Knowledge Engineering Project
Influence and inspiration in musical compositions
A.Y. 2022-2023

Francesco Alfieri
francesco.alfieri5@studio.unibo.it
ID 0001058698

Daniele Marini
daniele.marini3@studio.unibo.it
ID 0001060063

1 Introduction

Our project has been largely inspired by the LinkedJazz project (*linkedjazz.org*). In LinkedJazz, data about relationships and influences between jazz music artists were collected and represented as rdf triples, both via automated and manual processes (via, for example, *https://linkedjazz.org/52ndStreet/*). This kind of data, however, is largely unstructured, and only relies on triples with inter-personal predicates such as "*hasMet*", "*collaboratedWith*", "*mentorOf*". The goal of our project was to create an organized way of representing relationships and influences between music artists (not necessarily belonging to the Jazz genre) by creating an ontology, which can then be used to create more complex and semantically rich Knowledge Graphs. After having created the ontology, we proceeded to populate it with data from LinkedJazz, and with data extracted from texts via the Large Language Model ChatGPT.

2 Building the ontology

When building our *Musical Influences* ontology, we chose not to reuse parts of other ontologies at first. We still took a very strong inspiration from the *Music Ontology*, the *Polifonia Ontology Network* and the relations used within the *Linked Jazz* project. In fact, the majority of classes and properties have been mutuated and adapted from the mentioned ontologies.

However, by creating classes and properties from scratch (but still appropriately annotated with comments and labels), we managed to obtain slightly greater freedom and expressive power. In fact, while the basic building blocks have been taken from these ontologies, we created a large number of other classes and properties in order to make the ontology be able to represent knowledge needed to answer selected competency questions. After the ontology was created, further refining steps were taken in an attempt to conveniently align it to the *Polifonia Ontology Network*, by means for example of *SubClass Of*, *Equivalent To* axioms in Protegé. This alignment also included a manual "quality check", which consisted in verifying whether corresponding classes and properties were semantically similar or equivalent, and if the alignment would produce inconsistent results. During this step, we also encountered some issues with a number object and data properties in the core and music-meta modules of PON having some problems with domains and ranges, and we opened an issue on GitHub about them. For these reasons, the alignment was done manually only where considered “safe” and not all classes and properties were aligned to the corresponding ones.

The creation of a structured ontology for representing influences between artists allows not only to represent *direct* influences between artist (as it has been done in Linked Jazz), but also more complex, *indirect* influences. For example, the representation of time intervals, places and music genres allows to infer and/or represent relations such as “played in the same period”, “played the same genre”, etc. . . .

In fact, we created several competency questions that our ontology would need to be able to answer. For instance:

- Which music artist influenced another music artist?
- Which album/song, and of which artist or group, has been released and when?
- Who covered a role in a given music group during a certain time period?
- Where (in which city or country) and when an artist was born?
- Where (in which cities or countries) and when an artist lived?
- Which recording/performance/score realises a song?
- Who partecipated in the creation of a song?
- Which genre does a music piece/recording/performance belongs to?
- Which genre does an artist or group belong to?

Given the Competency Questions we extracted from textes, we reused some of the useful Ontology Design Patterns from the *ontologydesignpatterns.org* catalogues in order to be able to address them.

2.1 Ontology Design Patterns

2.1.1 Situation ODP

The (structurally) most complex pattern we adopted is the Situation ODP. This allowed us the represent complex n-ary relations. We defined a class *Situation*, with specialized subclasses such as *PerformanceSituation* and *GroupMembershipSituation*. In Figure 1 there is an example of implementation with a Graffoo diagram of the *performance situation*. The arguments of said n-ary relation are music artists and/or music groups, places (in our case cities and/or countries), performances (which realise music pieces), music roles, and dates. Then meaningful binary projections (for example *:memberOf* with *:MusicArtist* as domain and *:MusicGroup* as range) of the n-ary relations and necessary existential and universal restrictions were applied to the appropriate classes.

