Patterns2KG: JAMS Pipeline for Modeling Music Patterns

Abdul Shahid[1], Danny Diamond[2] and James McDermott[2]

[1]*National College of Ireland*
[2]*University of Galway, Ireland*

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, and integration with multiple data sources. 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 [1]) which formalizes key concepts in musical annotations. We developed a pipeline (Patterns2KG) to process existing data through the ontology, resulting in the production of Knowledge Graph triples. In this research, we processed a total of 40,152 compositions from `The Session` dataset to create the knowledge graph, comprising approximately 45 million (44,979,281) statements. We show sample SPARQL queries to document model usage. In assessing the effectiveness of our work, we engaged with musicologists to gain insights into their specific needs, which we translated into Competency Questions (CQs). These CQs serve to evaluate the model.

Keywords
Music Annotation, Music Patterns, Music Pattern Ontology, Patterns Knowledge Graph, 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.

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.

SWODCH'23: International Workshop on Semantic Web and Ontology Design for Cultural Heritage, November 7, 2023, Athens, Greece
abdul.shahid@ncirl.ie (A. Shahid); d.diamond1@universityofgalway.ie (D. Diamond); james.mcdermott@universityofgalway.ie (J. McDermott)

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. In the current paper, we describe the process of formalising this data as Linked Open Data (LOD), using an ontology and knowledge graph (KG).

1.1. What is a musical pattern?

Pattern is a central concept in many fields. Mathematics has been called "the science of patterns" [2]; "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 [4] claimed that repetition "is the basis of music as an art", Bent [5] proposed that "the central act" in all forms of music analysis is "the test for identity" and Lerdahl and Jackendoff [6] 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 $n = 4 \dots 10$. 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]https://github.com/polifonia-project/folk_ngram_analysis

1.2. Motivation for the work

The focus of this research is to present and preserve patterns within a robust structure, enabling effective music analysis. We propose to utilize ontologies and KGs to represent, organize, and structure musical pattern information, thus facilitating easier access, manipulation, and data analysis. Additionally, ontologies and KGs establish a shared vocabulary and standardized method for representing musical information. This, in turn, facilitates the integration of data from various corpora, cross-referencing, and linking related compositions. Furthermore, it provides a platform for discovering relationships among different pattern definitions as well.

1.3. Summary of the work

Therefore, firstly, we developed an ontology for musical patterns. It includes data such as the musical composition where the pattern is found, frequency and locations of pattern occurrences, the duration of the composition, the pattern contents, the 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. Additionally, the ontology includes metadata, 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. We will describe the ontology and KG in Section 2. Further, we also developed a custom software pipeline, based on previous work by project partners [7]. This pipeline includes several intermediate processes which will be described in Section 3.

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 knowledge graph.

2.1. Previous Work

Our work follows an existing workflow, the *Smashub* [2] workflow. Smashub uses an ontology for annotations in a musical score or recording, and this ontology is part of the Polifonia Ontology Network (PON)[3]. 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 [8]. 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 [7]. The other existing example is *Harmonic Memory* [9], a KG of harmonic patterns.

[2]https://github.com/smashub/choco
[3]https://github.com/polifonia-project/ontology-network

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

Several other ontologies focusing on music have been described in the literature, but none focussing on musical patterns.

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 [1]. 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.

