Ontologies, Knowledge Graphs, and User Interfaces for Exploration of Irish Traditional Music

Abdul Shahid [a], Rory Sweeney [b], Pushkar Jajoria [b], Danny Diamond [b], Mathieu D'Aquin [c] and James McDermott [b,*]

[a] *School of Business, South East Technological University, Waterford, Ireland*
E-mail: abdul.shahid@setu.ie
[b] *School of Computer Science, University of Galway, Ireland*
E-mails: rory.sweeney@universityofgalway.ie, pushkarjajoria@gmail.com, d.diamond1@universityofgalway.ie, james.mcdermott@universityofgalway.ie
[c] *Laboratoire Lorrain de Recherche en Informatique et ses Applications and Institut des Sciences du Digital, Université de Lorraine, France*
E-mail: mathieu.daquin@loria.fr

Abstract.
Musical patterns are important for many musicological tasks, such as genre classification, identifying common origins of pieces, and measuring similarities between compositions. Our previous research has defined several types of patterns in Irish traditional music and developed tools for extracting them from databases of musical scores. We now wish to enable flexible and efficient querying, open access, preservation, integration with multiple data sources, and user-friendly exploration of the data. To address these needs we use semantic web technologies. We present a music pattern ontology based on the Music Annotation Pattern (an ontology design pattern [13] which formalizes key concepts in musical annotations). We develop a pipeline (Patterns2KG) to process our pattern data through the ontology. We process approximately 40 thousand compositions from two datasets to give a knowledge graph (KG) of approximately 45 million triples. We evaluate the work against pre-developed competency questions. We then elicit requirements for a graphical user interface (GUI) in collaboration with musicologists, and develop custom modular GUI software which interfaces with the KG via SPARQL queries.

Keywords: Music Annotation, Music Patterns, Patterns KG, Music Similarity

1. Introduction

In musicology, musical analysis enhances our understanding of the content of the music. It provides insights into musical style and genre, assessing the compositional techniques used by the composer, tracking the musical development and evolution of different styles and genres over time, classifying music, recognizing genres, and even generating music. In oral traditions, where pieces are transmitted from generation to generation without being written down, finding musical relationships among pieces is particularly important because they can give insight into the evolution of, and connections between, musical traditions.

*Corresponding author. E-mails: abdul.shahid@setu.ie, james.mcdermott@universityofgalway.ie.

Musical analysis may focus on one or more "layers" of the music, such as musical structure, harmony, rhythm, dynamics, and texture. This work follows the most-common approach in studies of Western folk music, focusing on melody, which is defined simply as a sequence of notes, with durations and rests.

Patterns within or between such sequences are the focus of our research. Using multiple definitions of patterns, and tools for extracting patterns from databases of musical scores in the Irish folk tradition, we have previously created a large database of patterns. We wish to present and preserve this pattern data within a robust structure, enabling effective music analysis. Throughout the Polifonia project, we have represented datasets as Linked Open Data (LOD, i.e. ontologies and KGs). The motivation is to enable flexible and efficient querying, open access, preservation, integration across multiple data sources, and open-ended integration with future work [39]. Therefore in the current paper, we describe the process of formalising the pattern data as LOD.

1.1. What is a musical pattern?

Pattern is a central concept in many fields. Mathematics has been called "the science of patterns" [40]; "pattern recognition" is almost a synonym for the field of machine learning; in architecture, Alexander [3] codified common design patterns, a term then adopted by software developers also (and we will see it in this context later).

Pattern analysis plays a significant role in the rich understanding of music. For example, Schenker [38] claimed that repetition "is the basis of music as an art", Bent [7] proposed that "the central act" in all forms of music analysis is "the test for identity" and Lerdahl and Jackendoff [31] state that the importance of "parallelism" [i.e., repetition] in musical structure cannot be overestimated. The more patterns one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece. Pattern detection is seen as a central task in the field of music information retrieval (MIR). A repetition need not consist of literal identity to give rise to a pattern. For example, if a sequence of notes is repeated with transposition (i.e. all notes in the second occurrence have increased or decreased in pitch by some constant), it will certainly be regarded as a repetition and hence a pattern. Patterns can occur due to repetitions within a single melody or voice in a single piece; or between multiple voices in a piece; or between multiple pieces.

Creating a complete analytical taxonomy of pattern types and definitions has not been achieved in previous work, and we do not attempt this. To be concrete, we have focused on a limited taxonomy of pattern types, all based on *n*-grams of scale degree values. In Western music, a pitch can be defined simply as an integer (e.g., in the range [0, 127] in the common chromatic MIDI encoding). For example, the "middle C" note on the piano is MIDI note 60. Several other integer representations for pitch are also common but discussion is out of scope here. We also use the concept of *accented notes*: an accented note is a note which occurs on a beat. With these definitions, we have a simple taxonomy. In our input corpus we count all *n*-grams of consecutive accented notes (represented as scale degree values) for $n = 4 \ldots 10$. In our earlier work, we developed the *FoNN* (FOlk N-gram aNalysis) tools[1] which exhaustively and deterministically extract patterns from compositions. Briefly, all subsequences of length *n* are extracted, and any which occur more than once in the corpus are taken as patterns. Although we focus on Irish traditional music, the representation and all methods in the paper are suitable for other forms of European folk music and to some extent for other forms of music when represented monophonically.

1.2. Patterns and Tune Families in Irish Folk Music

The existence of common melodic patterns[2] between tunes in the Irish tradition was first noted by 19th century collectors George Petrie and William Forde [10]. The influential Irish-American music collector Francis O'Neill devoted a chapter in his 1913 book Irish Minstrels and Musicians to the qualitative identification of related melodies within the Irish tradition [32].

In 1950, American folk music collector and researcher Samuel Bayard published his theory of *tune families*: proposing that many traditional melodies could be traced, in the manner of a family-tree, to prototype melodies [5].

[1] https://github.com/polifonia-project/folk_ngram_analysis

[2] The following section draws on the M.Sc. thesis of author Danny Diamond[18].

In Bayard's words: "*...a group of melodies showing basic interrelation by means of constant melodic correspondence, and presumably owing their mutual likeness to descent from a single air that has assumed multiple forms through processes of variation, imitation and assimilation.*"

Bayard's tune family concept has been widely adopted in research on Irish and other Western folk music [10, 26, 48]. This provides the point of departure for our work. We assume that these important shared patterns can be captured well by a diatonic representation. Indeed, the diatonic representation deals well with the modal ambiguity common in Irish traditional music. We deal with transpositions among tune variants by normalising all melodies to a common tonal centre.

Further, we take advantage of previous work on structure in Irish traditional music to focus on "accented tones" only, i.e. those on the beat. Their importance (in contrast to the variability of the unaccented tones) in defining a stable outline melody crops up consistently in the literature [8, 43, 47, 48].

Thus, while we have worked with multiple definitions of patterns, in the current work we represent a pattern as an *n*-gram of integers representing a sequence of diatonic scale degrees of accented notes, which occurs more than once in the corpus. For example, the pattern (7, 1, 7, 6, 1) is a 5-gram pattern visible in Fig. 5.

In previous work, we have annotated 314 Irish tunes with ground-truth on tune family membership, based on the academic literature and expert judgement [18]. The ground-truth data is available[3].

1.3. Previous work on music-related ontologies and KGs

In the literature, there are several ontologies that focus mainly on the modeling of music scores and cataloging information. We have summarized those in Table 1.

For example, the Music Ontology [34] enumerates the concepts and properties for describing music (i.e., artists, albums, and tracks), while the DOREMUS Ontology [2] focuses on describing catalog information. There are some other ontologies as well, e.g., Music Theory Ontology (MTO) [35] and Music Score Ontology (Music OWL) [28]. The MTO describes theoretical concepts of music composition, while Music-OWL models similar concepts with a focus on music sheet notation. Apart from these, Music Notation Ontology focuses on the core "semantic" information present in a score and also establishes a relationship between a symbolic and audio representation [33].

The efforts described above have focused on cataloging and music score information, whereas patterns in musical objects are *annotations*. Apart from patterns, other typical annotations might contain information about instruments and chords detected during certain time intervals in the music. For example, an annotation could represent that the chord "C:major" is heard during the time interval 10s-20s, detected with confidence 99% [13]. The MIR community uses JAMS: A JSON Annotated Music Specification for Reproducible MIR Research [27] as a software specification for music annotations.

The previous research in MIR mentioned is mainly focusing on formalizing and cataloging music content and metadata. However, recently the Music Annotation Pattern – An Ontology Design Pattern (MAP-ODP) [13] was proposed aiming to homogenise different annotation systems and to represent several types of musical objects (e.g., chords, patterns, structures). Furthermore, a software pipeline was developed with the help of SPARQLAnything [4] to process a database of annotations in JAMS format and automatically populate a corresponding KG. Ontology Design Patterns (ODPs) are reusable solutions to common modeling problems that may arise when building an ontology. They provide guidelines for creating a well-structured ontology and ensuring that it is consistent, interoperable, and maintainable. To avoid confusion, we will use the term "design pattern" to refer to an ontology design pattern, and just "pattern" to refer to a pattern in musical content.