Moreover, this pattern allowed us to enrich several properties such as *performedWith*: in fact, we inserted some *Superproperty Of (Chain)* axioms with Protegé, which can allow a reasoner to infer, for example, that artists involved in a same performance situation should be associated via a *performedWith* property:

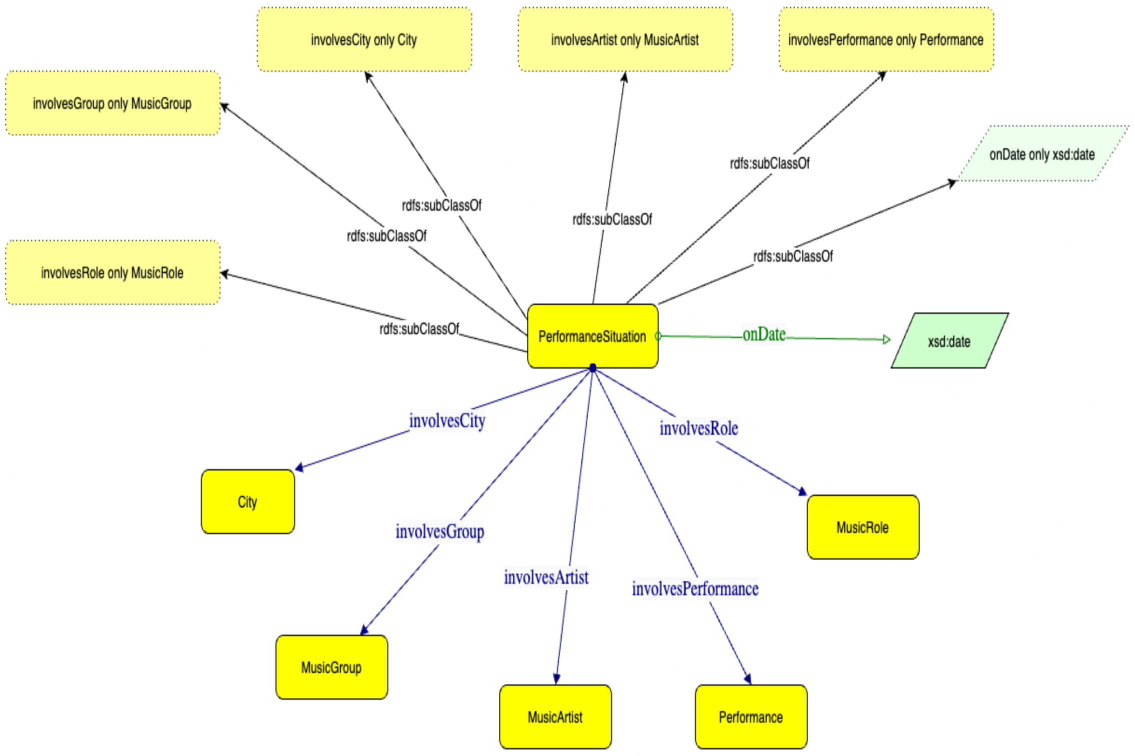

Figure 1: A Graffoo diagram for our performance situation pattern. The binary projection properties are not shown here in order to make the diagram more clear.

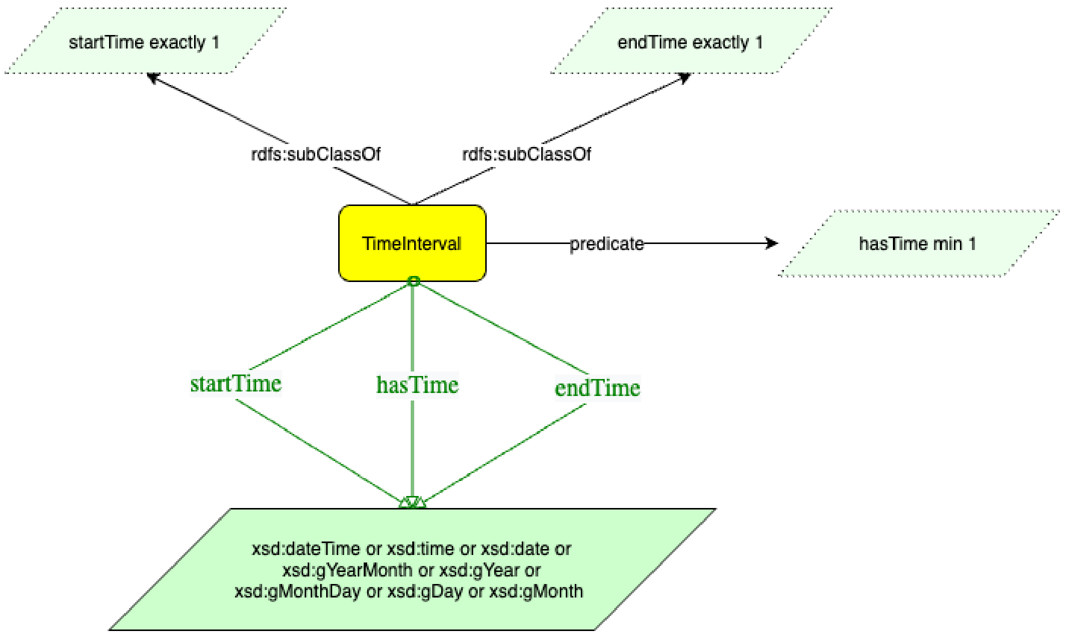

Figure 2: Graffoo diagram for the time interval pattern.

```
mi:performedWith owl:propertyChainAxiom (
    [owl:inverseOf mi:involvesArtist]
    mi:involvesPerformance
    [owl:inverseOf mi:involvesPerformance]
    mi:involvesArtist
)
```