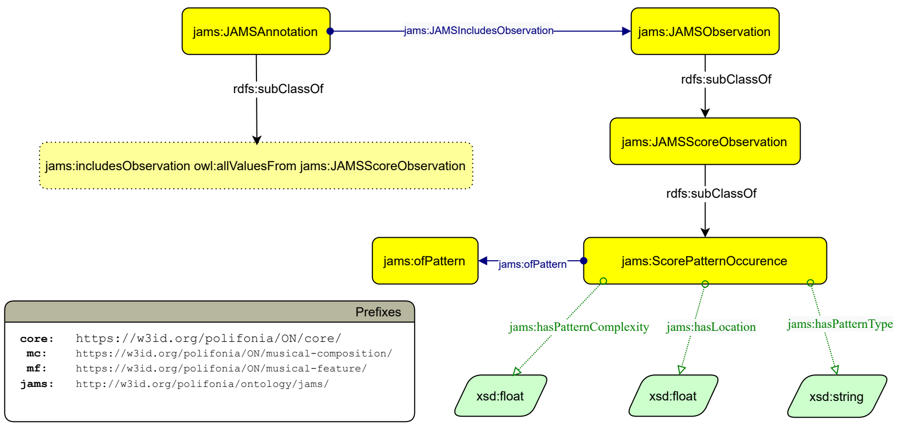

jams:JAMSAnnotation
jams:JAMSIncludesObservation
jams:JAMSObservation
rdfs:subClassOf
rdfs:subClassOf
jams:includesObservation owl:allValuesFrom jams:JAMSScoreObservation
jams:JAMSScoreObservation
rdfs:subClassOf
jams:ofPattern
jams:ofPattern
jams:ScorePatternOccurence
jams:hasPatternComplexity
jams:hasLocation
jams:hasPatternType
xsd:float
xsd:float
xsd:string
Prefixes
core: https://w3id.org/polifonia/ON/core/
mc: https://w3id.org/polifonia/ON/musical-composition/
mf: https://w3id.org/polifonia/ON/musical-feature/
jams: http://w3id.org/polifonia/ontology/jams/

Figure 1: Pattern Ontology based on MAP-ODP

There are two classes defined, named as `jams:JAMSObservation` and `jams:JAMSAnnotation`. The `jams:JAMSAnnotation` class includes multiple observations. The main concept that expresses patterns is the `jams:ScorePatternOccurrence` class. It specifies the occurrence of a pattern at a specific time in a specific composition. Its properties, such as `jams:hasLocation` and `jams:hasPatternType`, and `jams:hasPatternComplexity` are defined to represent the location, type, and complexity of a pattern respectively. The `jams:hasPatternComplexity` gives a simple numerical representation of complexity. It is the fraction of unique pitch values in a pattern. This can be used for various tasks such as filtering common patterns, interesting and unique patterns, etc.

The actual content (i.e., scale degree values) is described as a separate class, i.e., `jams:ofPattern` and it creates links between various patterns across the corpus. This will be used later to connect various composition in a corpus or different corpora. The `jams:hasPatternComplexity` represents how complex a pattern is which is the fraction of unique pitch values in a pattern and the length of the pitch. The current version of the ontology 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. A pattern occurrence is a location in time in a particular composition where a pattern occurs. A pattern has contents.

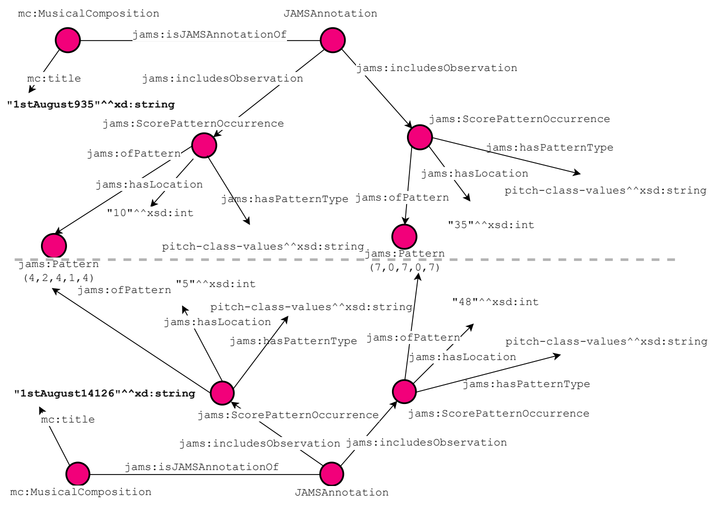


Figure 2: Extract from the pattern knowledge graph. One composition can be related to another via occurrences of the same pattern(s). The dotted line shows the "border" between compositions.

It illustrates also that pattern observations are recorded along with additional details such as pattern content (`jams:ofPattern`), types (`jams:hasPatternType`), and locations (`jams:hasLocation`). As described, many types of pattern are possible, but in our KG only the "pitch-class-values" type is used, so the pattern contents is the list of integer pitch values.

3. Patterns2KG: JAMS Pipeline

The Patterns2KG[4] pipeline consists of several stages. First, pattern sequences are extracted from the dataset using FoNN tools. Details are discussed in Section 3.2. The dataset includes composition metadata, 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. P2KG uses a specialized schema to represent patterns, which is described in Section 3.3. Next, P2KG 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 [10] 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.