The pattern found in the musical content is also a kind of annotation and thus it was ideal to use the MAP design pattern to model patterns and eventually create a KG. Thus, we developed our model on the basis of MAP-ODP to conceptualise *n*-gram pitch class patterns. Finally, we developed a software pipeline to populate a KG of n-gram patterns that automatically creates a KG from the patterns found in the tunes.

[3] https://github.com/polifonia-project/folk_ngram_analysis/tree/master/tune_family_annotations

Table 1
List of ontologies for musical data modeling in previous work.

Sr. No	Ontology	Description
1.	Music Ontology [34] http://purl.org/ontology/mo/	The purpose of Music Ontology is to describe music (i.e. artists, albums, and tracks). It defines a set of classes, properties, and relationships that can be used to describe various aspects of music, including musical works, performers, events, and genres.
2.	Tonality Ontology [23] http://purl.org/ontology/tonality/	The purpose of this ontology is to define a set of concepts and relationships that can be used to describe tonal structures, such as key intervals, interval class, and scales.
3.	Segment Ontology [24] http://www.linkedmusic.org/ontologies/segment	Segment Ontology is a formal representation of music that focuses on the segmentation of musical structures into smaller, meaningful parts. It defines a set of concepts and relationships that can be used to describe different types of music segments, such as phrases, sections, and motifs.
4.	Music Score Ontology [28] http://linkeddata.uni-muenster.de/ontology/musicscore	This ontology defines a set of concepts and relationships that can be used to represent different aspects of music scores, including staff notation, musical symbols, and temporal relationships between them.
5.	Music Theory Ontology [35] http://purl.org/ontology/mto/	This ontology is related to previous music ontologies such as the Music Ontology and the Music Score Ontology. It extends these ontologies by focusing specifically on music theory concepts and relationships. Some of the classes and properties of this ontology are key, chord, meter, cadence, etc.
6.	Music Notation Ontology [36] http://cedric.cnam.fr/isid/ontologies/MusicNote.owl	It defines a set of concepts and relationships that can be used to represent different aspects of music notation, including chord, score, note, measure, and music event. It also defines some object and data properties such as *has event*, *has measure*, *has note*, *has part*, *has syllable*, *has composer*, *has count*, *has pitch*, *has title*, etc.
7.	DOREMUS [2] http://data.doremus.org/ontology	The DOREMUS (DOing REusable MUSical data) extends the previous models CIDOC-CRM and FRBRoo for representing bibliographic information and adapting it to the domain of music. It uses several shared vocabularies about music-specific concepts (such as musical genres or keys) and is linked and published using the SKOS standard. It defines properties such as *has rhythmic pattern* but it does not serve our purpose to represent n-gram patterns.

1.4. Summary of the work

Our previous work in the Polifonia project developed ontologies and KGs for musical metadata [9, 12, 17]. They represent concepts such as composition name, family (applicable to "tune families" which occur in traditional music), composition type (e.g. jig or reel, again applicable to traditional music), key signature, and the name of the corpus from which the composition is drawn [4]. On top of these, we developed an ontology for musical patterns, including concepts such as the musical composition where the pattern is found, frequency and locations of pattern occurrences, composition duration, pattern contents, pattern type (i.e. the specific definition of the pattern in use), and the complete composition contents. The latter is to facilitate pattern tracing within a composition. We will describe our ontology and KG in detail in Section 2.

We developed a custom software pipeline, based on previous work by project partners [14], to produce the KG given the ontology and a pattern dataset. This pipeline includes intermediate processes which will be described in Section 3. Finally, we developed a UI for exploration of the patterns KG. This is described in Section 5.

2. Proposed Ontology and Knowledge Graph

In this section, we present the proposed ontology and relevant details related to the requirements for modeling patterns, as outlined in the previous sections. We provide information about the software components that were developed to populate the KG.

[4] https://github.com/polifonia-project/tunes-knowledge-graph

2.1. Building on the Smashub workflow

Our work follows an existing workflow, the *Smashub*[5] workflow. Smashub uses an ontology for annotations in a musical score or recording, and this ontology is part of the Polifonia Ontology Network (PON)[6]. The PON is a set of OWL ontology modules that describe the content and context of tangible and intangible musical cultural heritage assets across Europe [17]. Smashub takes advantage of a common format known as JAMS to represent these JAMS annotations. They are processed by SPARQL Anything with a custom query to generate a KG in RDF format. The first instance of the Smashub workflow is *ChoCo*, a chord corpus KG created by integrating and standardizing 18 existing chord collections [14]. The other existing example is *Harmonic Memory* [15], a KG of harmonic patterns.

To adapt the Smashub workflow to the case of melodic patterns, we created a custom JAMS pattern schema and SPARQL query, which will be described in Section 3.

2.2. Pattern Ontology – Music Annotation Framework

The purpose of the MAP-ODP is to model different types of musical annotation such as chords, structure, and patterns [13]. The MAP-ODP uses JAMS terminology because of its wide adoption in the MIR community, and thus the classes start with the name `jams`. The proposed pattern ontology uses MAP-ODP to model pattern information in a score. The ontology is shown in Figure 1.

There are two classes defined, named `jams:JAMSObservation` and `jams:JAMSAnnotation`. The `jams:JAMSAnnotation` class includes multiple observations. The main concept expressing patterns is the `jams:ScorePatternOccurrence` class. It specifies the occurrence of a pattern at a specific time in a composition. It has a property `jams:hasLocation` and to represent the location of a pattern.

The actual content (i.e., scale degree values) is described by a separate class, namely `jams:Pattern`. The same pattern may occur in multiple locations in a tune and in multiple tunes. This linkage enables the connection of different compositions within a corpus or even across multiple corpora. Several properties are defined for `jams:Pattern`, such as `xyz:pattern_complexity`, `xyz:pattern_length`, `xyz:pattern_content`, and `xyz:pattern_type`. The `xyz:pattern_type` property is a string. It is not possible to anticipate all future patterns types, so a string enables flexibility. Our KG contains patterns with `type diatonic scale degree`, `level=accent` and `n_vals`, together indicating that we represent diatonic notes, restricting to notes on strong (accented) beats, and showing the number of values in the particular pattern. The `xyz:pattern_complexity` property defines how complex a pattern is, which is the fraction of unique pitch values in a pattern and the length of the pitch. The current ontology version only shows the concepts and properties associated with patterns. In addition to this, various metadata information of the musical composition is also modeled, including `jams:key`, `jams:timeSignature`, `jams:transcriber`, `jams:tuneContent`, and `jams:tuneFamily`, etc.

2.3. Knowledge Graph

Figure 2 shows an illustrative extract from the KG, with important concepts and relationships. As shown, the KG contains musical compositions with metadata such as titles. Each composition may have annotations, in particular annotations of pattern occurrences. An occurrence of a pattern is a location in time in a particular composition where a pattern occurs. A pattern has contents.

3. Patterns2KG: JAMS Pipeline

The Patterns2KG[7] pipeline consists of several stages as shown in Figure 3. First, pattern sequences are extracted from the dataset using FoNN tools. Details are discussed in Section 3.2. The dataset includes composition metadata,

[5] https://github.com/smashub/choco
[6] https://github.com/polifonia-project/ontology-network
[7] https://github.com/polifonia-project/patterns-knowledge-graph

