern,". To include more results also the keyword "melod" was used, in order to look for words like melody or melodic.

Thanks to his process 38 datasets have been identified. Figures 3.1 and 3.2 show the list of datasets; for each of them it is specified whether they contain also audio files, what contents and metadata are and the URL from which the dataset could be downloaded.

To isolate entries featuring melodic annotations, data were manually collected regarding the exclusive presence of melodic annotations for a particular instrument. At the same time, data related to licences, musical genres, and number of annotations were collected in each dataset. As a result of this search, four datasets had to be excluded outright because they were no longer available online. Therefore, the produced table in Figures 3.3 and 3.4 contains ten additional columns:

- **available**: whether the dataset is actually available online to be downloaded or not;
- **genre**: a list of one or more musical genres that are represented in the dataset. To unify the notation the list of music genres available at `https://en.wikipedia.org/wiki/List_of_music_genres_and_styles` was used as a reference;
- **licence**: this field is particularly important in order to know under

[1] Alexander Lerch is the leader of the Music Informatics Group at the Georgia Institute of Technology, where he works on machine learning and intelligent signal processing solutions for music.

	Dataset	Audio	Contents	Metadata	Url
1	AAM	yes	3,000 music tracks (with single instrument multitracks)	onsets, pitches, instruments, melody instrument, keys, chords, tempos, beats, segments	https://zenodo.org/record/5794629
2	ADC2004	yes	20 excerpts	p, r, e, d, o, m, i, n, a, n, t, , p, i, t, c, h	http://labrosa.ee.columbia.edu/projects/melody
3	ASAP	yes (see MAESTRO)	1068 MIDI performances, 520 audio performances, 222 scores	aligned MIDI/audio performances and MIDI/XML scores, beats, downbeats, time signatures, key signatures	https://github.com/fosfrancesco/asap-dataset
4	ATEPP		1742 performances (~1000 hours) by 49 pianists and covers 1580 movements by 25 composers	symbolic music MIDI, musicXML, classification tasks, expressive piano performances	https://zenodo.org/record/6564406#.Y5QnkezP3OQ
5	AccoMontage2	no		s, o, n, g, , h, a, r, m, o, n, i, z, a, t, i, o, n, , a, n, d, , a, c, c, o, m, p, a, n, i, m, e, n, t, , a, r, r, a, n, g, e, m, e, n, t, , b, a, s, e, d, , o, n, , a, , l, e, a, d, , m, e, l, o, d, y	https://github.com/billyblu2000/accomontage2
6	CSD	yes	48 recordings	f0, MIDI	https://zenodo.org/record/2649950
7	Chopin22	yes	44 recordings	aligned MIDI	http://iwk.mdw.ac.at/goebl/mp3.html
8	ExpandedGrooveMD	rendered	45537 midi/audio pairs	drummer/session id, drum timing, kit name	https://g.co/magenta/e-gmd
9	GrooveMD	rendered	1150 MIDI recordings	drummer/session id, drum timing	https://magenta.tensorflow.org/datasets/groove
10	GuitarSet	yes	360 guitar excerpts (30s) with hexaphonic audio	midi, pitch, beat, chords	https://guitarset.weebly.com/
11	HookTheory	yes	50 hours of aligned melody and harmony annotations	aligned melody and harmony annotations	https://github.com/chrisdonahue/sheetsage
12	JGDB	yes	random generated excerpts	multitrack, MIDI	https://ccrma.stanford.edu/~jga/ismir2010/ismir2010.html
13	JKU-ScoFo	yes	16 recordings	audio, MIDI	http://www.cp.jku.at/resources/2019_RLScoFo_TISMIR
14	LMD	no	176581 MIDI files	MIDI, {"tempo": "http://www.tagtraum.com/download/schreiber_tempo_cnn_ismir2018.zip"}, {"key": "http://www.tagtraum.com/download/lmd-key.zip"}	https://colinraffel.com/projects/lmd/
15	LabROSA:APT	yes	29 piano excerpts	M, I, D, I	http://labrosa.ee.columbia.edu/projects/piano/
16	LabROSA:MIDI	yes	4 songs	audio, MIDI	http://labrosa.ee.columbia.edu/sounds/music/
17	MAESTRO	yes	172 hours of piano	audio aligned midi, velocity, sustain	https://magenta.tensorflow.org/datasets/maestro
18	MARG-AMT	yes	30 melodies	MIDI pitch, onset/offset times	http://marg.snu.ac.kr/?page_id=767
19	MAST	no	1018 performances	v, o, c, a, l, , p, e, r, f, o, r, m, a, n, c, e, , a, s, s, e, s, s, m, e, n, t	https://github.com/barisbozkurt/MASTmelody_dataset
20	MOODetector:Multi-Modal	yes	903 excerpts (30s)	lyrics, MIDI, mood	https://github.com/johnglover/modal
21	MSMD	no	497 pieces	piano notes/chords/pieces, synthetic audio, aligned MIDI, aligned sheet music images, OMR	https://zenodo.org/record/2597505/
22	MTC	partially	18000 melodies	phrases, key, meter	http://www.liederenbank.nl/mtc/
23	MedleyDB	yes	122 songs	multitrack, genre, melody f0, instrument activation	http://medleydb.weebly.com
24	MeloSol	no	783 melodies	melody, monophonic, symbolic, kern, key	https://davidjohnbaker1.github.io/melosol/

Figure 3.1: First segment of the dataset filtered using keywords such as "MusicXML," "xml," "MIDI," "MEI," "Kern," or "melod" with a Python script.

	Dataset	Audio	Contents	Metadata	Url
25	NES-MDB	on request	5000 songs	multi-track MIDI, aligned audio	https://github.com/chrisdonahue/nesmdb
26	Phenicx-Anechoic	yes	4 pieces	multi-track audio (orchestral music), aligned MIDI	http://mtg.upf.edu/download/datasets/phenicx-anechoic
27	QBT-Extended	MIDI	3365 queries/51 songs	t, a, p, s	https://ccrma.stanford.edu/groups/qbtextended/index.html
28	RWC	yes	115 songs/50 classical/100 songs	lyrics, 10 genre, 50 instruments, {"chords": "https://github.com/tmc323/Chord-Annotations"}, {"structure": "http://musicdata.gforge.inria.fr/structureAnnotation.html"}, {"aligned MIDI": "https://staff.aist.go.jp/m.goto/RWC-MDB/AIST-Annotation/SyncRWC/"}	http://staff.aist.go.jp/m.goto/RWC-MDB/
29	RockCorpus	no	200 songs	chords, melody, bars	http://rockcorpus.midside.com
30	SLAKH	yes	2100 mixes	MIDI, synthesized audio (tracks + mix)	http://www.slakh.com/
31	SMD	yes	50 recordings	audio, aligned MIDI	http://www.mpi-inf.mpg.de/resources/SMD/SMD_MIDI-Audio-Piano-Music.html
32	SWD	yes	24 songs, 9 performances	lyrics, scores (image, symbolic, MIDI), audio, measures, chords, local keys, global keys, structure	https://zenodo.org/record/4122060#.YPkxv-gzaUl
33	Schenker	no	41 pieces	MusicXML, Schenker analysis	http://www.cs.rhodes.edu/~kirlinp/diss.html
34	SymphonyMIDI	no	46187 MIDI scores	MIDI, symphonic	https://symphonynet.github.io/
35	TFD	yes	30 excerpts	audio, aligned MIDI	https://www.kaggle.com/jbraga/traditional-flute-dataset
36	bach10	yes	10 chorales	a, l, i, g, n, e, d, , m, u, l, t, i, t, r, a, c, k, , M, I, D, I	http://www2.ece.rochester.edu/projects/air/resource.html
37	corpusCOFLA	no	1800 flamenco recordings	editorial, predominant melody	http://www.cofla-project.com/?page_id=170
38	mirex05Train	yes	13 excerpts	p, r, e, d, o, m, i, n, a, n, t, , p, i, t, c, h	http://labrosa.ee.columbia.edu/projects/melody/

Figure 3.2: Second segment of the dataset filtered using keywords such as "MusicXML," "xml," "MIDI," "MEI," "Kern," or "melod" with a Python script.

which conditions the dataset can be used. Refer to Appendix A for an in-depth-analysis of the meaning of the different kinds of licence that have been found in this list of 34 datasets;

- **annotation format**: specifies what's the format of the the files contained in the dataset, like MIDI or MusicXML;
- **number of annotations**;
- **texture**: whether the files contain only a single instrument, **melody**, or multiple instruments and voices, **polyphony**;
- **Instruments**: the types of instruments represented in the dataset. If the list of instruments is too long, it is summarised with the keyword **several**.
- **piece**: whether the file was generated from the performance of a musician or not.

3.2 Dataset classification

An examination was conducted in a Python notebook[2] on the previously mentioned 34 datasets (see Figures 3.3, 3.4), focusing on the following criteria:

- licensing type
- music genres
- annotation formats, such as MIDI, musicXML, kern, or others
- whether the annotations originate from performances or not
- whether the annotations are melodic or harmonic