Unfortunately, this and other similar axioms come with the cost of making some artists associated to themselves via some properties which connect artists. Nevertheless we believe that the ability to infer this kind of knowledge when not directly available from data is very valuable. Moreover, this kind of properties can allow to obtain other types of indirect influences/relations. For example we are able infer that artists lived in the same period and/or in the same place, which could imply that they influenced each other.

2.1.2 TimeInterval ODP

Another content ODP [1] [7] we used is the *TimeInterval* (Figure 2). This allowed us to represent intervals of time during which situations happened. In fact, this pattern has often been used in conjunction with the Situation pattern as the class *TimeInterval* has been made the range for example of the property *hasTimeInterval*, which has as domain the class of situations.

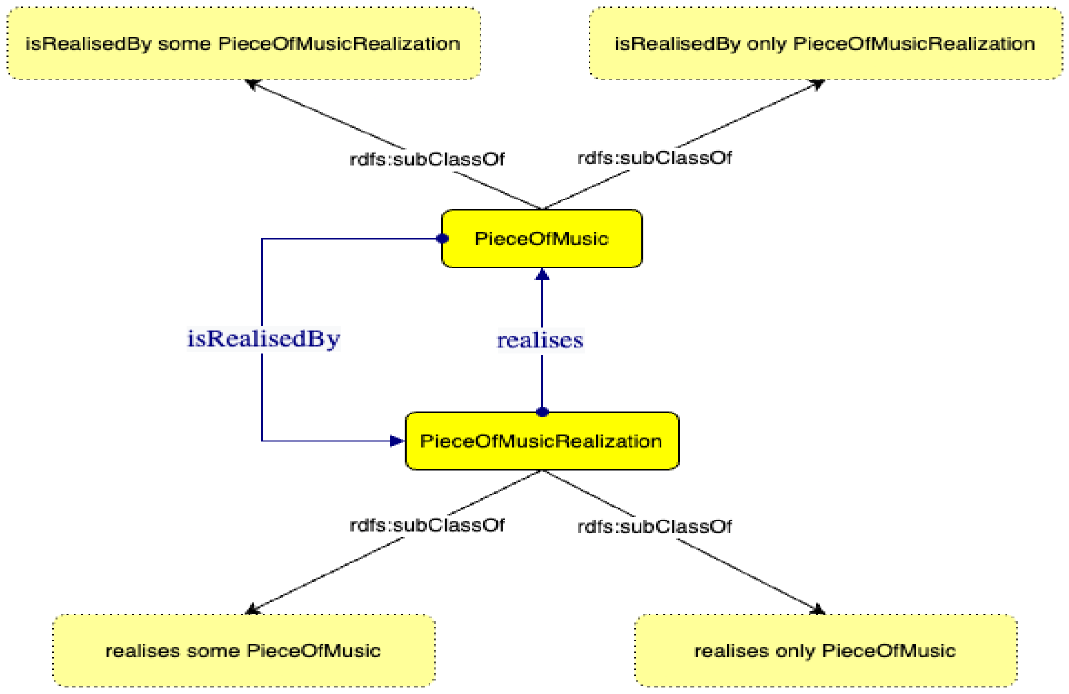
isRealisedBy some PieceOfMusicRealization
isRealisedBy only PieceOfMusicRealization
rdfs:subClassOf
rdfs:subClassOf
PieceOfMusic
isRealisedBy
realises
PieceOfMusicRealization
rdfs:subClassOf
rdfs:subClassOf
realises some PieceOfMusic
realises only PieceOfMusic

Figure 3: Graffoo diagram for the Information Realisation Pattern

2.1.3 Information Realisation ODP

We also used the information realisation pattern: we created a class *PieceOfMusic*, which represents songs and musical tracks as information objects, and a *PieceOfMusicRealisation* with subclasses *Lyrics, Performance, Recording, Score* as information realisations. This pattern allows us to distinguish between pieces of music (intended as "abstract" objects) and its various possible more concrete realisations. We decided not to introduce axioms *Disjoint Union Of* for *PieceOfMusicRealisation* in order to allow for any potential new classes to be added as subclasses of *PieceOfMusicRealisation*.