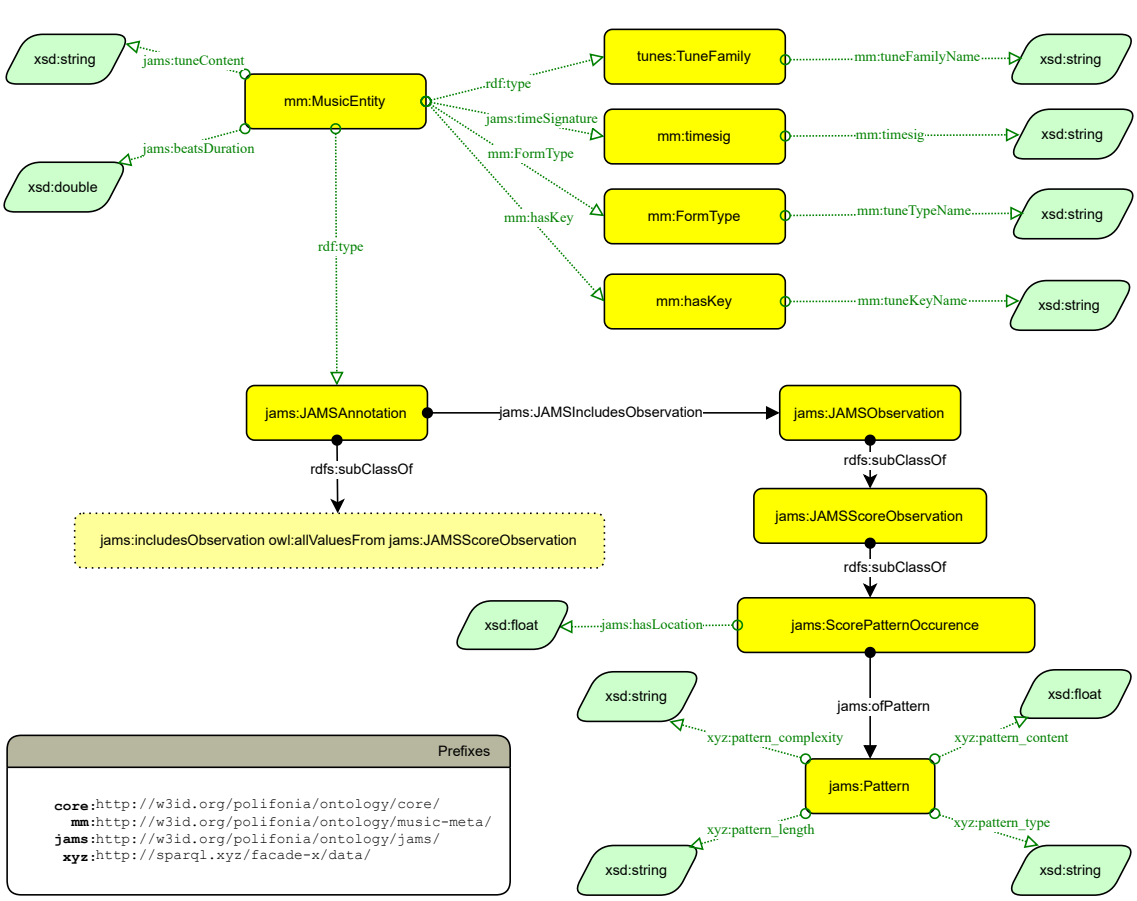

Fig. 1. The pattern ontology: relevant concepts of melody and patterns used to generate the KG.

and details can be found in Section 3.1. Next, the Patterns2KG pipeline processes the pattern sequences to generate a `.jams` file for each composition. Patterns2KG uses a specialized schema to represent patterns, which is described in Section 3.3. Next, Patterns2KG loads each `.jams` file and converts it to RDF. This is accomplished through a SPARQL query that automatically loads the JAMS file and uses SPARQL Anything [4] to generate the KG. A snippet of the KG is depicted in Figure 2. This process is repeated for all JAMS files, resulting in the creation of the KG for the entire dataset.

3.1. Dataset

We begin by specifying the dataset we used. In our previous work [19], we conducted a comprehensive pattern analysis on a large dataset of Irish folk music, known as *The Session*[8][29]. This dataset is crowd-sourced but is widely regarded as a definitive resource by practitioners. The dataset contains 40,152 compositions (as of our extract date), in ABC notation, a symbolic music format widely used by practitioners and students of Western folk music. A further source of tunes is the *Essen* corpus[9], which contains music from a wider variety of traditions.

[8]https://thesession.org/
[9]http://essen.themefinder.org/

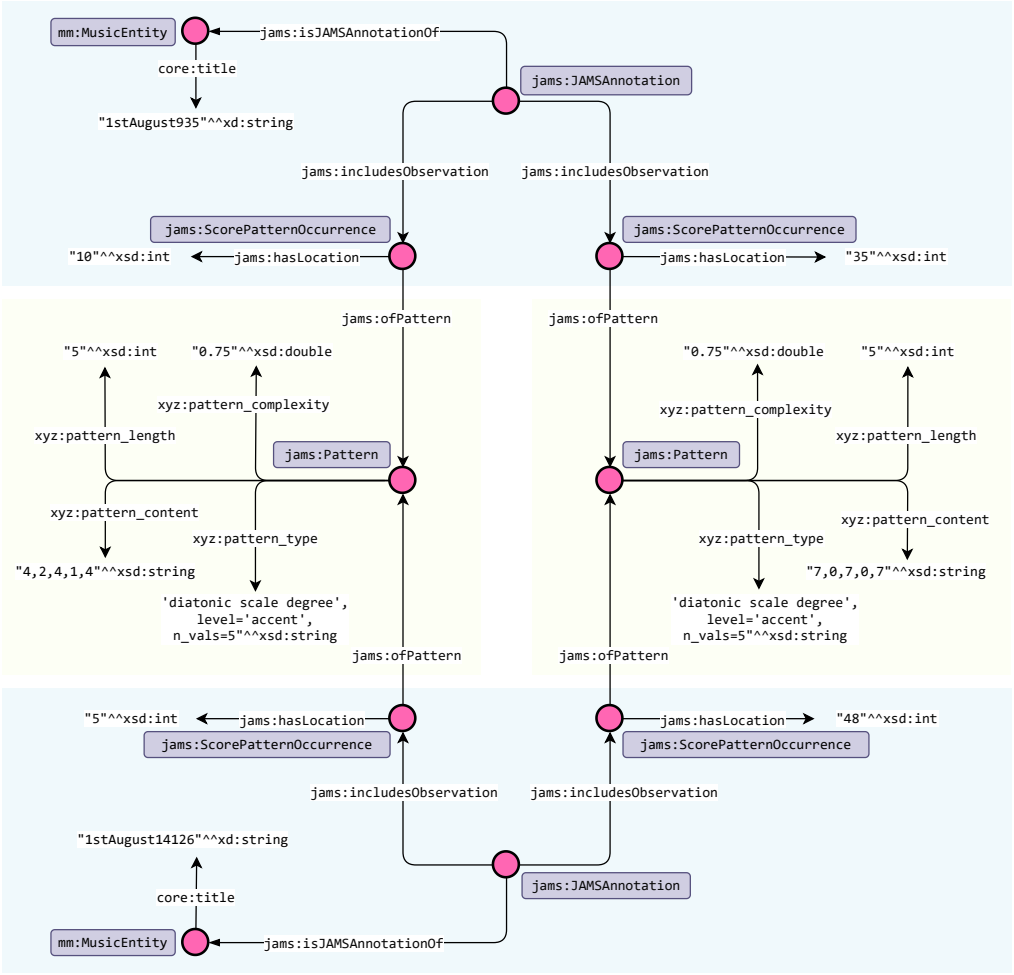

Fig. 2. Extract from the pattern KG. One composition (top) can be related to another (bottom) via occurrences of the same patterns (middle). The yellow boxes show information about patterns themselves, while blue boxes concern patterns' usage and location in tunes.

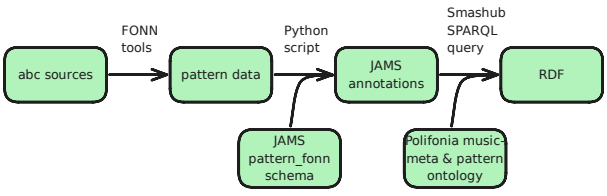

Fig. 3. Workflow of the proposed system

3.2. *Extracting patterns*

The FoNN tools take inputs from *The Session* in ABC format and, for each composition, create sequences representing pitch, onset, duration, and velocity values for each note using the `music21` Python library [11]. These *primary feature sequences* are extracted both at note-level (i.e., for every note) and at accent-level (i.e., for rhythmically accented notes only). Sequences of additional *secondary features* are derived from the primary sequences,

including the diatonic scale degree data used in this work. For *Essen*, tune data is available in integer representations through the MTCFeatures library[10].

The pattern extraction is then implemented on the diatonic scale degree sequences using *n*-grams. Only those *n*-gram patterns occurring more than once in the corpus are retained. Pattern occurrences are counted per composition, ranked by tf–idf [37], stored in a single corpus-level Pandas dataframe and then processed to a Python `pickle` format. Thus for both note-level and accent-level we have a large dataframe of pattern occurrence tf-idf values representing the entire corpus. The code of the FoNN tools is available on GitHub[11].

3.3. JAMS Pattern Schema

The MIR community uses JAMS: A JSON Annotated Music Specification for Reproducible MIR Research [27] as a software specification for music annotations.