	Dataset	Available	Audio	Contents	Metadata	Url	Paper	Genre	Licence	Annotation format	Number of annotations	Texture	Instruments	Piece	Performance
1	AAM	yes	yes	3,000 music tracks (with single instrument multitracks)	onsets, pitches, instruments, melody instrument, keys, chords, tempos, beats, segments	https://zenodo.org/record/5794629	https://asmp-eurasipjournals.springeropen.com/counter/pdf/10.1186/s13636-023-00278-7.pdf	None	CC BY	MIDI	3000	poliphony, melody	several	entire	no
2	ADC2004	yes	yes	20 excerpts	p, r, e, d, o, m, i, n, a, n, t, , p, i, t, c, h	http://labrosa.ee.columbia.edu/projects/melody	https://www.ee.columbia.edu/~dpwe/pubs/ismir05-melody.pdf	Rock, R&B & Soul, Pop, Jazz, Classical	CC BY-NC-SA	MIDI, recording	60	poliphony	several	entire	yes
3	ASAP	yes	yes	1068 MIDI performances, 520 audio performances, 222 scores	aligned MIDI/audio performances and MIDI/XML scores, beats, downbeats, time signatures, key signatures	https://github.com/fosfrancesco/asap-dataset	https://archives.ismir.net/ismir2020/paper/000127.pdf	Classical	CC BY-NC-SA	MIDI, musicXML	1068	poliphony	piano	entire	yes
4	ATEPP	yes	no	1742 performances (~1000 hours) by 49 pianists and covers 1580 movements by 25 composers	symbolic music MIDI, musicXML, classification tasks, expressive piano performances	https://zenodo.org/records/7182820	https://archives.ismir.net/ismir2022/paper/000053.pdf	Classical	CC BY	MIDI, musicXML	11674	poliphony	piano	entire	yes
5	AccoMontage2	yes	no		song, harmonization, and accompaniment, arrangement, based, on a lead, melody	https://github.com/billyblu2000/accomontage2	https://arxiv.org/pdf/2209.00353	Pop, R&B & Soul	CC BY	MIDI	64524	poliphony	singer, several	extract	no
6	CSD	yes	yes	48 recordings	f0, MIDI	https://zenodo.org/record/2649950	https://zenodo.org/records/2649950/files/Cuesta561.pdf?download=1	Folk, Classical	CC BY	MIDI	12	poliphony	choir	entire	yes
7	Chopin22	yes	yes	44 recordings	aligned MIDI	https://repo.mdw.ac.at/projects/IWK/the_vienna_4x22_piano_corpus/index.html	http://iwk.mdw.ac.at/goebl/papers/Goebl1999-Diplomarbeit.pdf	Classical	CC BY	MIDI	112	poliphony	piano	entire	yes
8	ExpandedGrooveMD	yes	rendered	45537 midi/audio pairs	drummer/session id, drum timing, kit name	https://g.co/magenta/e-gmd	https://arxiv.org/pdf/2004.00188	Funk, Jazz, R&B & Soul, African, Hip Hop, Electronic	CC BY	MIDI	45537	melody	drums	none	yes
9	GrooveMD	yes	rendered	1150 MIDI recordings	drummer/session id, drum timing	https://magenta.tensorflow.org/datasets/groove	https://arxiv.org/pdf/1905.06118	Funk, Jazz, R&B & Soul, African, Hip Hop, Electronic	CC BY	MIDI	1150	melody	drums	none	yes
10	GuitarSet	yes	yes	360 guitar excerpts (30s) with hexaphonic audio	midi, pitch, beat, chords	https://guitarset.weebly.com/	https://guitarset.weebly.com/uploads/1/2/1/6/121620128/xi_ismir_2018.pdf	Rock, Folk, Jazz, R&B & Soul	CC BY	MIDI	2160	poliphony	guitar	extract	yes
11	HookTheory		yes	50 hours of aligned melody and harmony annotations	aligned melody and harmony annotations	https://github.com/chrisdonahue/sheetsage	https://arxiv.org/pdf/2212.01884	Pop, Rock, Jazz, Classical	CC BY	recording	22000	poliphony	singer, several	extract	yes
12	JGDB		yes	random generated excerpts	multitrack, MIDI	https://ccrma.stanford.edu/~jga/ismir2010/ismir2010.html	https://ccrma.stanford.edu/~jga/ismir2010/gansemanISMIR2010.pdf	None	CC BY-NC-SA	MIDI	380	poliphony	several	extract	no
13	JKU-ScoFo	yes	yes	16 recordings	audio, MIDI	http://www.cp.jku.at/resources/2019_RLScoFo_TISMIR	https://transactions.ismir.net/articles/31/files/submission/proof/31-1-775-1-10-20191120.pdf	Classical	CC BY	MIDI	650	poliphony	piano	entire	no
14	LMD	yes	no	176581 MIDI files	MIDI, tempo: http://www.tagtraum.com/download/schreiber_tempo_cnn_ismir2018.zip, key: http://www.tagtraum.com/download/lmd-key.zip	https://colinraffel.com/projects/lmd/	http://colinraffel.com/publications/thesis.pdf	Rock, Pop, Alternative, Jazz, Hip Hop, Soundtrack, R&B & Soul, Electronic, Country	CC BY	MIDI	176581	poliphony	several	entire	no
15	LabROSA:APT	yes	yes	29 piano excerpts	MIDI	http://labrosa.ee.columbia.edu/projects/piano/	http://www.ee.columbia.edu/~graham/papers/piano.pdf	Classical	CC BY-SA	MIDI	103	poliphony	piano	entire	yes
16	LabROSA:MIDI	yes	yes	4 songs	audio, MIDI	http://labrosa.ee.columbia.edu/sounds/music/	http://www.ee.columbia.edu/~dpwe/pubs/ismir03-midi.pdf	Rock, Electronic, Pop	CC BY-NC	MIDI	40	poliphony	singer, several	entire	yes
17	MAESTRO	yes	yes	172 hours of piano	audio aligned midi, velocity, sustain	https://magenta.tensorflow.org/datasets/maestro	https://arxiv.org/pdf/1810.12247	Classical	CC BY-NC-SA	MIDI	1184	poliphony	piano	entire	yes
18	MAST	yes	no	1018 performances	vocal performance assessment	https://github.com/barisbozkurt/MASTmelody_dataset	http://www2.denizyuret.com/bib/bozkurt/bozkurt2017/cmmr2017_BozkurtBaysalYuret.pdf	Classical	CC BY-NC-SA	recording	3617	poliphony, melody	singing, piano	entire	yes
19	MOODetector:Multi-Modal	yes	yes	903 excerpts (30s)	lyrics, MIDI, mood	https://github.com/johnglover/modal	https://mural.maynoothuniversity.ie/4204/1/JT_Real-Time_Detection.pdf	NULL	GNU GPL	MIDI	2048	NULL	NULL	NULL	NULL

Figure 3.3: First part of the selected datasets' list, including information on related papers, musical genres, licenses, annotation types, instruments, and texture for each dataset.

	Dataset	Available	Audio	Contents	Metadata	Url	Paper	Genre	Licence	Annotation format	Number of annotations	Texture	Instruments	Piece	Performance
20	MSMD	yes	no	479 pieces	piano notes/chords/pieces, synthetic audio, aligned MIDI, aligned sheet music images, OMR	https://zenodo.org/record/2597505/	https://transactions.ismir.net/articles/12/files/submission/proof/12-1-302-1-10-20180904.pdf	Classical	CC BY	MIDI	479	poliphony	piano	entire	no
21	MedleyDB	yes	yes	122 songs	multitrack, genre, melody f0, instrument activation	http://medleydb.weebly.com	NULL	Songwriter, Classical, Rock, Folk, Fusion, Jazz, Pop, Rap	CC BY	recording	196	poliphony	several	entire	yes
22	MTC-LC-1.0	yes	no	18000 melodies	phrases, key, meter	https://www.liederenbank.nl/n	https://www.liederenbank.nl/n	Folk	CC BY NC SA	kern, MIDI	4830	melody	piano	extract	no
23	MeloSol	yes	no	783 melodies	melody, monophonic, symbolic, kern, key	https://davidjohnbaker1.github.io/melosol/	https://emusicology.org/index.php/EMR/article/download/7645/6268/28973	Classical	CC BY-NC	kern	783	melody	singer	entire	no
24	NES-MDB	yes	no	5000 songs	multi-track MIDI, aligned audio	https://github.com/chrisdonahue/nesmdb	https://arxiv.org/pdf/1806.04278	Electronic	MIT	MIDI	5278	poliphony	electronic	entire	no
25	Phenicx-Anechoic	yes	yes	4 pieces	multi-track audio (orchestral music), aligned MIDI	http://mtg.upf.edu/download/datasets/phenicx-anechoic	https://onlinelibrary.wiley.com/doi/pdf/10.1155/2016/8363507	Classical	CC BY-NC-SA	MIDI	4	poliphony	several	entire	yes
26	QBT-Extended	yes	no	3365 queries/51 songs	taps	https://ccrma.stanford.edu/groups/qbtextended/index.html	https://zenodo.org/records/1415756/files/KaneshiroKHOBS13.pdf?download=1	Pop	CC BY-NC-SA	MIDI	51	poliphony	singer, several	extract	no
27	RWC	yes	yes	115 songs/50 classical/100 songs	lyrics, 10 genre, 50 instruments, chords: https://github.com/tmc323/Chord-Annotations, structure: http://musicdata.gforge.inria.fr/structureAnnotation.html, aligned MIDI: https://staff.aist.go.jp/m.goto/RWC-MDB/AIST-Annotation/SyncRWC/	http://staff.aist.go.jp/m.goto/RWC-MDB/		Popular, Classical, Jazz	authorization needed	MIDI					
28	RockCorpus	yes	no	200 songs	chords, melody, bars	http://rockcorpus.midside.com	https://rockcorpus.midside.com/2011_paper/declercq_temperley_2011.pdf	Rock	CC BY	Roman numbers	100	poliphony	singer, several	entire	yes
29	SLAKH	yes	yes	2100 mixes	MIDI, synthesized audio (tracks + mix)	http://www.slakh.com/	https://arxiv.org/pdf/1909.08494	Rock, Pop, Alternative, Jazz, Hip Hop, Soundtrack, R&B & Soul, Electronic, Country	CC BY	MIDI	2100	poliphony	Piano, Bass, Guitar, Drums, Strings, Synth Pad, Reed, Brass, Organ, Pipe, Synth Lead, Chromatic Percussion	extract	no
30	SMD	yes	yes	50 recordings	audio, aligned MIDI	http://www.mpi-inf.mpg.de/resources/SMD/SMD_MIDI-Audio-Piano-Music.html	https://resources.mpi-inf.mpg.de/SMD/2011_MuellerKonzBoglerArifi_SaarlandMusicData_ISMIR-LateBreaking.pdf	Classical	CC BY-NC-SA	MIDI	200	poliphony	piano	entire	yes
31	SWD	yes	yes	24 songs, 9 performances	lyrics, scores (image, symbolic, MIDI), audio, measures, chords, local keys, global keys, structure	https://zenodo.org/records/10839767	https://dl.acm.org/doi/10.1145/3429743	Classical	CC BY	MIDI	24	poliphony	piano, singer	entire	yes
32	Schenker	yes	no	41 pieces	MusicXML, Schenker analysis	http://www.cs.rhodes.edu/~kirlinp/diss.html	https://dl.acm.org/doi/pdf/10.1145/3429743	Classical	CC BY-NC	musicXML	41	poliphony	piano, singer	extract	no
33	SymphonyMIDI	yes	no	46187 MIDI scores	MIDI, symphonic	https://symphonynet.github.io/	https://arxiv.org/pdf/2205.05448	Classical	MIT	MIDI	46359	poliphony	several	entire	yes
34	TFD	yes	yes	30 excerpts	audio, aligned MIDI	https://www.kaggle.com/jbraga/traditional-flute-dataset	NULL	Classical	CC BY-NC-SA	MIDI	30	melody	flute	extract	yes

Figure 3.4: Second part of selected datasets' list, including information on related papers, musical genres, licenses, annotation types, instruments, and texture for each dataset.