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

3.1. Dataset

We begin by specifying the dataset we used. In our previous work [11], we conducted a comprehensive pattern analysis on a large dataset of Irish folk music, known as *The Session*[5] [12]. This dataset is crowd-sourced but is widely regarded as a definitive resource by practitioners. The dataset contains 40,152 compositions, each in ABC Notation, a symbolic music format widely used by practitioners and students of Western folk music.

3.2. Extracting patterns

The FoNN tools take input in ABC format and, for each composition, create sequences representing pitch, onset, duration, and velocity values for each note using the music21 Python library [13]. 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. 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 [14], and stored in a single corpus-level Pandas dataframe. Thus for both note-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[6].

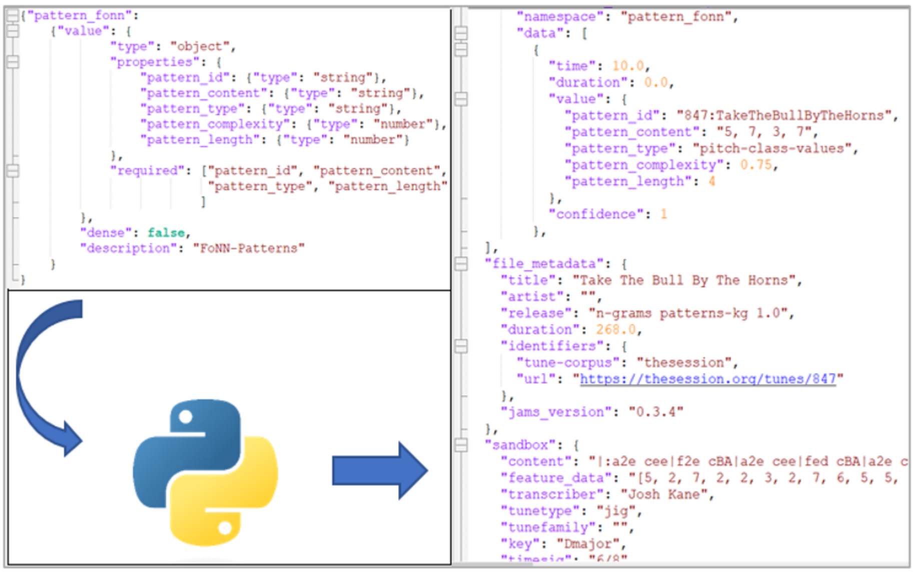
3.3. JAMS Pattern Schema

The MIR community uses JAMS: A JSON Annotated Music Specification for Reproducible MIR Research [15] 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 schema was designed, suitable for patterns extracted by FoNN tools, as shown on the left side of Figure 3. This schema is named `pattern_fonn`, with components including the pattern contents and location.

Using custom text-processing Python scripts, the pattern database was converted to `.jams` files which follow this schema. A snippet of the output is shown on the right side of Figure 3. 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 a composition file on the *The Session* website.

[5] https://thesession.org/
[6] https://github.com/polifonia-project/folk_ngram_analysis



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

```
      "namespace": "pattern_fonn",
      "data": [
        {
          "time": 10.0,
          "duration": 0.0,
          "value": {
            "pattern_id": "847:TakeTheBullByTheHorns",
            "pattern_content": "5, 7, 3, 7",
            "pattern_type": "pitch-class-values",
            "pattern_complexity": 0.75,
            "pattern_length": 4
          },
          "confidence": 1
        },
  ],
  "file_metadata": {
    "title": "Take The Bull By The Horns",
    "artist": "",
    "release": "n-grams patterns-kg 1.0",
    "duration": 268.0,
    "identifiers": {
      "tune-corpus": "thesession",
      "url": "https://thesession.org/tunes/847"
    },
    "jams_version": "0.3.4"
  },
  "sandbox": {
    "content": "|:a2e cee|f2e cBA|a2e cee|fed cBA|a2e c
    "feature_data": "[5, 2, 7, 2, 2, 3, 2, 7, 6, 5, 5,
    "transcriber": "Josh Kane",
    "tunetype": "jig",
    "tunefamily": "",
    "key": "Dmajor",
    "timesig": "6/8"
```

Figure 3: Left: FoNN-tools pattern schema for JAMS; Right: Sample JAMS annotation output