The JAMS annotation offers a default pattern schema named `pattern_jku`; however, the default schema is inadequate to model our requirements. For instance, it consists of fields such as `pattern_id`, `midi_pitch`, `occurrence_id`, and `staff` whereas we require to define types of patterns, lengths of patterns, and pattern contents so that we can preserve them in KG in the later stages. Therefore, a custom JAMS schema was designed, suitable for patterns extracted by FoNN tools. This schema is named `pattern_fonn`, with components including the pattern contents and location, as shown below.

```
{"pattern_fonn":
    {"value": {
            "type": "object",
            "properties": {
                "pattern_id": {"type": "string"},
                "pattern_content": {"type": "string"},
                "pattern_type": {"type": "string"},
                "pattern_frequency": {"type": "number"},
                "pattern_length": {"type": "number"},
                "pattern_location": {"type": "number"}
            },
            "required": ["pattern_id", "pattern_content", "pattern_type",
                         "pattern_frequency", "pattern_length"]
        },
        "dense": false,
        "description": "FoNN-Patterns"}}
```

Using custom text-processing Python scripts, the pattern database was converted to `.jams` files that follow this schema. The content is organised in two main sections, one for pattern information and the other for file metadata. Each composition leads to one `.jams` file, containing a list of annotations under `data`. Each annotation contains generic JAMS fields (suitable for any JAMS annotation), such as time and duration; and also contains `pattern_fonn`-specific fields under `value`. In the `file_metadata` section the `identifiers` link to an online reference source.

3.4. Knowledge Graph – RDF generation using SPARQL Anything

In the next stage, each `.jams` file is converted into a KG in the form of an RDF file. This is done using a custom SPARQL query[12] This is a dynamically generated SPARQL construct query that transforms the JAMS file into RDF graph statements.

[10] https://github.com/pvankranenburg/MTCFeatures
[11] https://github.com/polifonia-project/folk_ngram_analysis
[12] Thanks to Andrea Poltronieri and Polifonia partners for help with this.

Table 2
List of competency questions developed in conjunction with musicologists

No.	Question
CQ1	Metadata: Find composition metadata such as key signature, composition type (e.g. reel or jig), and name of the transcriber.
CQ2	Pattern types: Identify the types of patterns present, e.g. a pattern might be composed of a list of notes, or of accented notes, pitch-class values, etc.
CQ3	Pattern search: Given a pattern, find a list of compositions it occurs in.
CQ4	Pattern search (2): Given two patterns, find a list of compositions both occur in.
CQ5	Pattern frequency: Retrieve patterns and their frequencies per tune.
CQ6	Pattern location: Given a pattern, retrieve its location in a tune (beginning, middle, or end).
CQ7	Similar compositions: Given a composition, find a ranked list of similar compositions (based on pattern similarity).
CQ8	Characteristic patterns: Given a tune family, tune type (e.g., reel), or national origin, find the patterns that are characteristic of that subset of compositions.
CQ9	Pattern containment: Given a pattern, find all compositions when it or a pattern that contains that pattern occurs.

```
PREFIX jams:<http://w3id.org/polifonia/ontology/jams/>
PREFIX mm:<http://w3id.org/polifonia/ontology/music-meta/>
PREFIX core:<http://w3id.org/polifonia/ontology/core/>
PREFIX xyz:<http://sparql.xyz/facade-x/data/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tunes:<http://w3id.org/polifonia/ontology/tunes/>
PREFIX prov: <http://www.w3.org/ns/prov#>
```

Listing 1: List of prefixes required for executing SPARQL queries.

Two details of the SPARQL query are worth noting, as follows.

First, to create a unique URI for representing a `jams:JAMSAnnotation` is created by applying the `SHA1` hash function to the combination of the `TuneId` and `TuneTitle` to produce the latter part of the URI. This approach ensures interoperability with the other Polifonia Ontology Network (PON) modules.

Second, each `jams:Pattern` is also represented by a hashed URI. This URI means that multiple occurrences of the same pattern point to the same pattern object, as shown in Figure 2. It also shows how patterns of the same content are linked together to create an aggregated view of the whole KG. It is also worth mentioning that a composition can belong to any corpus and thus compositions of different corpora can be conveniently linked together, and thus it offers an opportunity to perform inter-corpus pattern analysis. Furthermore, it is scalable in the sense that further corpora can be automatically added to enrich the KG without making any changes to the existing model. It is also highly flexible: third parties can introduce novel pattern types (e.g., not based on *n*-grams) using `xyz:pattern_type`.

A total of approximately 45 million (44,979,281) statements were generated. The process takes time linear in the number of compositions – several hours for our complete KG. In the version we host for public use, we store just the tune-family annotated subset of *The Session* together with all of *Essen*, totalling 8784 tunes. It is available via Blazegraph[13] with a SPARQL endpoint[14].

4. Evaluation

We used the eXtreme Design methodology, a well-known method for the evaluation of ontologies. According to this, an Ontology shall address a set of competency questions (CQs). Later the CQs are translated to SPARQL

[13]https://blazegraph.com/
[14]https://polifonia.disi.unibo.it/fonn/sparql

queries for extracting the modeled knowledge to ensure the retrieved result serves its purpose. The CQs play a frame of reference role for evaluating ontologies [25]. The CQs link to tasks the user might wish to do. Showing that our system answers the CQs helps us to evaluate the scope of the domain modeling. Therefore, the CQs should be developed in collaboration with domain experts. In our research, we conducted several meetings with musicologists to understand their needs. Some of those who took part in these meetings were members of the Polifonia project, so they understood the context, while others were independent. To save space, we present just four CQs, listed in 2. The full list is available online.

In order to address these CQs, we formulated SPARQL queries as shown below. They demonstrate the suitability of our proposed model in capturing pattern-related information of use to musicologists.

Later, a comprehensive analysis of each question was conducted, and below are the responses and actions. Essential namespace prefixes are given in Listing 1 for use in later Listings. These queries can be executed using the SPARQL endpoint already mentioned.

```
SELECT DISTINCT ?Title ?TuneID ?Duration ?Timesig ?Key ?Form
WHERE {
  VALUES ?Title {"Drowsy Maggie"}
  ?Tune rdfs:label ?Title.
  ?Tune core:id ?TuneID.
  ?Tune mm:hasKey ?KeyURI;
        jams:timeSignature ?TimesigURI;
        jams:beatsDuration ?Duration;
        mm:hasFormType ?FormURI.
  ?FormURI mm:tuneTypeName ?Form.
  ?TimesigURI mm:timesig ?Timesig.
  ?KeyURI mm:tuneKeyName ?Key.
} ORDER BY ?Title ?TuneID LIMIT 4
```

Title	TuneID	Duration	Timesig	Key	Form
Drowsy Maggie	12406	66.0	4/4	Edorian	reel
Drowsy Maggie	12407	18.0	4/4	Edorian	reel
Drowsy Maggie	12408	50.0	4/4	Dmajor	reel
Drowsy Maggie	12409	34.0	4/4	Edorian	reel

Listing 2: Finding metadata information on a composition or corpus: query (top) and results (bottom). Notice that providing a title will retrieve multiple variants, which have the same title but distinct IDs.

For many compositions in `The Session` corpus, we have multiple related variants that share a common title but are differentiated via unique composition ID numbers. Therefore, various compositions may be retrieved if a search is performed through the composition title. In this case, notice that we have provided a composition title instead of composition Id, therefore, multiple compositions are retrieved as shown in Listing 2.

In the requirement outlined in 3, the user is interested in loading various types of patterns modeled in KG. The results show that, at present, the KG contains diatonic scale degree patterns of varying lengths.

The requirements listed in 4 represent a common use case, where a user is interested in finding the list of compositions in which a given pattern occurs. This requirement is addressed using the SPARQL query provided in 4. The results also include additional relevant information for the user, such as Tune Type, Key, and Time Signature.

The requirement defined in 6 reflects a scenario where a user is interested in identifying all compositions in which two specific patterns co-occur. This need is fulfilled by the SPARQL query shown in the upper part of 5. The query matches both patterns against the KG and retrieves only those tunes where both patterns are present. The result table, shown below the query, includes the tune ID, title, and the frequency of each pattern within the matched compositions.

```
SELECT ?PatternType (COUNT(?PatternType) AS ?Occurrences)
WHERE {
    ?Pattern xyz:pattern_type ?PatternType.
}
GROUP BY ?PatternType
ORDER BY DESC(?Occurrences) LIMIT 10
```

PatternType	Occurrences
feature='diatonic scale degree', level='accent', n_vals=6	35612
feature='diatonic scale degree', level='accent', n_vals=5	13721
feature='diatonic scale degree', level='accent', n_vals=4	3821

Listing 3: Find all pattern types in the KG and total number of each pattern

```
SELECT DISTINCT ?Title ?TuneID ?Timesig ?Key ?Form
WHERE{
    ?Observation jams:ofPattern ?PatternURI.
    ?PatternURI xyz:pattern_content "5, 1, 6, 2, 4, 1".
    ?Annotation jams:includesObservation ?Observation.
    ?Annotation jams:isJAMSAnnotationOf ?Tune.
    ?Tune rdf:type mm:MusicEntity.
    ?Tune core:id ?TuneID.
        ?Tune rdfs:label ?Title.
        ?Tune mm:hasKey ?KeyURI;
        jams:timeSignature ?TimesigURI;
        jams:beatsDuration ?Duration;
        mm:hasFormType ?FormURI.
        ?FormURI mm:tuneTypeName ?Form.
        ?TimesigURI mm:timesig ?Timesig.
        ?KeyURI mm:tuneKeyName ?Key.
} ORDER BY ?Title ?TuneID LIMIT 4
```

Title	TuneID	Timesig	Key	Form
The Road to Lisdoonvarna	12972	12/8	Edorian	slide
The Road to Lisdoonvarna	21776	12/8	Edorian	slide
The Road to Lisdoonvarna	250	12/8	Edorian	slide
The Road to Lisdoonvarna	27843	12/8	Edorian	slide

Listing 4: Finding compositions where a given pattern was found: query (top) and results (bottom).