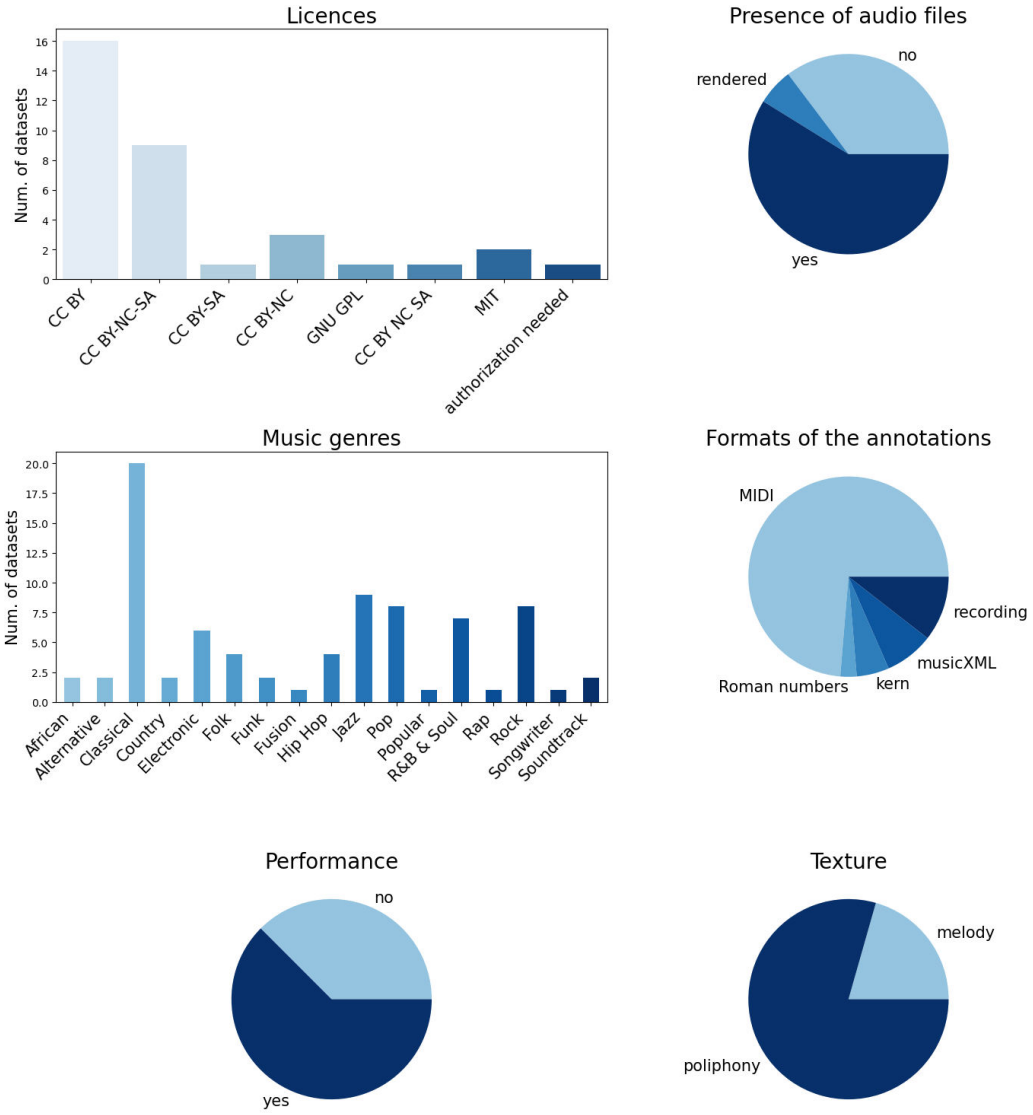


Figure 3.5: Charts illustrating the selected datasets: from left to right, the distribution of licenses, audio track inclusion, variety in music genres, categories of melodic annotations, performance status, and texture type.

The results of this analysis are shown in Figure 3.5, showing that Creative

[2] https://github.com/esthy13/tesi-mir/blob/main/Dataset_Selection/datasets_analisis.ipynb

Commons licences, particularly CC BY, are the most commonly used ones, allowing for the modification and reuse of datasets for both non-commercial and commercial purposes, provided that the original creators are acknowledged. In addition, the Real World Computing Music Database, composed of 315 performance recordings along with related MIDI files from various genres, requires specific authorisation for use. As a result, this dataset was also excluded from the analysis. From the charts in Figure 3.5 we can besides notice that the majority of datasets contain melodic annotations, with MIDI emerging as the most prevalent annotation format, and Classical music as the most represented genre. In terms of texture, the datasets are primarily polyphonic, which simplifies the process of selecting the ones which feature exclusively melodic annotations. Additionally, most of these datasets are derived from performance-based annotations and often include corresponding audio files.

3.3 Selected datasets

It was crucial to select datasets containing melodic annotations for the construction of the MEMUK knowledge graph. The selection process prioritised datasets that provided a diverse range of musical genres and various annotation types. Diversity is an important elements, as, for instance, "using the "most famous" or "most popular" songs to represent a wider tradition risks eliminating an exploration of that tradition's underlying diversity of practice" [37]. Including multiple annotation formats was particularly important as it would support the development of a file converter capable of handling different file formats. In addition, it helps showcase the functionalities and the effectiveness of the developed conversion software.

With these criteria in mind, the following datasets were initially identified as potential candidates for inclusion:

- ASAPP [38]
- ATEPP [39]

- MTC-LC-1.0 [40]
- Melosol [41]
- TFD[3]

Upon further analysis of these datasets, it became evident that the TFD dataset was not a suitable candidate. This dataset consisted of few performance-derived data, where each MIDI file represented an extract of a larger musical piece, and therefore did not accurately reflect the entire composition. In addition, performance data are not taken into account as explained in Section 1.4. Furthermore, the MTC-LC-1.0 dataset, which contains fragments of Dutch folk songs, offered limited variety in terms of musical genres, thus failing to meet the diversity requirement.

In order to create a dataset that contains a broader range of genres, beyond just classical music, it was established to include the LMD MIDI [42] dataset. Despite including files with multiple instruments, this dataset was considered suitable for the project as individual tracks for each instrument can be easily separated from the original multi-track files. Actually, the full LMD MIDI dataset was not used, as it contains a few thousands of invalid MIDI files (i.e. files not respecting the MIDI standard file specification). Instead, the Clean MIDI subset[4] was used. The filenames used in this dataset indicate the artist and title of each file.

[3]Available at: `https://www.kaggle.com/jbraga/traditional-flute-dataset`
[4]Available for download at: `https://colinraffel.com/projects/lmd/`

Chapter 4

Data's conversion process

A dataset might contain from a few to thousands of files. Generally, the larger the number of files is, the better the dataset may be, as a larger dataset often provides more diversity and variability, which can improve the accuracy and robustness of the model. The chosen datasets, Melosol, ATEPP-1.1 and Clean LMD contain, respectively, 783, 11,674 and 17,256 files. As manual conversion was not a viable option, the development of a conversion software in Python was needed.

Melosol contains songs for teaching and learning how to sing, composed only by R. Berkowitz[1]. Even if ATEPP-1.1 contains MIDI files from virtuoso pianists performances, it was chosen as it also contains MusicXML files. Furthermore, the Clean LMD dataset was chosen to assure diversity to the integrated melodic dataset available at: `https://github.com/esthy13/memuk/tree/main/melody/partitions`, because it includes many different music genres such as Rock, Pop, Alternative, Jazz, Hip Hop, Soundtrack, R&B & Soul, Electronic, and Country.

This work proposes an innovative approach not only by diversifying the music genres and composers represented in the datasets, but also by standardising all files into a single, common format: MEI. The integration of these datasets is facilitated by the development of the conversion tool presented in

[1]Ralph Berkowitz was an American composer, classical musician, and painter. [43]

this chapter. The aim is to enhance the reproducibility of experiments using these three datasets and to encourage future expansion of the integrated dataset by simplifying the conversion of additional files into the standardised MEI format.

This chapter outlines the system requirements needed to run the software and provides an overview of its usage describing the conversion process in Figure 4.1. Finally, the several challenges faced while producing this software are discussed.

The main library used in this project is **Music21**[2], a series of powerful tools for computing-aided musicology. Music21 does not natively support exporting to the MEI format. This limitation is addressed by the Converter21 module enhances the functionality of the Music21 library, enabling conversions not only to MEI but also to kern files.

4.1 System requirements

In order to run the software, ensure that Python is installed on your system. Moreover, pip is essential for installing the necessary project modules: Music21, PrettyMidi[3], and Converter21[4]. The program's code is available in the GitHub repository at `https://github.com/esthy13/tesi-mir/tree/main/melody`. After setup is complete, the program can be started from the command line using this command:

```
python main.py /your/path/to/dataset <execute_function>
```

All arguments are necessary to execute the code correctly. To avoid producing errors, the main dataset directory should comply with this format: datasetName/raw, where *raw* is the folder containing all subdirectories as well as files for conversion, and *datasetName* is linked with the appropriate function to assign titles and composers to each file. As an `<execute_function>`, several functions are currently available and can be

[2]Documentation is available at: `https://www.music21.org/music21docs/#`
[3]Documentation available at: `https://pypi.org/project/pretty_midi/`
[4]Documentation available at: `https://pypi.org/project/converter21/`

run by specifying their names in the command:

- *convert_files*: transforms single-track MIDI, MusicXML, or kern files into MEI files;
- *split_midis*: separates multi-track MIDI files into single-track MIDI files;
- *correct_file*: resolves metronome notation errors in Melosol files.