3 Populating the Knowledge Graph

3.1 Using data from LinkedJazz

A first experiment in order to populate the graph has been done with the data made available by the LinkedJazz project. In order to do that, we queried the semlab [5] (where the LinkedJazz project has been transferred) sparql endpoint at *https://query.semlab.io/*, with queries of the following type:

```
SELECT ?sLabel ?oLabel ?s ?o
WHERE
{
  ?s wdt:P39 ?o.
  ?s wdt:P1 <http://base.semlab.io/entity/Q1>.
  ?o wdt:P1 <http://base.semlab.io/entity/Q1>.
  SERVICE wikibase:label { bd:serviceParam wikibase:language
                             "[AUTO_LANGUAGE],en".
                           }
}
```

Here *wdt:P1* is the property *instance Of*, *<http://base.semlab.io/entity/Q1>* is the class *person*, and *wdt:P39* is the relation *influenced by* used in the LinkedJazz project [3]. Hence, this query retrieves all people related by said relation, together with their own labels. We repeated the query for all the relations from LinkedJazz that we translated into our ontology, and saved all the results as JSON files.

After that, we queried each JSON file with SPARQL Anything [2] in the following way:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xyz: <http://sparql.xyz/facade-x/data/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mi:<http://www.semanticweb.org/KEProject/MusicalInfluences#>

CONSTRUCT {
    ?s mi:*RELATION* ?o.
    ?s rdfs:label ?sLabel.
    ?o rdfs:label ?oLabel.
    ?s rdf:type mi:MusicArtist.
    ?o rdf:type mi:MusicArtist.
}
WHERE {
     SERVICE <x-sparql-anything:file://*FILE*.json> {
      ?b xyz:s ?s.
      ?b xyz:o ?o.
      ?b xyz:sLabel ?sLabel.
      ?b xyz:oLabel ?oLabel.
     }
}
```

This query creates a graph in which each entity (either subject or object) from the previously obtained JSON files are individual of the class *Music Artist*. Then, these entities are associated to each other with relations taken from our ontologies corresponding to the relations that had been extracted from semlab. Finally, each individual is assigned its own label.

We proceeded to save all the resulting triples as .ttl files and we created a dataset using them together with the .ttl file of our ontology via *Apache Jena Fuseki* [4].

This is already enough to query the data that can be obtained from the LinkedJazz project. As an example, we can ask who were the musicians who were influenced by Louis Armstrong, and who were the musicians who were in turn influenced by such artists (if any):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX mi:<http://www.semanticweb.org/KEProject/MusicalInfluences#>

SELECT DISTINCT ?s1label ?slabel
```

```
WHERE {
  ?s1 rdfs:label ?s1label.
  ?s1 mi:wasInfluencedBy ?o.
  ?o rdfs:label "Louis␣Armstrong".
  ?s mi:wasInfluencedBy ?o.
  OPTIONAL{
  ?s mi:wasInfluencedBy ?s1.
    ?s rdfs:label ?slabel.}
}
```

The query returns the results of Table 3.1.

"s1label"	**"slabel"**
"Danny Barker"	
"Buster Williams"	
"Louie Bellson"	
"Abbey Lincoln"	
"Lionel Hampton"	
"Lionel Hampton"	"Dave Brubeck"
"Roy Eldridge"	
"Roy Eldridge"	"Roswell Rudd"
"Mary Lou Williams"	
"Roswell Rudd"	
"Mona Hinton"	
"Doc Cheatham"	
"Milt Hinton"	
"Dave Brubeck"	
"Jimmy Owens"	
"Billy Taylor"	
"Buddy DeFranco"	
"Roy Haynes"	
"Bob Haggart"	
"Leslie Johnson"	
"Jimmy Scott"	
"Gerald Wiggins"	
"Delfeayo Marsalis"	

Table 1: Results from the query above: in the first column there are music artists who were influenced by Louis Armstrong according to Linked Jazz data. The second column contains the artists who were influenced by the corresponding artist in the first column on the same row.

3.2 ChatGPT experiments

In order to add more complex data to our ontology to create richer knowledge graphs, we also experimented with the ChatGPT [6] Large Language Model. Our approach was to give the model a sort of zero-shot learning task: we prompted it by providing the list of classes and properties from our ontology (and the *rdf:type* relation), together with texts taken from various biographies of musicians. By using this approach, we managed to obtain some interesting results, even if they presented a number of problems.