CQ5: Pattern frequency: Retrieve patterns and their frequencies per tune. Here a user is interested in finding common patterns in a given tune.

To address this requirement, the query in 6 retrieves patterns occurring in a specific composition by explicitly filtering results for a given tune title. While the current query focuses on a single composition, it can be easily generalized to handle multiple tunes. Since the property `jams:ofPattern` links observations to a pattern URI, identical patterns occurring in different compositions will still reference the same pattern entity. To compute how frequently each pattern appears within the selected tune, we apply a `GROUP BY` clause on the tune title and pattern content. The results, shown below the query, highlight the most frequently occurring patterns in the tune "Drowsy Maggie."

CQ6: Pattern location: Given a pattern, retrieve its location in a tune (beginning, middle, or end). To address this

```
SELECT ?Title ?TuneID (COUNT(DISTINCT ?Observation1) AS ?Pattern1Freq)
 ↪  (COUNT(DISTINCT ?Observation2) AS ?Pattern2Freq)
WHERE {
  ?Pattern1 xyz:pattern_content "5, 1, 3, 1" .
  ?Pattern2 xyz:pattern_content "5, 3, 1, 1, 5, 1" .
  ?Observation1 jams:ofPattern ?Pattern1 .
  ?Observation2 jams:ofPattern ?Pattern2 .
  ?Annotation1 jams:includesObservation ?Observation1 .
  ?Annotation2 jams:includesObservation ?Observation2 .
  ?Annotation1 jams:isJAMSAnnotationOf ?TuneID .
  ?Annotation2 jams:isJAMSAnnotationOf ?TuneID .
  OPTIONAL { ?TuneID rdfs:label ?Title }
}
GROUP BY ?TuneID ?Title
HAVING (COUNT(DISTINCT ?Observation1) > 0 && COUNT(DISTINCT ?Observation2)
 ↪  > 0)
ORDER BY DESC(?Pattern1Freq) DESC(?Pattern2Freq)
```

Title	TuneID	Pattern1Freq	Pattern2Freq
Blackbird, The	39782	3	1
Blackbird, The	27198	2	2

Listing 5: Given two patterns, find a list of compositions both occur in. Query (top) and results (bottom).

```
SELECT ?Title ?PatternContent (COUNT(?Pattern) AS ?PatternFreq)
WHERE {
  VALUES ?Title {"Drowsy Maggie"}
  ?Tune rdfs:label ?Title .
  ?JamsFile jams:isJAMSAnnotationOf ?Tune .
  ?JamsFile jams:includesObservation ?Observation .
  ?Observation jams:ofPattern ?Pattern .
  ?Pattern xyz:pattern_content ?PatternContent.
}
GROUP BY ?Title ?PatternContent
ORDER BY DESC(?PatternFreq) LIMIT 5
```

Title	PatternContent	PatternFreq
Drowsy Maggie	1, 1, 7, 1	41
Drowsy Maggie	6, 1, 1, 7	41
Drowsy Maggie	6, 1, 1, 7, 1	41
Drowsy Maggie	2, 1, 6, 1	34
Drowsy Maggie	5, 2, 1, 6	32

Listing 6: Retrieve the patterns and their frequencies per tune. Here, a user is interested in finding common patterns in a given tune. Query (top) and results (bottom).

type of requirement, we include the total length of the tune as "BeatDuration" and also incorporate pattern location information alongside the pattern. Therefore, the query listed in 7 can cater to this need.

CQ7: - Similar compositions: Given a composition, find a ranked list of similar compositions (based on pattern similarity).

```
SELECT DISTINCT ?Title ?TuneID ?Location ?BeatDuration ?PositionCategory
WHERE {
  ?Pattern xyz:pattern_content "5, 1, 6, 2, 1, 5" .
  ?Observation jams:ofPattern ?Pattern ;
               jams:hasLocation ?Location .
  ?JamsFile jams:includesObservation ?Observation ;
            jams:isJAMSAnnotationOf ?Tune .

  ?Tune rdf:type mm:MusicEntity ;
        core:id ?TuneID ;
        jams:beatsDuration ?BeatDuration .
  OPTIONAL { ?Tune rdfs:label ?Title }
  BIND( IF( ?Location <= (?BeatDuration * 0.25), "Start",
        IF( ?Location >= (?BeatDuration * 0.75), "End", "Middle")
    ) AS ?PositionCategory)
} ORDER BY ?Title ?Location LIMIT 10
```

Title	TuneID	Location	BeatDuration	PositionCategory
Foxhunters, The	22807	60.0	82.0	Middle
Foxhunters, The	22808	60.0	82.0	Middle
Foxhunters, The	30575	60.0	82.0	Middle
O'Sullivan's March	2204	85.0	115.0	Middle
O'Sullivan's March	2204	101.0	11.0	End

Listing 7: Where does a pattern (5, 1, 6, 2, 1, 5) appear in a tune? For instance, it could be found at the beginning, middle, or end of a tune. Query (top) and results (bottom).

In Listing 8, the query is designed to retrieve a ranked list of compositions that share common patterns with a given tune. This is achieved by exploiting the fact that observations of patterns (`?obs1` and `?obs2`) point to the same pattern URI when the content is identical. As a result, if two compositions share the same pattern, their observations will reference the same `?SharedPattern` URI. This enables us to compute the intersection of patterns across compositions.

To avoid returning the given tune itself or introducing symmetric duplicates (e.g., A–B and B–A), filters are applied to ensure that only distinct and ordered tune pairs are considered. The results are grouped by both tune IDs and their corresponding labels, and the count of shared patterns is used to rank similarity.

Although the query in 8 is intended to find closely related versions of a tune, the same structure can be adapted to identify other similar tunes that are not direct variations. These compositions may share a moderate number of patterns due to stylistic or structural resemblance. By adjusting the sort order (e.g., using `ORDER BY ASC`) and applying a `HAVING` clause (e.g., to select those with at least 10 shared patterns), we can expand the scope to retrieve loosely related yet musically similar compositions. This variant approach was used to generate the results shown in Listing 9.

CQ8: Given a tune family, tune type (e.g., reel), or national origin, find the patterns that are characteristic of that subset of compositions.

Note: We take the definition of the tune family from [6] which states that "A tune family is a group of melodies showing basic interrelation by means of constant melodic correspondence, and presumably owing their mutual likeness to descent from a single air that has assumed multiple forms through processes of variation, imitation, and assimilation." In the `The Session` dataset, most of the compositions do not have the tune family information, however, we can extract tune family information with a query listed in 10. In this query, we have added certain conditions to limit the overall results. Additionally, we can refine the results as per the specific needs of the users. The query presented in Listing 10 retrieves the most characteristic melodic patterns for a given family, in this case,

```
SELECT ?GivenTuneTitle ?GivenTuneID ?SimilarTuneTitle ?SimilarTuneID
 ↪  (COUNT(DISTINCT ?SharedPattern) AS ?SharedPatternCount)
WHERE {
  VALUES ?GivenTuneTitle {"Rakes Of Kildare, The"}
  ?GivenTune rdfs:label ?GivenTuneTitle ;
             core:id ?GivenTuneID .
  ?JamsFile1 jams:isJAMSAnnotationOf ?GivenTune ;
             jams:includesObservation ?Observation1 .
  ?Observation1 jams:ofPattern ?SharedPattern .
  ?JamsFile2 jams:includesObservation ?Observation2 ;
             jams:isJAMSAnnotationOf ?SimilarTune .
  ?Observation2 jams:ofPattern ?SharedPattern .
  ?SimilarTune rdf:type mm:MusicEntity ;
               core:id ?SimilarTuneID ;
               rdfs:label ?SimilarTuneTitle .
  FILTER(?SimilarTune != ?GivenTune)
  FILTER(STR(?GivenTuneID) < STR(?SimilarTuneID))
}
GROUP BY ?GivenTuneTitle ?GivenTuneID ?SimilarTuneTitle ?SimilarTuneID
ORDER BY DESC(?SharedPatternCount) LIMIT 5
```

GivenTuneTitle	GivenTuneID	SimilarTuneTitle	SimilarTuneID	SharedPatternCount
Rakes Of Kildare, The	34240	Rakes Of Kildare, The	84	89
Rakes Of Kildare, The	12586	Rakes Of Kildare, The	12587	83
Rakes Of Kildare, The	12588	Rakes Of Kildare, The	34240	63
Rakes Of Kildare, The	12588	Rakes Of Kildare, The	84	63
Rakes Of Kildare, The	12589	Rakes Of Kildare, The	34240	58

Listing 8: Given a composition, find a ranked list of similar compositions (based on pattern similarity): query (top) and results (bottom).