Additional functions may be developed according to user needs and incorporated into the list of executable functions in the main file.

4.2 The conversion process

The software currently supports conversion exclusively from MIDI, Kern, and MusicXML formats to the MEI format only for single-instrument files. Single-instrument files reduce conversion errors related to metadata, such as the instrument name. For this reason, datasets with single-instrument files were preferred. Therefore, datasets with MIDI files containing multiple instruments can be transformed into single-instrument files datasets preserving the original folder structure. The transformed dataset will be created at your/path/datasetName/splitted with the following command:

`python main.py /your/path/to/datasetName/raw split_midis`.

Once this is complete, all files in the dataset can be converted to MEI format by executing the following command:

`python main.py /your/path/datasetName/splitted convert_files`

In order to convert a dataset other than those already defined (Melosol, ATEPP-1.1 and clean_LMD), a title_composer function must be implemented. For both ATEPP-1.1 and clean_LMD, this function assigns composer and title information based on the folder structure. If conversion is required from formats other than Kern, MusicXML, or MIDI, the parser and converter functions may need to be modified to support the new file type more effectively.

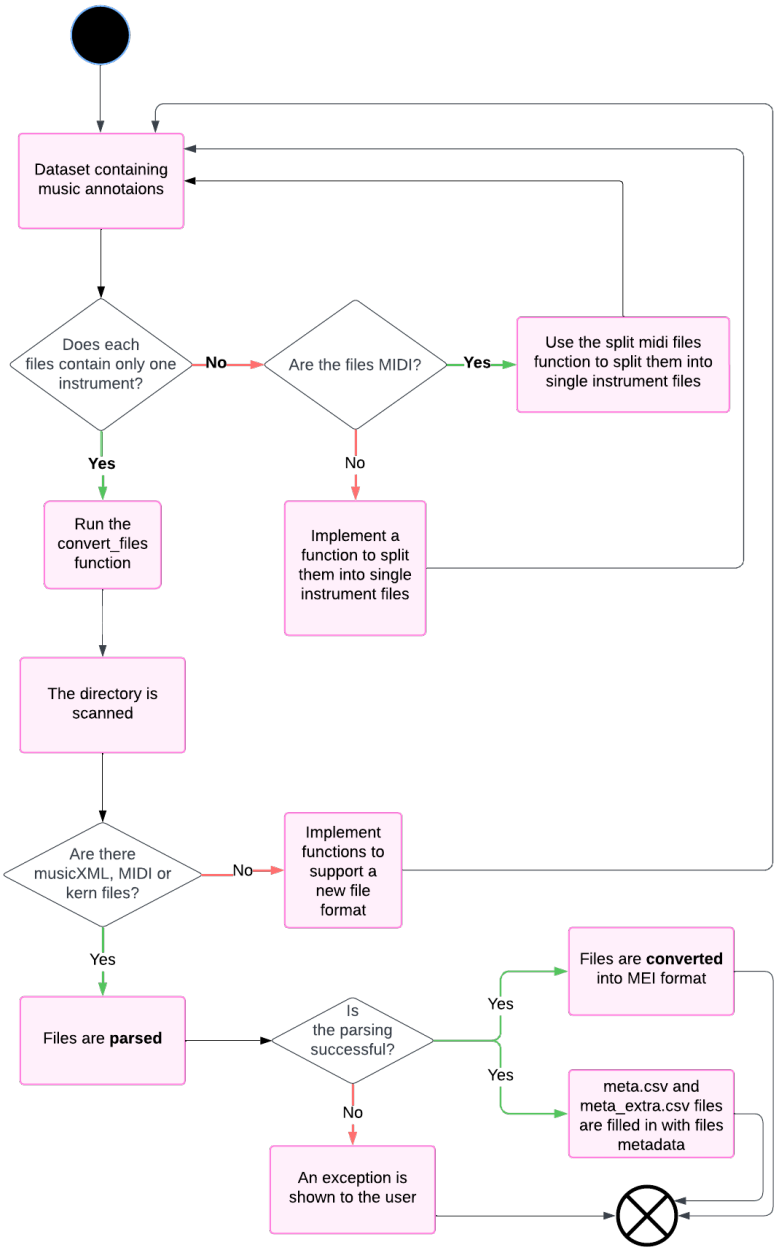


Figure 4.1: Conversion process diagram.

4.3 Problems and solutions

While creating a workflow to transform musical files into the MEI format, a number of technical obstacles were identified and addressed. Initially, *Music21* lacks native support for MEI conversion. To overcome this issue, a library called *Converter21* was identified on the PyPI[5] repository. This library was incorporated to allow for consistent MEI output.

Other issues arose when using Music21's default parser, `music21.converter.parseFile(file_path)`, to convert files to MIDI. As the default parser occasionally introduced errors in the conversion process, such as overlapping notes that should have been played consecutively. The error was mitigated by incorporating specific arguments into the converter function:

```
music21.converter.parseFile(file_path, format='midi',
                forceSource=True, quantizePost=True,
                quarterLengthDivisors=(12, 16))
```

The argument `format='midi'` is used to explicitly tell to the Music21 converter that the current file is a MIDI file. `fourceSource=True` is used to force Music21's parser to take the source file as-is, ignoring any possible inconsistencies that would normally raise errors or warnings. This ensures that the parsing process continues even if there are some minor issues in the file, improving the reliability of MIDI conversions when working with imperfect files. `quantizePost=True` instructs music21 to apply quantisation after parsing the file, by adjusting note durations to fit a standardised rhythmic grid (e.g., eighth notes or quarter notes). This is particularly beneficial when working with MIDI files, as they often exhibit irregular or non-standard timing. In the MIDI format, note timing is specified in absolute microseconds, which may require adjustments for proper analysis or conversion into other formats such as MEI or MusicXML.

[5]The Python Package Index (PyPI) is a repository for Python language software, aiding in the discovery and installation of tools developed and shared by the Python community.

Finally, `quarterLengthDivisors=(12, 16)` defines the allowable divisions of a quarter note during quantisation. A divisor of 12 corresponds to ternary rhythm, enabling note durations to be divided into triplets, while a divisor of 16 corresponds to binary rhythm, accommodating standard divisions such as half notes (1/2), quarter notes (1/4), eighth notes (1/8), and sixteenth notes (1/16), which follow powers of two. Also when separating multiple tracks within MIDI files, Music21 proved to be inadequate; it occasionally altered note values or lost critical information. Consequently, the pretty_midi library was selected as a more effective alternative.

The default method to convert a MIDI file to MEI using both Music21 and Converter21 libraries is:

```
import converter21
import os
import music21

fp = os.path.join('..', 'provaFile',
↪  'allemande_fifth_fragment.midi')
score = music21.converter.parseFile(file_path, format='midi',
                forceSource=True, quantizePost=True,
                quarterLengthDivisors=(12, 16))
converter21.register()
out = os.path.join('..', 'provaFile',
↪  'allemande_fifth_fragment.mei')
score.write('mei', out)
```

However, this method does not preserve at all additional metadata such as tempo and instrument details. To solve this problem, `xml.etree.ElementTree` was used for parsing the MEI file once created by the Converter21, in order to add those informations. Another metadata-related challenge involved title and composer information. While MIDI files inherently lack these details, even other file types in the datasets did not

consistently contain this information. Instead, valuable bibliographical data was often embedded in the folder structure of each dataset. To systematically extract and assign titles and composers, a `title_composer()` function was implemented for each dataset, tailored to the unique folder structure and naming conventions, as no standardised format was observed across the different datasets. Additionally, within the Melosol dataset, certain files contained incorrect instrument labels, where each instance of `*I"Piano` appeared instead of `*IPiano` resulting in missing instrument markings after conversion. To solve this problem, a custom Python function was developed to automatically correct these errors within all Kern files, by removing the extra `"` character from each instrument indication.

Finally, due to the large number of files requiring conversion, processing efficiency became a significant concern. To accelerate the workflow, multiprogramming was implemented in Python, allowing for concurrent file processing, which reduced overall conversion time and optimised resource usage.

Chapter 5

Technical Validation of Data Conversion

This chapter explains the scoring method created to assess the accuracy of the conversion process. The accuracy score is obtained by comparing the original music file with its MEI-converted version. The Music21 Python library was used for this task, as it can parse all the music file formats used in the project: MIDI, MusicXML, Kern, and MEI. To improve the reliability of the evaluation, at least three original scores from each format, along with their MEI-converted versions, were selected for analysis.

5.1 Criteria for assessing the conversion accuracy

A music file represents many different elements like a traditional music score: bars, notes, key signature, metronome indications, articulations, dynamics, etc. For further explanation on how the different digital formats encode these informations, refer to Chapter 1. To ensure a reliable evaluation of the conversion process for each of the following elements, an accuracy score was calculated:

- number of measures
- notes
- time signature
- metronome indications
- key signatures
- dynamics
- clefs

The number of measures is definitely the easiest way to check out whether the conversion was successful and accurate or not: if the number of measures is different, then some notes are surely missing, but also some others elements listed above might not be well converted. On the other hand, if the number of measures is correct, we cannot take for granted that the conversion was successful: the other information can still be incomplete or contain some errors.

To verify the accuracy of note conversion, the notes in a score are grouped into trigrams, i.e. sets of three notes. For each note, its pitch, rhythm and articulations are assessed using the following attributes of the Music21 note object: `note.name`, `note.octave`, `note.pitch`, `note.pitch.accidental`, `note.quarterLength`, and `note.articulations`. Trigrams are considered mismatched if there are differences in any of these parameters between the original and the converted version. However, in MIDI files the length of a note is expressed with extreme precision in milliseconds, which can lead to slight discrepancies when converting these values to the fractional notations used in written music. To take into account these differences, the note accuracy score for MIDI files is calculated as the average of two accuracy scores: one assessing all the attributes previously mentioned and another excluding `note.quarterLength`. This adjustment minimises the impact of slight timing differences on the overall accuracy assessment.

Provided that both the number of measures and the notes are converted without any mistake, it can be stated that the identity of a musical piece is preserved. For example, if the pitch is correct, it does not really matter in which clef it is represented, as the listener would hear the correct pitch. However, an accurate representation of other information, such as time signatures, metronome markings, key signatures, dynamics, and clefs, is important for musicologists, musicians, and computers to properly analyse and catalogue works of a composer. For instance, the MEMUK knowledge graph could be used to observe which time signatures are preferred by a particular composer. Unfortunately, if the MEI files do not contain this information, the results obtained may not be accurate.