Table 1
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. The Session corpus contains these fields and others.
CQ2	Pattern search: Given a pattern, find a list of compositions it occurs in.
CQ3	Similar compositions: Given a composition, find a ranked list of similar compositions (based on pattern similarity).
CQ4	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 mc: <http://w3id.org/polifonia/ontology/musical-composition/>
PREFIX prov: <http://www.w3.org/ns/prov#>
```

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

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[7] This is a dynamically generated SPARQL construct query that transforms the JAMS file into RDF graph statements.

[7]Thanks to Andrea Poltronieri and Polifonia partners for help with this.

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

First, the unique URI for representing a `jams:JAMSAnnotation` is hashed with `SHA1` to ensure 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 knowledge graph. 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 knowledge graph 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 `jams:hasPatternType`.

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. The resulting KG is publicly available via Blazegraph[8] with a SPARQL endpoint[9].

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 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 [16]. 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 1. 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 we now present our responses and actions below. Essential namespace prefixes are given in Listing 1 for use in later Listings. These queries can be executed using the SPARQL endpoint already mentioned.

In Listing 4, the `?observation` URI is constructed using pattern content, allowing the same URI to be identified in other compositions representing the same pattern. Consequently, in the subsequent statement, `?anotherTuneJamsAnnotation jams:includesObservation ?observation` provides access to another composition's name. Finally, a filter is applied to ensure that both annotations are not identical, to avoid self-retrieval.

[8]https://blazegraph.com/

[9]https://polifonia.disi.unibo.it/fonn/sparql

```
SELECT distinct ?title ?tuneType ?key ?signature ?transcriber
WHERE {
VALUES ?givenTuneTitle {'Bucks Of Oranmore, The'}
?tuneFile jams:isJAMSAnnotationOf ?tune .
?tune mc:title ?title .
FILTER (?title = ?givenTuneTitle) .
?tune jams:tuneType ?tuneType .
?tune jams:key ?key .
?tune jams:timeSignature ?signature .
?tune jams:transcriber ?transcriber .
?tune jams:tuneContent ?tuneContent .
} LIMIT 4
```

Tune	TuneType	Key	Signature	Transcriber
2-Bucks Of Oranmore, The	reel	Dmajor	4_4	Jeremy
29659-Bucks Of Oranmore, The	reel	Dmajor	4_4	JACKB
29662-Bucks Of Oranmore, The	reel	Dmajor	4_4	JACKB
22356-Bucks Of Oranmore, The	reel	Dmajor	4_4	JACKB

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

```
SELECT distinct (concat(?tuneId, "-" ,?tuneTitle) as ?TuneInfo)
       ?TuneType ?TuneSignature
WHERE {
VALUES ?Pattern {<5_1_6_2_4_1>}
?observation  jams:ofPattern ?pattern .
?tuneFile jams:includesObservation ?observation .
?tuneFile jams:isJAMSAnnotationOf ?tune .
?tune mc:title ?tuneTitle .
?tune jams:tuneId ?tuneId .
?tune jams:tuneType ?TuneType .
?tune jams:timeSignature ?TuneSignature .
} LIMIT 4
```

TuneInfo	TuneType	TuneSignature
10963-A Day In Sligo	jig	6_8
11909-A Jig For Bernie	jig	6_8
12265-All Alive	jig	6_8
13055-Apples In Winter	jig	6_8

Listing 3: Finding compositions where a given pattern was found: query and results.