GivenTuneTitle	GivenTuneID	SimilarTuneTitle	SimilarTuneID	SharedPatternCount
Rakes Of Kildare, The	22230	Queen Of The Earth, Child Of The Skies	27066	11
Rakes Of Kildare, The	12588	Johnny Cope	34131	12
Rakes Of Kildare, The	12590	Johnny Cope	40691	12
Rakes Of Kildare, The	22229	Drunken Sailor's, The	32089	12
Rakes Of Kildare, The	22215	Long Note, The	9850	12

Listing 9: Example 2 – This result is a variant generated using the query in 8, enhanced with a `HAVING` clause and sorted using `ORDER BY ASC` to retrieve compositions that are not direct variations but exhibit close relevance based on pattern similarity.

Lord McDonald's. The query filters for patterns with a minimum length of six notes. However, we can easily add further filters to retrieve more interesting results.

CQ9: Can we relate one pattern to another? For example, pattern similarity, containment (one pattern is contained by another pattern), transposition, etc. This user requirement pertains to pattern relationships.

Given a pattern, find all compositions when it or a pattern that contains that pattern occurs. The query and results for this CQ are listed in 11. We store n-gram patterns for multiple values of n. Consequently, we can use this query to identify compositions where the specified pattern is a part of another pattern. This represents a containment relationship.

```
SELECT ?PatternContent (COUNT(?Pattern) AS ?PatternFrequency)
WHERE {
?Pattern xyz:pattern_content ?PatternContent ;
       xyz:pattern_length ?patternLength .
FILTER(?patternLength >= 6)
?Observation jams:ofPattern ?Pattern .
?TuneFile jams:includesObservation ?Observation ;
          jams:isJAMSAnnotationOf ?Tune .
?Tune core:isMemberOf
↪  <http://w3id.org/polifonia/resource/tunefamily/Lord_McDonald_s> .
}
GROUP BY ?PatternContent
ORDER BY DESC(?PatternFrequency) LIMIT 5
```

PatternContent	PatternFrequency
3, 3, 5, 5, 2, 3	22
5, 3, 5, 2, 5, 2	17
5, 2, 5, 3, 5, 2	15
5, 2, 5, 2, 5, 2	15
5, 3, 3, 5, 5, 2	15

Listing 10: Given a tune family, tune type (e.g., reel), or national origin, find the patterns that are characteristic of that subset of compositions: query (top) and results (bottom).

```
SELECT DISTINCT ?Title ?TuneID ?MatchedPatternContent ?Location
↪  ?BeatDuration
WHERE {
  VALUES ?targetContent {"5, 1, 6, 2"}
  ?Pattern xyz:pattern_content ?MatchedPatternContent .
  FILTER(CONTAINS(STR(?MatchedPatternContent), ?targetContent))
  ?Observation jams:ofPattern ?Pattern ;
               jams:hasLocation ?Location .
  ?JamsFile jams:includesObservation ?Observation ;
            jams:isJAMSAnnotationOf ?Tune .
  FILTER(CONTAINS(STR(?JamsFile), "thesession20211212"))
  ?Tune rdf:type mm:MusicEntity ;
        core:id ?TuneID ;
        jams:beatsDuration ?BeatDuration .
  OPTIONAL { ?Tune rdfs:label ?Title }
} ORDER BY ?Title ?Location LIMIT 5
```

Title	TuneID	MatchedPatternContent	Location	BeatDuration
Britches Full Of Stitches, The	31614	5, 1, 6, 2	14.0	33.0
Britches Full Of Stitches, The	31614	5, 1, 6, 2, 5	14.0	33.0
Foxhunters, The	30575	5, 6, 5, 1, 6, 2	50.0	82.0
Foxhunters, The	13438	5, 6, 5, 1, 6, 2	50.0	82.0
Foxhunters, The	22808	5, 6, 5, 1, 6, 2	50.0	82.0

Listing 11: List of compositions and patterns where a given pattern ("5, 1, 6, 2") is subsumed by another pattern.

In conclusion, we have successfully addressed all of the competency questions outlined in this study, ranging from basic metadata retrieval (CQ1) to more complex pattern-based queries such as similarity (CQ7), characteristic motif discovery (CQ8), and pattern containment (CQ9). Each query demonstrates how the underlying knowledge graph enables expressive and musically relevant interrogation. While these competency questions cover a wide spectrum of analytical needs, we acknowledge that many additional requirements may emerge from diverse research or creative scenarios. To accommodate such needs, we have compiled an extended collection of SPARQL queries, which are publicly available in our GitHub repository[15].

5. Graphical User Interface

In this section we describe the GUI we have developed to take advantage of the Patterns KG. We briefly discuss related work; our co-design process with musicologists; two forms of evaluation and feedback from potential useres; and the final version of the GUI in operation.

5.1. Existing GUIs for music corpus exploration

Many GUIs have been developed for the exploration of music libraries in a popular music context, but we focus on more experimental GUIs for musicological and content-based exploration.

Tovstogan et al. used a GUI for the exploration and discovery of personal music collections in which labelled segments of musical compositions are represented as points in a 2D visualisation. Dimension reduction algorithms were used to position points in the visualisations and suggest similarity. They found that users perceived this system as engaging, rewarding and useful [42]. Knees et al. developed a three-dimensional GUI for the exploration of musical collections, in which similar compositions are clustered together, and their properties processed into a height value to form a landscape through which the user can navigate a music collection [30]. The *Digital Music Lab* system of Abdallah et al. supports large-scale musicological research across large music corpora. Their system uses Linked Open Data, allowing for a distributed system across multiple corpora. It includes a GUI through which analyses of musical collections can be visualised as bar charts, line graphs or histograms [1]. De Berardinis et al. [16] developed a "harmonic relation graph" to show inferred relationships between pairs of pieces of music in a network diagram. Graph layout algorithms produce a static graph that visualises clusters of related pieces[16].

Focussing now on folk music, Walshaw developed the abcnotation.com web interface to search an online corpus of traditional music stored in ABC notation [44–46]. The interface provides tools to view tunes in musical notation and play MIDI audio. The interface also features the *TuneGraph* network visualisation, which presents tunes as nodes with connections to a small set of similar tunes. The *TunePal* interface, https://tunepal.org, allows querying a corpus of traditional Irish, Welsh, Scottish and Breton music by title and by playing music. The score of the selected composition is displayed on a musical staff and in ABC notation. MIDI and recorded audio of tunes can be played [20–22]. Finally, we mention the *LOD Live* interface, designed for exploration of arbitrary KGs[17]. This UI is not music-oriented, but served as inspiration for a live network diagram with show / hide functionality based on linked open data.

5.2. Co-Design Process

Following our review of related work, we engaged in a co-design process with potential users. We used semi-structured interviews with musicologists and music researchers of varying backgrounds. They were driven by prototypes and related work, and then used brainstorming and open discussion. Taken together with our review of related work, we arrived at an initial list of desirable features for our GUI. This allowed us to develop a first complete version of the GUI. User testing with this was then carried out in two formats: in-depth testing with individuals,

[15] https://github.com/polifonia-project/patterns-knowledge-graph
[16] https://github.com/polifonia-project/harmonic-similarity
[17] E.g. http://en.lodlive.it/?https://w3id.org/italia/env/ld/place/municipality/00201_042002

who were required to carry out tasks; and a group tutorial, intended to lead to comments and questions. Results and feedback led to the refinement of the required features, which now included fuzzy search facilities over metadata and over contents, display of patterns on musical staves, audio playback of patterns, a facility to export a stable URL for citation of a particular result, code resources such as SPARQL queries on a Resources page, and a network diagram for exploration of relationships. Further details are out of scope here, but were described in previous work [41].

5.3. GUI Description

In this section, the final design of the application is described from the point of view of the user.

The interface opens onto a *Search* page which features three search methods for finding tunes, by fuzzy search over *Title*, search over *Pattern* (search allowing for a variety of input formats to facilitate the user), and *Advanced search* which allows for multiple-select drop-down boxes for 'Corpus', 'Key', 'Time Signature', and 'Tune Type', the options of which can be filtered using associated text fields.