Then, the overall accuracy score for each original file format was determined using a weighted mean of these partial accuracy scores:

$$\frac{(S_1 * W_1) + \ \ldots \ + (S_n * W_n)}{(W_1 + \ \ldots \ + W_n)}$$

where S_i represents the accuracy score for each element, and W_i is the assigned weight. The weights in table 5.1 were assigned to each partial accuracy score according to the considerations mentioned above.

Element	**Weight**
Number of measures	5
Notes	5
Time signature	3
Metronome	3
Key signatures	2
Dynamics	1
Clefs	1

Table 5.1: Weights assigned to the partial accuracy scores on a scale from one to five

5.2 Accuracy Scores

The overall accuracy scores for the conversion process were as follows:

- 83.54% for MusicXML files, with errors resulting from missing conversion of time signatures, metronome markings, key signatures, and clefs;
- 100% for KERN files;
- 90.67% for MIDI files, where the sole errors occurred due to trigram mismatches, related to discrepancies in note duration.

Chapter 6

The MEMUK Knowledge Graph

A Knowledge Graph (KG) generated by the integration of several MIR datasets can be explored through symbolic reasoning to derive novel musical knowledge and test musicological hypotheses [9]. As explained in Chap. 2 a KG uses RDF triples (subject - predicate - object) to describe the structure of entities and their connections with other entities. When it comes to complex domains, like music, this structure allows more meaningful interpretations, by querying data with the SPARQL[1] language.

In opposition to the Choco KG mentioned in Chap. 2, which semantically integrates harmonic data[2], the MEMUK KG focuses on the horizontal aspect of music: the melody. Therefore, all the notes and the various music notation elements that appear in a musical score are described. In addition, also metadata regarding the title of the score, the name of the composer and the instrumentation are represented. MEMUK can be queried at: `https:`

[1]SPARQL is a semantic query language for retrieving and manipulating data stored following the Resource Description Framework (RDF), for more detailed information on the syntax and semantics of the SPARQL language please refer to the W3C documentation available at `https://www.w3.org/TR/sparql11-query/`

[2]Harmonic data represents the vertical aspect of music, i.e. the chords and their role in "combining notes in music to produce a pleasing effect greater than the sum of its parts" [44]

`//polifonia.disi.unibo.it/memuk/sparql`.

The MEMUK KG was generated using the **sparql-anything** tool, a "system for Semantic Web re-engineering that allows users to query *anything* with SPARQL", as the developers state on GitHub in their repository [45]. Actually, the supported formats are: XML, JSON, CSV, HTML, Excel, Text, Binary, EXIF, File System, Zip/Tar, Markdown, YAML, Bibtex, DOCx, PPTX. Therefore, with sparql-anything it's possible to generate RDF data from MEI files, which are XML-based, and save these data into `.ttl` files[3].

6.1 Knowledge Graph construction

6.1.1 System Requirements

To create MEMUK on your local machine and test the software that genaretes it, SPARQL Anything requires a version equal or greater than JAVA 17. The SPARQL Anything executable jar can be found in the official release page on GitHub: `https://github.com/SPARQL-Anything/sparql.anything/releases`. In fact, two different version of SPARQL Anything could be downloaded: one is the `sparq-anything-<version>.jar`, and the other one is the `sparql-anything-server-<version>.jar`. The first version is the one used in this projects. Meanwhile, the second one is a server and its UI can be accessed at the address `http://localhost:3000/sparql` after running the jar file from the Command Line Interface (CLI) as follows:

```
$ java -jar sparql-anything-server-<version>.jar
```

This UI can be very handy to get familiar with the SPARQL syntax and to try out queries. However, the first version is the one used in this project, and should be downloaded into the bin folder.

[3]TTL (pronounced 'turtle') stands for "Terse RDF Triple Language" and is a file format used to express RDF data. This format is a W3C standard that is described as a "general-purpose language for representing information in the web". Representing RDF, .ttl files store facts as triples. [46].

While a Docker installation is optional, it is highly recommended, both to provide a secure and consistent environment for running the code and for minimising the risk of compatibility issues caused by different programming setups. The *Dockerfile* takes care of installing the correct Python version and all the required libraries for a smooth code execution.

6.1.2 Transformation workflow

Along side with the creation of a SPARQL query to transform XML data from MEI files to RDF, using the structure of the PON ontology, other tools were created to generate the MEMUK KG. All files used to create and test this transformation process are available at `https://github.com/esthy13/memuk/tree/main/kg-generation`. Each MEI file is converted to RDF using the mei2rdf.py script. This Python script accepts parameters from the CLI, such as the input and output path of the file and the path of SPARQL Anything jar's location. kg-generation.py, instead, automatises the transformation process for all the files contained in a specific folder.

6.2 Modeling melodic data with PON

Four modules of the PON ontology were used to model melodic data, i.e. the musical notation elements:

- **The Core Ontology** (core): defines general concepts, relationships, and ontology design patterns. It's specialised by the other of the PON ontology [47].

- **The Music Meta Ontology** (mm): "is a rich flexible model to describe Western music metadata and its provenance at different levels of granularity [...], with automatic alignments to the Music Ontology [18], the DOREMUS Ontology [20], and Wikidata (`https://www.wikidata.org/`)" [48].

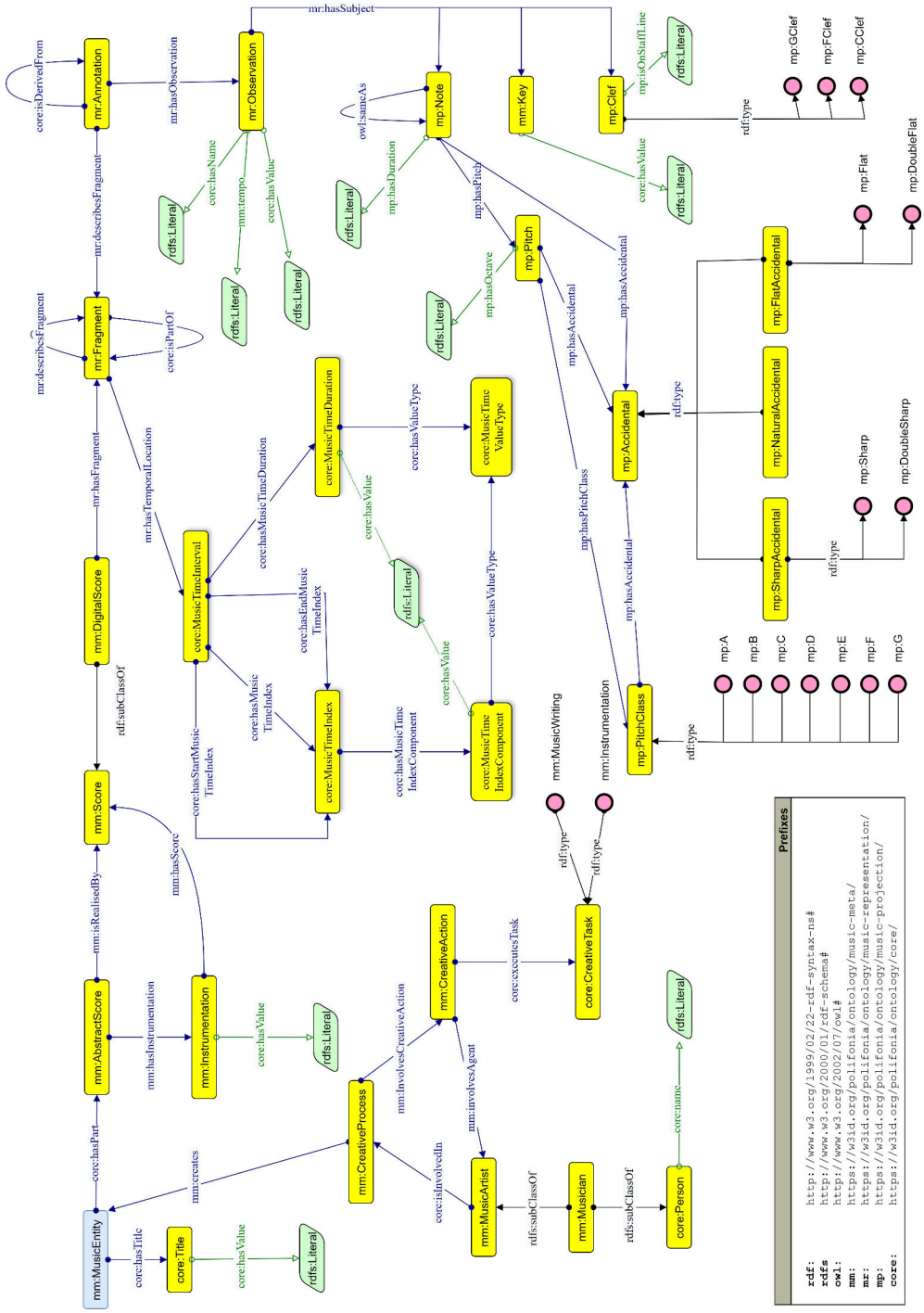


Figure 6.1: Diagram of the relationship between PON modules in MEMUK.

- **The Music Representation Ontology** (mr): Schema for musical object analysis, connecting scores, audio, and annotations.

- **The Music Projection Ontology** (mp): establishes a formal representation of musical entities, including both traditional musical elements, such as notes and chords, as well as more subjective annotations, like mood and danceability [47].

Figure 6.1 illustrates the relationship between these modules. In order to create the diagram the Graffoo notation was used (`https://essepuntato.it/graffoo/`): yellow boxes are classes, blue/green, arrows are object/-datatype properties, pink circles are individuals and green polygons are datatypes. The central entity is `mm:MusicEntity`, and information object defined as the "sum of all the elements" that make up a piece of music [6][49]. A `mm:MusicEntity` is created by a `mm:CreativeProcess`, which involves a `mm:CreativeAction`, to indicate the execution of a `mm:CreativeTask` of type `mm:MusicWriting` and `mm:Instrumentation`,as each music piece was composed for a particular instrument or a designated ensemble of instruments.