```
SELECT distinct
    (concat(?givenTunetitle) as ?GivenTuneInfo)
    (concat(?matchedTuneTitle) as ?MatchedTuneInfo)
    ?SharedPattern ?PatternComplexity
WHERE {
VALUES ?givenTune {'Bucks Of Oranmore, The'}
?tune mc:title ?givenTune .
?tuneFile jams:isJAMSAnnotationOf ?tune .
?tuneFile jams:includesObservation ?observation .
?observation jams:ofPattern ?SharedPattern .
?observation jams:hasPatternComplexity ?PatternComplexity .
?anotherTuneFile jams:includesObservation ?anotherTuneObser .
?anotherTuneFile jams:isJAMSAnnotationOf ?anotherTune .
?anotherTuneObser  jams:ofPattern ?SharedPattern .
FILTER(?tuneFile != ?anotherTuneFile) .
?tune mc:title ?givenTunetitle .
?tune jams:tuneId ?givenTuneId .
?anotherTune mc:title ?matchedTuneTitle .
?anotherTune jams:tuneId ?matchedTuneId .
} LIMIT 5
```

GivenTuneInfo	MatchedTuneInfo	SharedPattern	PatternComplexity
Bucks Of Oranmore, The	For The Sake Of Old Decency	2_5_2_5_2_5	”0.33”
Bucks Of Oranmore, The	Tom Keane’s	2_5_2_5_2_5	”0.33”
Bucks Of Oranmore, The	Hedgehog, The	2_5_2_5_2_5	”0.33”
Bucks Of Oranmore, The	Furze In Bloom, The	2_5_2_5_2_5	”0.33”
Bucks Of Oranmore, The	Lady Madelina Sinclair	2_5_2_5_2_5	”0.33”

Listing 4: Given a composition, find a ranked list of similar compositions (based on pattern similarity): query and results.

5. Conclusions

The Knowledge Graph (KG) serves as a semantic network that connects entities of diverse natures, allowing for easy integration of multiple data sources and scaling up the encoded information. Additionally, it facilitates the execution of complex queries that reveal relationships among participating entities. In this study, a pattern ontology based on Musical Annotation patterns was proposed, and a JAMS pipeline was developed to create a KG of the music patterns (n-gram pitch-class-values). We processed a total of 40,152 compositions of `The Session` dataset to create the KG, which comprises approximately 45 million (44,979,281) statements. The KG was evaluated using competency questions. By describing these CQs we have demonstrated some of the typical analyses which might be carried out by musicologists. The current modeling provides a solid foundation to undertake a deeper analysis of musical compositions and also has

```
SELECT (concat(?tuneId,"-",?tuneName) as ?TuneInfo) ?Pattern
WHERE {
VALUES  ?givenPattern {'5_2_5_2'}
?observation jams:ofPattern ?Pattern .
FILTER regex(str(?Pattern), ?givenPattern).
?tuneFile jams:includesObservation ?observation .
?tuneFile jams:isJAMSAnnotationOf ?musicalComposition .
?musicalComposition mc:title ?tuneName .
?musicalComposition jams:tuneId ?tuneId .
} LIMIT 5
```

TuneInfo	Pattern
2069-Moving Bog, The	2_5_2_5_2_5
2069-Moving Bog, The	5_2_5_2_5_2
2069-Moving Bog, The	5_2_5_2_5_2
2069-Moving Bog, The	5_2_5_2_1
10268-White Houses Of Shieldaig	7_3_5_2_5_2

Listing 5: Given a pattern, find all compositions when it or a pattern that contains that pattern occurs: query and results. 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.

the potential to answer typical musicologist requirements. In the future, we aim to integrate multiple corpora for cross-corpora analysis. We are also working on a user interface which uses our KG as a back-end, allowing non-technical musicologists to ask and answer research questions based on our KG. Finally, we would like to engage a wider community to ensure we can answer more complex types of user queries.

Acknowledgments

This work is supported by the Polifonia project. The Polifonia project has received funding from the EU Horizon 2020 research and innovation program under grant agreement No 101004746. The primary author completed this work while with the University of Galway.

References

[1] J. de Berardinis, A. M. Penuela, A. Poltronieri, 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.

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

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

[4] H. Schenker, Harmony, volume 1, University of Chicago Press, 1954.

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

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

[7] J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri, V. Presutti, ChoCo: a chord corpus and a data transformation workflow for musical harmony knowledge graphs, Scientific Data 10 (2023) 641.

[8] J. de Berardinis, V. A. Carriero, N. Jain, N. Lazzari, A. Meroño-Peñuela, A. Poltronieri, V. Presutti, The Polifonia Ontology Network: Building a semantic backbone for musical heritage, in: Proceedings of the 22nd International Semantic Web Conference, 2023.

[9] J. de Berardinis, A. Meroño-Peñuela, A. Poltronieri, 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.

[10] L. Asprino, E. Daga, A. Gangemi, P. Mulholland, Knowledge graph construction with a façade: A unified method to access heterogeneous data sources on the web, ACM Trans. Internet Technol. (2022). URL: https://doi.org/10.1145/3555312. doi:10.1145/3555312.

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

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

[13] M. S. Cuthbert, 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.

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

[15] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. M. Bittner, J. P. Bello, JAMS: A JSON annotated music specification for reproducible MIR research., in: ISMIR, 2014, pp. 591–596.

[16] 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.