Search results show relevant metadata such as tune type (e.g. jig, reel, strathspey), key, and time signature, as shown in Figure 4. Clicking on a search result opens the *Composition* page for the selected tune.

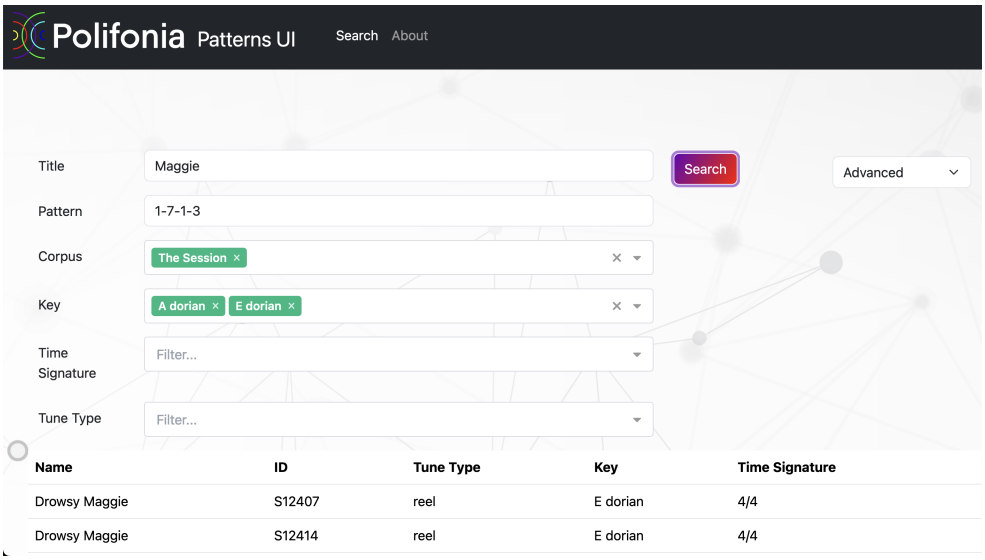

Fig. 4. The *Search* page showing the *Advanced* search interface. The *Title* field contains the partial title 'Maggie', which can match a number of tune titles as a result of the fuzzy *Title* search feature. The pattern '1, 3, 1, 7' has been entered in the *Pattern* field using hyphens as a delimiter. 'The Session' has been selected from the *Corpus* drop-down. The *Key* drop-down further limits the search to tunes featuring a Dorian key, and the *Time Signature* drop-down specifies a '4/4' time signature. The results of the search are shown in a table.

An example of a *Composition* page is shown in Figure 5. It includes an interactive network visualisation which can be expanded by the user to include more tunes and patterns. Double-clicking a tune node navigates to the clicked node's *Composition* page, while double-clicking a *Pattern* node navigates to this node's *Pattern* page. The *Composition* page also features a panel on the left side listing the most common melodic patterns contained in the selected tune along with their occurrence frequencies. Trivial patterns can be excluded from the list by adjusting a toggle switch. Trivial patterns are patterns with low pattern complexity, based on a proportion of unique note values.

The *Pattern* page features a musical stave representing the melodic pattern described by the page in musical notation, along with a MIDI player that can play audio of the pattern. Below the stave, there is a list of the tunes that contain the selected pattern, each entry linking to its respective *Tune* page. A screenshot of the *Pattern* page can be seen in Figure 6.

The *Tune Family* page for a given tune can be accessed from a link on its *Composition* page below the page title. The *Tune Family* page lists all the tunes in the selected tune family. Each tune in the list links to its respective *Composition* page. The *Composition* page also features a link to external resources for each tune.

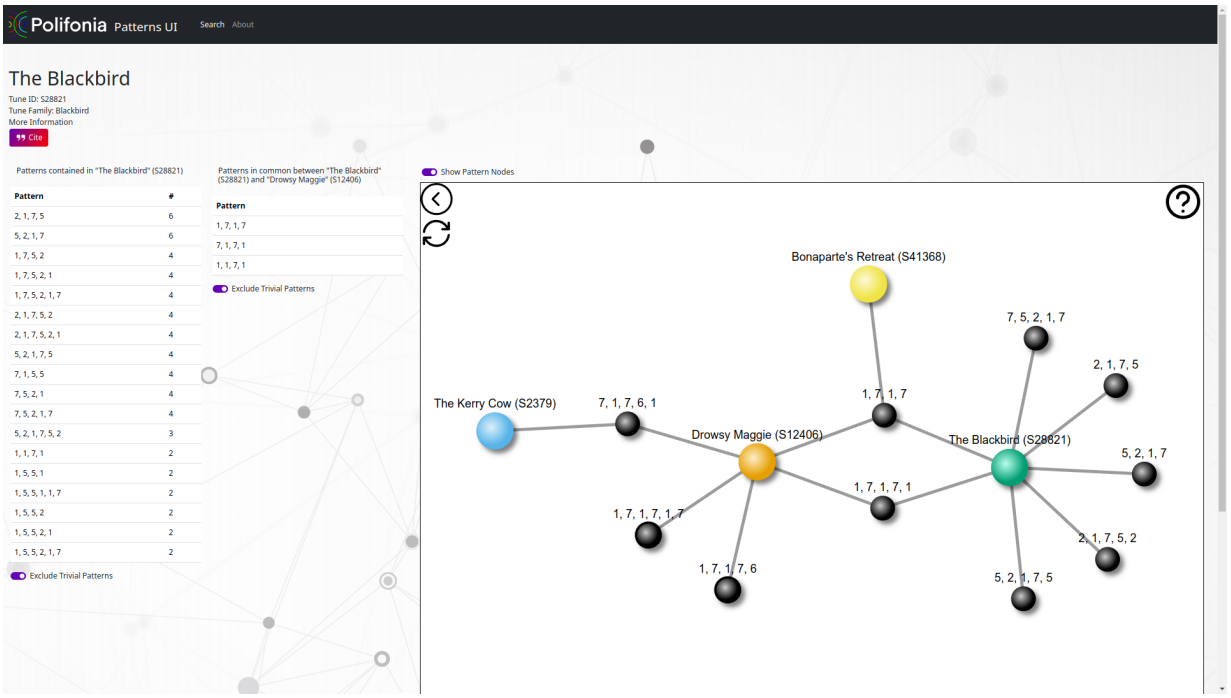

Fig. 5. Patterns UI *Composition* page. On the left, a table of the most common patterns in the selected tune; a middle panel containing a list of patterns in common with the previously selected tune; on the right, a network visualisation showing the current tune, some of its most common patterns (small, black nodes), and other tunes that contain the same pattern. At the top of the page, the tune title and tune family name are displayed above a link to the original source for the tune and a *Cite* button.

On several pages, there is a feature that generates a citation for the *current* page, which can be copied to the clipboard using the *Copy* button.

6. Conclusions

This study combines novel user interfaces and a structured KG for exploring musical patterns. The KG, constructed using a pattern ontology based on Musical Annotation patterns, enables seamless integration of multiple data sources, facilitating complex queries that uncover relationships among musical entities. We have processed a dataset of 40,152 compositions, resulting in a KG with approximately 45 million statements, evaluated using competency questions to demonstrate its analytical capabilities. Usability testing of Patterns UI confirmed its effectiveness, providing positive feedback and affirming its role in answering research queries.

This work has a wide range of applications. For instance, it could be used in recommendation systems and MIR, enabling musicians to analyze patterns, compare compositions across corpora, and extract meaningful insights from vast musical datasets. Furthermore, with the help of an interactive UI, non-technical users can engage with the system to explore trends, uncover similarities between compositions, and support creative or educational endeavors.

Acknowledgements

This work is part of the Polifonia Project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N. 101004746.

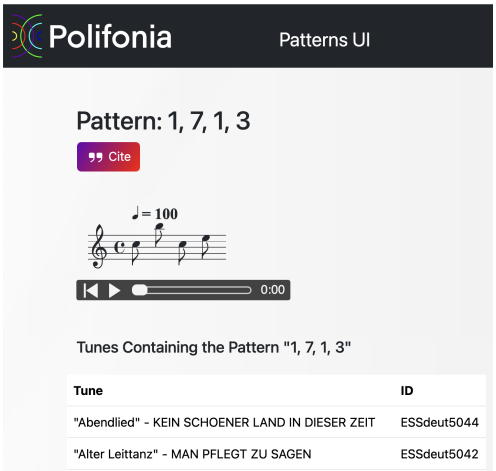

Fig. 6. The *Pattern* page shows a musical stave featuring the pattern in musical notation, as well as a MIDI player that can play audio of the pattern. Below them is the full list of tunes containing the selected pattern. Below the page title is the *Cite* button, giving a shareable link to the current view.