The metadata retained from each piece of music were the title (`core:Title`) and the composer(`mm:MusicArtist`). `mm:Musician` represents the composer of each piece, as a subclass of `mm:MusicArtist`, involved in the `mm:CreativeProcess`, and a subclass of the more general class `core:Person`.

Each `mm:MusicEtity` has a part, represent by the `mm:AbstractScore`. This class assures reusability of the the MusicMeta module even when the music piece has not been transcribed as real score, as a part of an oral tradition, or as music created during an improvisation. Within this framework, each `mm:AbstractScore` is realised by a `mm:Score`. Each `mm:MusicEntity` is a `mm:DigitalScore` a subclass of `mm:Score`.

A `mm:DigitalScore` is then split split into several `mr:Fragment`s. Each `mr:Fragment` is described by a `mr:Annotation`. A `mr:Annotation` may have different kind of `mr:Observation`s:

- `mp:Note`s have three properties: `mp:Pitch`, `mp:Accidental`, and a du-

ration, expressed as an `rdfs:Literal`, i.e. a string or a number, which corresponds to the value of the `xyz:dur` property of a <Note> tag in a MEI file.

- `mm:Key` refers to the Key signature of a musical piece, which is also represented as a `rdfs:Literal`, composed by the concatenation of the `xyz:keyPname`, `keyMode` and `keySig` attributes of a <KeySig> tag in MEI files.

- `mm:Clef` has two attributes, one indicating the type of key, `mp:GClef`, `mp:CClef` and `mp:CClef`, and the other one indicates on which staff line the key is positioned.

As the PON has no class to represent meter and tempo information, these elements are represented as attributes of the `mr:Observation`, by giving a name a name to a such `mr:Observation`, either `"meter"` or `"tempo"`. Furthermore, the value of this observation is associated as an `rdf:Literal`, and meter is the rythmic meter represented in the beginning of a section or of an entire music piece, while tempo is a number expressing a BPM measurement.

Each `mr:Fragment` is also associated with a temporal location, represented by `core:MusicTimeInterval`, which enables the expression of musical time intervals in different ways, depending on the type of musical object. For instance, if the fragment is a part of a recording, its temporal intervals are likely measured in seconds. Although MEMUK currently contains exclusively musical scores, this feature provides flexibility for potential future extensions to include recordings.

To express music time intervals, which are defined by `core:MusicTimeIndex`, a combination of measures and beats is used. A musical time interval has a start time index and an end time index. Each index is defined by one or more `core:MusicTimeIndexComponent`s, each defined by a value (`core:hasValue`) and a value type (`core:MusicTimeValueType`). Additionally, a musical time interval also includes a duration (`core:MusicTimeDuration`) represented as a value and a value type,

which for musical scores, is measured in beats.

6.3 Example

This section shows an example of how data is transformed from a MEI file into RDF. The file used for this example comes from the Melosol dataset. Its original format is a kern file, available in the repository of this project at: `https://github.com/esthy13/memuk/blob/main/melody/partitions/Melosol/raw/Berkowitz107.krn`. The visual representation of the original file (Figure 6.2) was created thanks to the online digital music editor `https://verovio.humdrum.org/`.

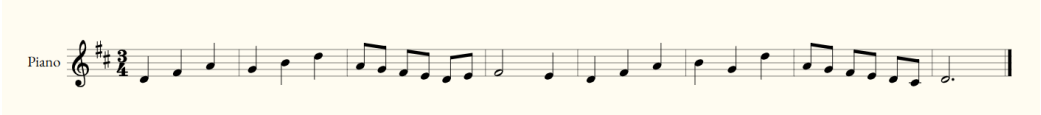


Figure 6.2: Visual representation of the file Berkowitz107.krn .

The file was then transformed into a MEI file following the conversion process described in Chapter 4 (Figure 6.3). Title and composer's metadata are encoded in the MEI file in the tag `<meiHead>` (Figure 6.4).

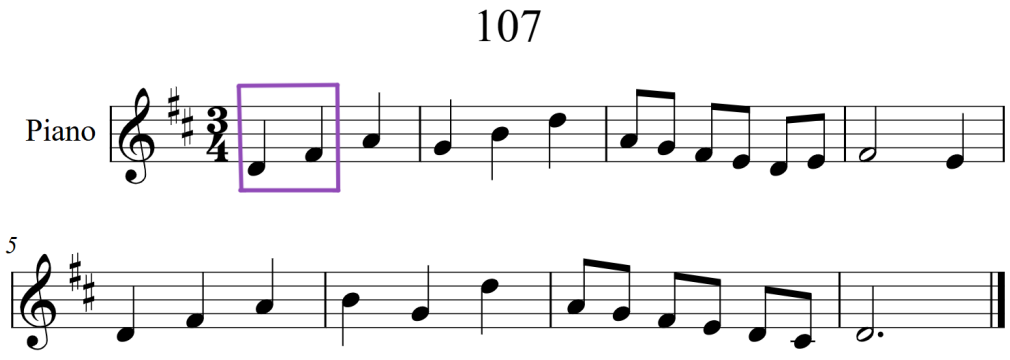


Figure 6.3: Visual representation of the file Melosol_000009.mei .

```
<meiHead>
   <fileDesc>
      <titleStmt>
         <title>107</title>
      </titleStmt>
      <pubStmt>
         <unpub>This MEI file was created by converter21's MEI writer. When
                published, this unpub element should be removed, and the
                enclosing pubStmt element should be properly filled out.</unpub>
      </pubStmt>
   </fileDesc>
   <encodingDesc>
      <appInfo>
         <application version="3.2.0">
            <name>converter21</name>
         </application>
         <application>
            <name>music21 v.9.1.0</name>
         </application>
      </appInfo>
   </encodingDesc>
   <workList>
      <work xml:id="work0_encoded" type="encoded">
         <title type="uniform">
            <titlePart type="main" analog="humdrum:OTL">107</titlePart>
         </title>
         <composer>
            <persName analog="humdrum:COM">Berkowitz</persName>
         </composer>
      </work>
   </workList>
</meiHead>
```

Figure 6.4: First fragment of the file Melosol_000009.mei: tag `<title>` contains the title of the music piece and the tag `<persName>` contains the name of the composer.

```
<measure n="1">
  <tempo /><staff n="1">
    <layer n="">
      <note dur="4" pname="d" oct="4" stem.dir="up" />
      <note dur="4" pname="f" oct="4" stem.dir="up" accid.ges="s" />
      <note dur="4" pname="a" oct="4" stem.dir="up" />
    </layer>
  </staff>
</measure>
```

Figure 6.5: The second fragment of the file Melosol_000009.mei shows how the notes in the first bar are econded in the MEI notation.

The first two notes in measure one are D and F♯. They are modeled with the Polifonia Ontology Network as two `mr:NoteAnnotations`, each containing an `mr:Observation` referencing a `mp:Note`. As shown in Figure 6.6, their pitches, duration, and octave are correctly retained. The first note in the MEI file has the following attributes: `dur="4"`, `pname="d"`, and `oct="4"` (Figure 6.5). The same values are also reported in Figure 6.6 as `mp:hasDuration "4"^^xsd:integer`, `mp:D`, and `mp:hasOctave "4"^^xsd:integer`.

6.3.1 Queries Examples

Two queries were formulated in order to show the capabilities of this knowledge graph:

- Which music pieces were composed by *Berkowitz*?
- What are the note in measure 1 of the music piece called *104* from *Berkowitz?*

The written SPARQL code for these queries can be found at: `https://github.com/esthy13/memuk/tree/main/kg-generation/queries4kg`.

A limitation in the creation of more queries was caused by the management of notes indexes in MEI files. In a MEI file beam[4] and notes are

[4]A beam is the graphical lines that connects a group of two or more notes, like in measure 3 and 7 from Figure 6.3