One of the most important issues is that said language model was not able to accurately reconstruct the semantics of classes and relations just by their names. In fact, a lot of mistakes were due to it misinterpreting domains and/or ranges of properties (either assuming them to be wrong classes, or just switching domains and ranges). This could potentially be addressed by providing the model with the complete description of the ontology; unfortunately, we attempted to do that, but the size of it was too large to fit in a prompt. Moreover, larger prompts could in general deteriorate significantly the accuracy and quality of results.

In addition, even if we provided the model with the list of classes and properties from our ontology, it had a tendency to create triples with predicates or classes that are not in the ontology.

Lastly, the URIs it created for individuals were not always consistent with each other. Even when triples made sense semantically, ChatGPT often creates multiple distinct URIs for the same intended individual. In this case, an additional step of entity-linking (for example, via *owl:sameAs* relation pointing to appropriate DBpedia entities) would be required.

Nevertheless, this way of populating the graph with data extracted from text (with the cost of it sometimes being inconsistent) allows to query it to obtain more complex informations than the ones we could obtain from Linked Jazz data only. In fact, with this method we are able to obtain enough data to answer some of our previously mentioned Competency Questions. For instance, we are able to answer to: *Where (in which city or country) and when an artist was born?* via the following query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX mi: <http://www.semanticweb.org/KEProject/MusicalInfluences#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT * WHERE {
  ?artist mi:birthDate ?date.
  OPTIONAL {?artist mi:birthCity ?city;
                    mi:birthCountry ?country}
}
```

In this case, we obtain as a result the following:

artist	**date**	**city**	**country**
mi:bingCrosby	May 3, 1903	mi:Tacoma	mi:UnitedStates
mi:DizzyGillespie	1917-10-21	mi:Cheraw	mi:UnitedStates
mi:LouisArmstrong	August 4, 1901	mi:NewOrleans	mi:UnitedStates

Table 2: Results from the query above. The Uri have been shortened with appropriate prefix in order to improve readability.

For completeness, in order to showcase the ability of the ontology to answer more complex competency questions, we also introduced some "manually crafted" data that allows for relatively richer knowledge to be extracted:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX mi: <http://www.semanticweb.org/KEProject/MusicalInfluences#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?artistName ?album ?date WHERE {
  ?releaseSituation mi:releaseOfAlbum ?album.
  ?releaseSituation mi:involvesArtist ?artist.
  ?artist rdfs:label ?artistName.
  ?releaseSituation mi:onDate ?date.
}
```

The query returns as result:

artistName	**album**	**date**
"Michael Jackson"	mi:thriller	"1982"

4 Further possible improvements

Creating an ontology for representing influences and relations between artists proved to be a complex challenge. The ontology we built allows to represent even relatively complex and structured knowledge in a satisfactory way. The reuse of ODPs and the adaptation of classes and properties from other existing ontologies greatly simplified the work and improved the quality of the results.

However, the population of the graph proved to be a somewhat hard task: when data is already structured in a way that is easily transferrable to classes and properties in the ontology (as in the case of LinkeJazz data), the population of the graph is a much smoother process, but knowledge extracted can be remarkably limited and not exploit the expressive power of the ontology; on the other hand, when we try to extract data from texts with LLMs, we can obtain more complex data, at the cost of some incorrect knowledge and inconsistent naming (which would require further entity-linking steps) and/or semantic misunderstanding of entities and properties. Each of these issues could potentially be tackled, but we deem this would be beyond the scope of our project.

Further future improvements might involve the use of more advanced and/or specific to the task LLMs (such as AutoGPT), and the exploitation of the graph structure of data with techniques such as random walks and Markov chains together with weights of links learning in order to learn and predict unknown, possibly indirect, influences relations.

References

[1] Ontology design pattern, http://ontologydesignpatterns.org/wiki/main_page, 2009.

[2] Luigi Asprino, Enrico Daga, Aldo Gangemi, and Paul Mulholland. Knowledge graph construction with a façade: A unified method to access heterogeneous data sources on the web. *ACM Trans. Internet Technol.*, 2022.

[3] Pratt Institute School of Library Information Science Cristina Pattuelli. Linked jazz, https://linkedjazz.org/, 2012.

[4] Apache Software Foundation. Apache jena, https://jena.apache.org/, 2021.

[5] Cristina Pattuelli Matt Miller. Semantic lab, https://semlab.io/, 2016.

[6] OpenAI. Chatgpt, https://chat.openai.com/, 2021.

[7] Hitzler Pascal, Gangemi Aldo, Janowicz Krzysztof, Krisnadhi Adila, and Presutti Valentina. Ontology engineering with ontology design patterns: Foundations and applications, 2016.