References

[1] S. Abdallah, E. Benetos, N. Gold, S. Hargreaves, T. Weyde and D. Wolff, The Digital Music Lab: A Big Data Infrastructure for Digital Musicology, *J. Comput. Cult. Herit.* **10**(1) (2017). doi:10.1145/2983918.

[2] M. Achichi, P. Lisena, K. Todorov, R. Troncy and J. Delahousse, DOREMUS: A graph of linked musical works, in: *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17*, Springer, 2018, pp. 3–19.

[3] C. Alexander, *A Pattern Language: towns, buildings, construction*, Oxford university press, 1977.

[4] 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 Trans. Internet Technol.* (2022). doi:10.1145/3555312.

[5] S.P. Bayard, Prolegomena to a study of the principal melodic families of British-American folk song, *The Journal of American Folklore* **63**(247) (1950), 1–44, Publisher: University of Illinois Press. doi:10.2307/537347. http://www.jstor.org/stable/537347.

[6] S.P. Bayard, Prolegomena to a study of the principal melodic families of British-American folk song, *The Journal of American Folklore* **63**(247) (1950), 1–44.

[7] I. Bent and W. Drabkin, *Analysis*, The New Grove Handbooks in Music, The MacMillan Press Ltd, Houndmills, Basingstoke, Hampshire & London, 1987.

[8] B. Breathnach, Between the jigs and the reels, 1982. http://msikio.online.fr/Breathnach/breandn.htm.

[9] V. Carriero, J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri and V. Presutti, The Music Meta Ontology: A Flexible Semantic Model for the Interoperability of Music Metadata, in: *ISMIR 2023 Hybrid Conference*, 2023.

[10] J.R. Cowdery, A fresh look at the concept of tune family, *Ethnomusicology* **28**(3) (1984), 495. doi:10.2307/851236. https://www.jstor.org/stable/851236?origin=crossref.

[11] M.S. Cuthbert and C. Ariza, music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data, in: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, 2010, pp. 637–642.

[12] J. De Berardinis, A. Meroño-Pe nuela, A. Poltronieri and Presutti, The Music Annotation Pattern, in: *WOP2022: 13th Workshop on Ontology Design and Patterns*, Hangzhou, China, 2022.

[13] J. de Berardinis, A.M. Penuela, A. Poltronieri and V. Presutti, The Music Annotation Pattern, in: *The Semantic Web–ISWC 2022 21st International Semantic Web Conference: 13th Workshop on Ontology Design and Patterns (WOP2022)*, 2022.

[14] 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* **10**(1) (2023), 641.

[15] J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri and V. Presutti, The Harmonic Memory: a Knowledge Graph of harmonic patterns as a trustworthy framework for computational creativity, in: *Proceedings of the ACM Web Conference 2023*, 2023, pp. 3873–3882.

[16] J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri and V. Presutti, The Harmonic Memory: a Knowledge Graph of harmonic patterns as a trustworthy framework for computational creativity, in: *Proceedings of the ACM Web Conference 2023*, 2023, pp. 3873–3882.

[17] J. de Berardinis, V.A. Carriero, N. Jain, N. Lazzari, A. Meroño-Peñuela, A. Poltronieri and V. Presutti, The Polifonia Ontology Network: Building a Semantic Backbone for Musical Heritage, in: *Proceedings of the 22nd International Semantic Web Conference*, 2023.

[18] D. Diamond, Automatic tune family detection in a corpus of Irish traditional dance tunes, Master's thesis, University of Galway, 2025, Forthcoming.

[19] D. Diamond, J. McDermott and M. d'Aquin, Tune family detection in Irish traditional music [Manuscript in preparation], 2023.

[20] B. Duggan, Tunepal: the traditional musician's toolbox, in: *Proceedings of the Second Workshop on EHeritage and Digital Art Preservation*, eHeritage '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 25–30–. ISBN 9781450301565. doi:10.1145/1877922.1877931.

[21] B. Duggan and B. O'Shea, Tunepal - Disseminating a Music Information Retrieval System to the Traditional Irish Music Community, in: *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, J.S. Downie and R.C. Veltkamp, eds, International Society for Music Information Retrieval, 2010, pp. 583–588. http://ismir2010.ismir.net/proceedings/ismir2010-100.pdf.

[22] B. Duggan and B. O'Shea, Tunepal: Searching a Digital Library of Traditional Music Scores., *OCLC Systems & Services* **27** (2011). doi:10.1108/10650751111182597.

[23] D.P. Escuredo, The Tonality Ontology: Draft specification of the Tonality Ontology, The Centre for Digital Music, Queen Mary, University of London, Accessed: 08-05-2023.

[24] B. Fields, K. Page, D. De Roure and T. Crawfordz, The segment ontology: Bridging music-generic and domain-specific, in: *2011 IEEE International Conference on Multimedia and Expo*, IEEE, 2011, pp. 1–6.

[25] A. Gómez-Pérez, Some ideas and examples to evaluate ontologies, in: *Proceedings the 11th Conference on Artificial Intelligence for Applications*, IEEE, 1995, pp. 299–305.

[26] A. Hillhouse, Tradition and innovation in Irish instrumental folk music, Master's thesis, The University of British Columbia,, Vancouver, Canada, 2005.

[27] E.J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R.M. Bittner and J.P. Bello, JAMS: A JSON Annotated Music Specification for Reproducible MIR Research., in: *ISMIR*, 2014, pp. 591–596.

[28] 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*, 2017, pp. 1222–1229.

[29] J. Keith, The Session, 2001. https://thesession.org.

[30] P. Knees, M. Schedl, T. Pohle and G. Widmer, An innovative three-dimensional user interface for exploring music collections enriched, in: *Proceedings of the 14th ACM International Conference on Multimedia*, MM '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 17–24–. ISBN 1595934472. doi:10.1145/1180639.1180652.

[31] F. Lerdahl and R.S. Jackendoff, *A Generative Theory of Tonal Music*, MIT press, 1996.

[32] F. O'Neill, *Irish minstrels and musicians*, Regan Printing House, Chicago, USA, 1913.

[33] A. Poltronieri and A. Gangemi, The music note ontology, in: *Workshop on Ontology Patterns*, 2021.

[34] Y. Raimond, S.A. Abdallah, M.B. Sandler and F. Giasson, The Music Ontology., in: *ISMIR*, Vol. 2007, Vienna, Austria, 2007, p. 8th.

[35] 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*, 2018, pp. 6–14.

[36] 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*, 2017, pp. 1–4.

[37] C. Sammut and G.I. Webb, *Encyclopedia of machine learning*, Springer Science & Business Media, 2011.

[38] H. Schenker, *Harmony*, Vol. 1, University of Chicago Press, 1954.

[39] A. Shahid, D. Diamond and J. McDermott, Patterns2KG: JAMS Pipeline for Modeling Music Patterns, in: *International Joint Workshop on Semantic Web and Ontology Design for Cultural Heritage*, 2023. https://api.semanticscholar.org/CorpusID:266377142.

[40] L.A. Steen, The Science of Patterns, *Science* **240**(4852) (1988), 611–616.

[41] R. Sweeney, P. Jajoria, D. Diamond, M. D'Aquin and J. McDermott, Patterns UI, An interactive tool for music exploration, in: *Sound and Music Computing Conference*, 2024.

[42] P. Tovstogan, X. Serra and D. Bogdanov, Visualization of Deep Audio Embeddings for Music Exploration and Rediscovery, in: *Proceedings of the 19th Sound and Music Computing Conference*, 2022, pp. 493–500.

[43] C. Walshaw, A multilevel melodic similarity framework, in: *Extended Abstracts for the Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conference, 2015*, Málaga, Spain, 2015.

[44] C. Walshaw, TuneGraph, an online visual tool for exploring melodic similarity, in: *Proc. Digital Research in the Humanities and Arts*, 2015. https://api.semanticscholar.org/CorpusID:2605457.

[45] C. Walshaw, Constructing Proximity Graphs to Explore Similarities in Large-scale Melodic Datasets., in: *6th Intl Workshop on Folk Music Analysis*, 2016, pp. 22–29.

[46] C. Walshaw, A Visual Exploration of Melodic Relationships within Traditional Music Collections, in: *2018 22nd International Conference Information Visualisation (IV)*, 2018, pp. 478–483. doi:10.1109/iV.2018.00089.

[47] D. Ó Maidín, A programmer's environment for music analysis, PhD thesis, University of Limerick, Limerick, Ireland, 1995. http://rgdoi.net/10.13140/2.1.3387.2969.

[48] M. Ó Suilleabháin, The creative process in Irish traditional music, in: *Irish musical studies 1: musicology in Ireland*, Irish musical studies, Vol. 1, Irish Academic Press, Dublin, Ireland, 1990, pp. 117–130.