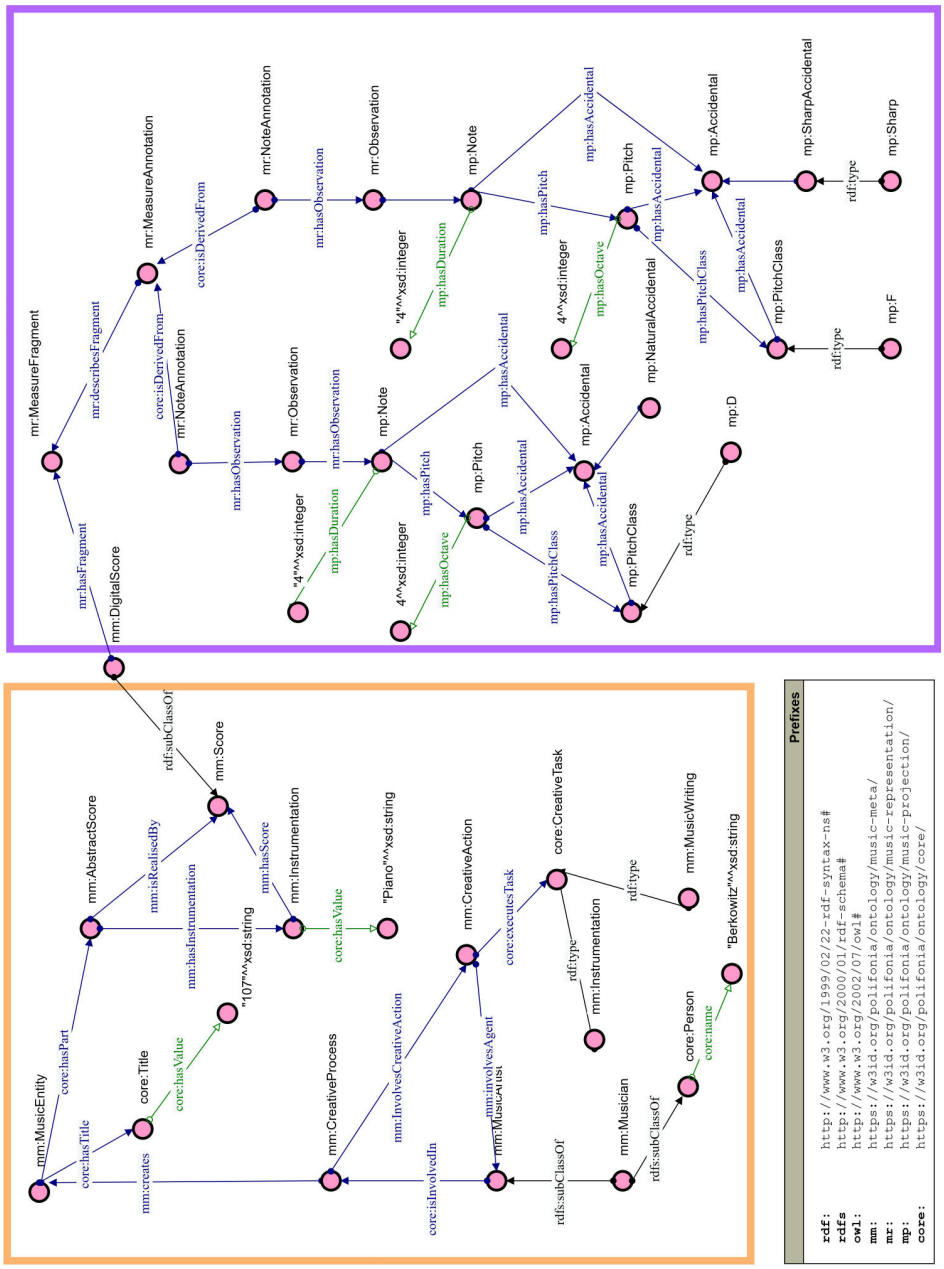


Figure 6.6: Example of data modelled using Polifonia Network Ontology, extracted from track 107 from the Melosol dataset.The orange rectangle encodes the metadata: title, composer and instrument. The purple rectangles shows how the first two notes of the first measure are modelled.

considered at the same level causing duplicated index values when the two elements are mixed together in the same bar.

6.4 Challenges

Although sparql-anything is a really powerful tool, offering many examples of uses and quickstart guides that can be found in `https://github.com/SPARQL-Anything`, converting MEI files to.ttl files presented several challenges. First, given a limited time of less than three weeks to convert these files, the learning curve to create SPARQL queries and automate the conversion process for an entire dataset is quite steep, especially for those who are not familiar with SPARQL or RDF.

Another problem encountered while designing the SPARQL conversion query was handling missing data. For example, files coming from the Melosol database [41] did not contain tempo information (neither expressed as a BPM measurement nor as an agogic indication). The solution in these situations consists of using the `OPTIONAL { }` clause in SPARQL. This clause handles missing information, without returning empty query results, by inserting into the brackets the triples that should be retrieved.

6.5 Future improvements

In MEMUK, triples are generated according to PON, "a set of new ontologies formalising the semantics of music representation, metadata, annotation, analysis, mediums of performance (instruments), and historical sources (provenance), enabling the creation of interoperable KGs from music datasets" [47]. However MEMUK could still be further improved.

First of all, a larger number of datasets could be integrated, following Choco's workflow. This way, considerations and analysis between the two KGs on overlapping repertoire could be conducted. Furthermore, each `mm:MusicEntity` could be connected to external resources, such as cata-

loguing information from online music databases like MusicBrainz (`https://musicbrainz.org`), as already showcased in Choco.

Lastly but not least, practical applications could be showcased to enhance the usability of MEMUK. For instance, after solving the problem related to note indexes, queries to check for the presence of pitch sequences could be written. Both musicians and musicologists could benefit of insights into melodic patterns, for example, gaining a deeper understanding of their diffusion and development across historical periods or among certain composers. Therefore, new perspectives could be offered, or existing beliefs about music history and theory could be validated.

Conclusion

The aim of this project was to develop a robust data integration workflow to improve the reproducibility of MIR experiments, such as testing music recommendation or genre classification systems. MEI turned out to be the best annotation format for a melodic music dataset that is directed to future expansion, as this formats allows to represent both classical, commercial, contemporary and Ancient music. A Python conversion software was created to convert files from MusicXML, MIDI and kern to the MEI format. The conversion has been possible thanks to Music21, Converter21 and PrettyMidi libraries. This thesis contributes to the creation of a heterogeneous melodic dataset that encompass various genres of Western music, containing 126,352 single-instrument MEI files.

A semantic integration of melodic music data is provided by MEMUK. This knowledge graph aims to model the relationships between the music notation elements that compose a music piece, using the Polifonia Ontology Network. It's a semantic approach that enables the integration of diverse datasets and provides tools for comparing melodic aspects, such as identifying recurring pitch sequences across multiple music pieces. In addition, it ensures interoperability with other Polifonia-based KG like the Choco Chord Corpus.

Finally, the music industry could benefit from this example of melodic knowledge graph to develop more efficient music data management systems, or improve music recommendation systems by using queries that check pitch similarities.

6.6 Future work

To improve the creation of MEMUK, further work is required. First of all, all possible musical notation elements needs to be handled, paying particular attention on how to model rests and chords.

The limitations caused by the management of notes indexes in MEI files should be solved, in order to avoid duplicated note index values when beam and notes elements are mixed together in the same bar. Once this problem will be solved, MEMUK could be queried for detecting occurrences of pitch sequences.

A further development could, also, consist in modelling semantic relationships from ancient western notation systems.

Lastly but not least, both the integrated dataset and MEMUK could be enlarged by exploiting the data integration workflow and software created in this thesis, even by adding supports for other digital file formats.

Appendix A

Licenses

A.1 GNU General Public License

This license guarantees four fundamental freedoms to users:

1. The freedom to utilize the software for any intended purpose.
2. The freedom to access and analyse the source code.
3. The freedom to create and distribute copies of the software.
4. The freedom to modify the source code, develop derivative versions of the software, and redistribute them.

However, this license imposes a crucial restriction: neither the original software nor any of its modifications may be converted into proprietary software. In other words, both the initial application and its derivatives must remain free software, preserving their openness and accessibility under the same licensing terms. [50]

A.2 MIT License

This is a licence created by the Massachusetts Institute of Technology to distribute free software. It grants permission to use, copy, modify, merge, publish, distribute, sublicense and sell copies of the software, as long as the copyright notice and the text of the MIT licence remain intact. No warranty or liability for damages resulting from the use of the software is provided by the copyright holder, authors or other contributors, thus protecting the parties involved from legal consequences. Therefore, the software is provided 'as is' without any guarantees or assurances regarding its performance, quality or suitability for specific purposes, reinforcing the lack of warranty and liability protection for the creators and distributors of the software. It is compatible with other licences, i.e. software, components and libraries licensed under the MIT licence can be integrated with projects using different licences. For example, the MIT licence is also compatible with popular copyleft licences such as the GNU General Public License (GPL). However, while the MIT licence allows integration and sublicensing without the obligation to share modifications or derivative works, the GPL requires that the derivative work must also be distributed under the same licence, preserving the ethics of free and open source software. [51]

A.3 Creative Commons licences

- **CC0**: allows creators to distribute their material as being in the public domain so that it can be used and redistributed in any way and format, without restrictions.
- **CC BY**: allows re-users to distribute, remix, adapt and build upon the material in any manner or format, provided attribution is given to the creator. The licence permits commercial use.
- **CC BY-SA**: permission to distribute, remix, adapt and build upon the material in any manner or format, provided credit is given to the

creator. The licence permits commercial use. Derivative material must be provided with the same licence.

- **CC BY-NC**: permits reusers to distribute, remix, adapt and build upon the material in any manner or format for non-commercial purposes, provided credit is given to the creator.
- **CC BY-NC-SA**: permission to reuse, remix, adapt and build upon the material in any manner or format, for non-commercial purposes only, provided the creator is indicated. Modified material must be provided with the same type of licence.
- **CC BY-ND**: permits copying and redistribution of the material in any medium and format. The licence allows commercial use but does not permit the creation of derivative works. Credit must be given to the creator.
- **CC BY-NC-ND**: allows material to be copied and redistributed for non-commercial purposes only, provided the creator is credited. This is the most restrictive type of Creative Commons licence.

[52]

Bibliography

[1] M. Schedl, E. Gómez, and J. Urbano, *Music Information Retrieval: Recent Developments and Applications*. Now Publishers, 2014.

[2] M. Gorgoglione, A. C. Garavelli, U. Panniello, and A. Natalicchio, "Information Retrieval Technologies and Big Data Analytics to Analyze Product Innovation in the Music Industry," *Sustainability*, vol. 15, no. 1, L. Aldieri, Ed., p. 828, 2023. DOI: 10.3390/su15010828.

[3] UNESCO, *Cultural Times: The First Global Map of Cultural and Creative Industries*, Accessed on 13/10/2022), 2015. [Online]. Available: https://en.unesco.org/creativity/files/culturaltimesthefirstglobalmapofculturalandcreativeindustriespdf.

[4] C. Hatton, *IFPI Issues Global Music Report 2021*, Accessed on 13/10/2022), 2021. [Online]. Available: https://www.ifpi.org/ifpi-issues-annual-global-music-report2021/.

[5] H. W. Chesbrough, *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Cambridge, MA, USA: Harvard Business Press, 2003, ISBN: 978-1-57851-837-1.

[6] J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri, and V. Presutti, "ChoCo: a Chord Corpus and a Data Transformation Workflow for Musical Harmony Knowledge Graphs," *Scientific Data*, vol. 10, Sep. 2023. DOI: 10.1038/s41597-023-02410-w.

[7] G. E. Poliner and D. P. W. Ellis, "A Discriminative Model for Polyphonic Piano Transcription," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, Jun. 2006. DOI: 10.1155/2007/48317.

[8] W. Goebl. "The Vienna 4x22 Piano Corpus." English. (1999).

[9] V. Carriero, F. Ciroku, J. de Berardinis, *et al.*, "Semantic Integration of MIR Datasets with the Polifonia Ontology Network," English, in *International Society for Music Information Retrieval Conference (ISMIR 2021)*, 2021.

[10] Garzanti, Ed., *La nuova enciclopedia della musica*, 1st. Garzanti, 1983.

[11] M. C. Mazzi, *Raccontare la musica*, 1st, A. Parisini, Ed. Eufonia, 2016.

[12] *Enciclopedia Treccani*, Accessed on the 29/09/2024. [Online]. Available: https://www.treccani.it/enciclopedia/.

[13] M. Müller, "Music Representations," in *Fundamentals of Music Processing, Using Python and Jupyter Notebooks*. Springer, 2021, pp. 1–35.

[14] G. Wiggins, D. Müllensiefen, and M. Pearce, "On the Non-Existence of Music: Why Music Theory is a Figment of the Imagination," *Musicae Scientiae*, vol. 14, pp. 231–255, Mar. 2010. DOI: 10.1177/10298649100140S110.

[15] F. Zalkow and M. Müller, *Symbolic Format: CSV*, Accessed on the 29/09/2024, 2015. [Online]. Available: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S2_CSV.html.

[16] [Online]. Available: https://dictionary.cambridge.org/it/dizionario/inglese/critical-edition.

[17] C. S. Sapp, *Humdrum Toolkit User Guide*, Accessed on the 28/09/2024, Aug. 2024. [Online]. Available: https://www.humdrum.org/guide/.

[18] Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson, "The Music Ontology," in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, Sep. 2007.

[19] S. Rho, S. Song, E. Hwang, and M. Kim, "Comus: Ontological and Rule-based Reasoning for Music Recommendation System," in *Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg, 2009.

[20] P. Lisena and R. Troncy, "Doing Reusable Musical Data (DOREMUS)," in *Proceedings of Workshops and Tutorials of the 9th International Conference on Knowledge Capture (K-CAP2017)*, ser. CEUR Workshop Proceedings, vol. 2065, Austin, Texas, USA: CEUR-WS.org, Dec. 2017.

[21] D. P. Escuredo, *The Tonality Ontology*, Accessed on the 25/11/2024., Mar. 2008. [Online]. Available: `http://purl.org/ontology/tonality/`.

[22] G. Fazekas, Y. Raimond, K. Jacobson, and M. Sandler, "An Overview of Semantic Web Activities in the OMRAS2 Project," *Journal of New Music Research*, vol. 39, Dec. 2010.

[23] S. M. Rashid, D. De Roure, and D. L. McGuinness, "A Music Theory Ontology," in *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music*, ser. SAAM '18, New York, NY, USA: Association for Computing Machinery, 2018, pp. 6–14.

[24] S. S.-s. Cherfi, C. Guillotel, F. Hamdi, P. Rigaux, and N. Travers, "Ontology-based Annotation of Music Scores," in *Proceedings of the Knowledge Capture Conference*, ser. K-CAP 2017, New York, NY, USA: Association for Computing Machinery, 2017.

[25] J. Jones, D. de Siqueira Braga, K. Tertuliano, and T. Kauppinen, "MusicOWL: The Music Score Ontology," in *Proceedings of the International Conference on Web Intelligence*, ser. WI '17, New York, NY, USA: Association for Computing Machinery, 2017.

[26] A. Meroño-Peñuela, R. Hoekstra, A. Gangemi, *et al.*, "The MIDI Linked Data Cloud," in *The Semantic Web – ISWC 2017*, Cham: Springer International Publishing, 2017.

[27] G. Fazekas and M. B. Sandler, "The Studio Ontology Framework," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami, Florida, USA: University of Miami, Oct. 2011.

[28] T. Wilmering, G. Fazekas, and M. B. Sandler, "The Audio Effects Ontology," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, Curitiba, Brazil, Nov. 2013.

[29] H. H. Kim, "A Semantically Enhanced Tag-based Music Recommendation Using Emotion Ontology," in *Intelligent Information and Database Systems*, Springer Berlin Heidelberg, 2013.

[30] K. Jacobson, Y. Raimond, and M. B. Sandler, "An Ecosystem for Transparent Music Similarity in an Open World," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe International Conference Center, Kobe, Japan: International Society for Music Information Retrieval, Oct. 2009.

[31] X. Cui, X. Qu, D. Li, Y. Yang, Y. Li, and X. Zhang, "MKGCN: Multi-Modal Knowledge Graph Convolutional Network for Music Recommender Systems," *Electronics*, Dec. 2023.

[32] D. Weigl, T. Crawford, A. Gkiokas, *et al.*, "FAIR Interconnection and Enrichment of Public-Domain Music Resources on the Web," *Empirical Musicology Review*, vol. 16, pp. 16–33, Dec. 2021. DOI: 10.18061/emr.v16i1.7643.

[33] B. Fields, K. R. Page, D. De Roure, and T. Crawford, "The Segment Ontology: Bridging Music-generic and Domain-specific," in *Proceedings of the 2011 IEEE International Conference on Multimedia and Expo (ICME 2011)*, Barcelona, Catalonia, Spain: IEEE Computer Society, Jul. 2011.

[34] A. Allik, G. Fazekas, and M. B. Sandler, "An Ontology for Audio Features," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York City, United States, Aug. 2016.

[35] C. Sutton, Y. Raimond, and M. Mauch, *The OMRAS2 Chord Ontology*, version Draft 1, Accessed on the 25/11/2024, Oct. 2007. [Online]. Available: `http://purl.org/ontology/chord/`.

[36] N. Harley and G. Wiggins, "An Ontology for Abstract, Hierarchical Music Representation," in *Demo at the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015.

[37] N. Shea, L. Reymore, C. White, L. VanHandel, B. Duinker, and N. Biamonte, "Diversity in music corpus studies," *Music theory online*, vol. 30, no. 1, 2024, ISSN: 1067-3040. DOI: `10.30535/mto.30.1.8`. [Online]. Available: `https://mtosmt.org/issues/mto.24.30.1/mto.24.30.1.shea_et_al.html`.

[38] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *International Society for Music Information Retrieval Conference*, 2020, pp. 534–541.

[39] H. Zhang, J. Tang, S. Rafee, S. Dixon, G. Fazekas, G. Wiggins, *et al.*, "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, Bengaluru, India, Dec. 2022.

[40] P. van Kranenburg, M. de Bruin, L. P. Grijp, and F. Wiering, *The Meertens Tune Collections*, Dec. 2014.

[41] D. J. Baker, "MeloSol Corpus," *Empirical Musicology Review*, vol. 16, pp. 106–113, 1 Dec. 2021. DOI: `10.18061/emr.v16i1.7645`.

[42] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[43] C. Erbele, "Ralph Berkowitz, Dean of Tanglewood (1946 - 1964) piano soloist, accompanist and teacher administrator and lecturer 1910 -," *Western Jewish History Center Judah Magnes Museum*, 1991.

[44] N. D. Cook and T. X. Fujisawa, "The psychophysics of harmony perception: Harmony is a three-tone phenomenon.," *Empirical Musicology Review*, 2006.

[45] L. Asprino, E. Daga, A. Gangemi, and P. Mulholland, "Knowledge Graph Construction with a Façade: A Unified Method to Access Heterogeneous Data Sources on the Web," *ACM Transactions on Internet Technology*, 2022, ISSN: 1533-5399. DOI: 10.1145/3555312. [Online]. Available: https://doi.org/10.1145/3555312.

[46] RDFOX, Oxford Semantic Technologies, *What is a ttl file?* en, Accessed on the 28/11/2024. [Online]. Available: https://www.oxfordsemantic.tech/faqs/what-is-a-ttl-file.

[47] J. de Berardinis, V. A. Carriero, N. Jain, *et al.*, "The Polifonia Ontology Network: Building a Semantic Backbone for Musical Heritage," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14266 LNCS, Springer Science and Business Media Deutschland GmbH, 2023, pp. 302–322, ISBN: 9783031472428. DOI: 10.1007/978-3-031-47243-5_17.

[48] J. de Berardinis, V. A. Carriero, A. Meroño-Peñuela, A. Poltronieri, and V. Presutti, "The Music Meta Ontology: a flexible semantic model for the interoperability of music metadata," *arXiv preprint arXiv:2311.03942*, 2023.

[49] A. Gangemi and S. Peroni, "The information realization pattern," in *Ontology Engineering with Ontology Design Patterns*, IOS Press, 2016, pp. 299–312.

[50] D. Marongiu, "Parte II - Oltre il copyright: la rete "libera"," Italian, in *Il Copyright su Internet*. Maggioli, May 2019.

[51] AppMaster.io. "Licenza MIT." Accessed on the 14/09/2024. (Sep. 2023), [Online]. Available: `https://appmaster.io/it/glossary/licenza-mit`.

[52] *Creative Commons*, Accessed on the 14/09/2024. [Online]. Available: `https://creativecommons.org/share-your-work/cclicenses/`.

[53] H. Vinet, "The Representation Levels of Music Information," in *Computer Music Modeling and Retrieval*, U. K. Will, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 193–209.

[54] *Licenza MIT*, Accessed on the 14/09/2024, Sep. 2023. [Online]. Available: `https://appmaster.io/it/glossary/licenza-mit`.

[55] S. R. Huang, A. Holzapfel, B. L. Sturm, and A.-K. Kaila, "Beyond Diverse Datasets: Responsible MIR, Interdisciplinarity, and the Fractured Worlds of Music," *Transaction of the International Society for Music Information Retrieval*, 2023.

[56] D. J. Baker, "MeloSol corpus," *Empirical Musicology Review*, vol. 16, no. 1, pp. 106–113, 2021.

Ringraziamenti

Desidero innanzitutto ringraziare i due relatori di questa tesi, la prof.essa Valentina Presutti e il Dott. Andrea Poltronieri. Guidandomi nella stesura di questa tesi mi hanno permesso di intrecciare tre grandi passioni: la musica, l'informatica e le lingue straniere. La loro competenza e disponibilità sono state fondamentali per il raggiungimento di questo traguardo.

Un ringraziamento speciale va poi a tutti quelli che mi hanno sostenuto personalmente durante la stesura di questa tesi, in particolare ai miei genitori e ai miei nonni, ma anche a Victor per i suoi consigli di sviluppo software, a Tania per le consulenze musicali, e a Chiara ed Eva per i suggerimenti linguistici.

Voglio anche esprimere profonda riconoscenza a tutti i Professori del corso di Informatica per il Management che ho avuto modo di incontrare e conoscere durante il percorso. Grazie a voi, fin dall'inizio mi sono appassionata a questa materia per me completamente nuova. Mi avete permesso di conoscere sia il rigore che la bellezza di questa disciplina così vasta ed affascinate qual è l'informatica.

Un sentito ringraziamento va anche a tutti i compagni di corso con cui ho condiviso indimenticabili momenti di svago e di studio durante questi tre anni.