Empirical ontology design patterns and shapes from Wikidata

Valentina Anita Carriero [a,*], Paul Groth [b] and Valentina Presutti [c]

[a] *Department of Computer Science and Engineering, University of Bologna, Italy*
E-mail: valentina.carriero3@unibo.it
[b] *Faculty of Science, University of Amsterdam, Netherlands*
E-mail: p.t.groth@uva.nl
[c] *Department of Languages, Literatures and Modern Cultures, University of Bologna, Italy*
E-mail: valentina.presutti@unibo.it

Abstract. The ontology underlying the Wikidata knowledge graph (KG) has not been formalized. Instead, its semantics emerges bottom-up from the use of its classes and properties. Flexible guidelines and rules have been defined by the Wikidata project for the use of its ontology, however, it is still often difficult to reuse the ontology's constructs. Based on the assumption that identifying ontology design patterns from a knowledge graph contributes to making its (possibly) implicit ontology emerge, in this paper we present a method for extracting what we term *empirical* ontology design patterns (EODPs) from a knowledge graph. This method takes as input a knowledge graph and extracts EODPs as sets of axioms/constraints involving the classes instantiated in the KG. These EODPs include data about the probability of such axioms/constraints *happening*. We apply our method on two domain-specific portions of Wikidata, addressing the *music* and *art, architecture, and archaeology* domains, and we compare the empirical ontology design patterns we extract with the current support present in Wikidata. We show how these patterns can provide guidance for the use of the Wikidata ontology and its potential improvement, and can give insight into the content of (domain-specific portions of) the Wikidata knowledge graph.

Keywords: ontology design patterns, shapes, knowledge graphs, Wikidata, empirical knowledge engineering

1. Introduction

Ontologies are often developed in a top-down manner, for example, by starting from specific requirements coming from domain experts or through deriving concepts from application needs. Then, as a second step, those formally defined ontologies are populated by knowledge graphs (KGs). However, this is not always the case.

As an exemplary use case, Wikidata[1] is a knowledge graph that stores structured data for its Wikimedia sister projects [1] – including Wikipedia and Wiktionary. Wikidata contains a large number of factual statements about entities and events in the real world, belonging to various domains. It is built collaboratively and is edited on a daily basis. Similar to other knowledge graphs, Wikidata includes statements about its schema, for example, about the properties or classes of entities. In Wikidata, these ontological constructs are defined in an ontology whose definition is bottom-up and partially implicit. Moreover, the ontology underlying Wikidata is constantly subject to change due to frequent updates by its editors, and the way they model the data.

---

*Corresponding author. E-mail: valentina.carriero3@unibo.it.
[1]https://www.wikidata.org/

Due to this bottom-up and flexible definition, and to its constant evolution, it can sometimes become challenging for an (external) user to understand the knowledge graph, and to effectively reuse both the ontology and its data [2, 3]. The analysis of the content of a KG may have as goals (i) to understand the structure of the KG and its main contents, (ii) to determine whether the KG answers possible questions from the user, and (iii) to find the subset of triples that are pertinent to the user's use case [4]. In this context, it would be useful to exploit the KG's data to derive axioms or constraints as building blocks of a *background* ontology. Such an inferred ontology would represent an access point to the content of the knowledge graph, possibly supporting its reuse. Wikidata does provide some flexible guidelines around use (see Section 4); however, there still remains room to provide additional, more detailed, guidance on how to use Wikidata's ontology by exploiting its *actual usage*.

In [5], we introduced a method for extracting what we name *empirical ontology design patterns*[2] (EODPs). These patterns may be specific to a certain domain, based on the KG that is given as input, and consist of frequent domain-property-range triplets, accompanied by their usage statistics. Using the Wikidata knowledge graph as a use case, we demonstrated how these empirical patterns provide additional information about a domain-specific portion of the knowledge graph, not available from the existing guidelines on the usage of the semi-formally defined Wikidata ontology.

This paper extends [5], by presenting the development of two additional steps of the method, that transform these frequent triplets into:

- *probabilistic* ontology design patterns, using RDF-star, a syntax that extends RDF in order to make statements about statements, as sets of axioms that are given a probability value based on their actual usage in the KG;
- *probabilistic* ShEx shapes[3] as sets of constraints annotated with their probability through comments.

We also provide a clear definition of what we mean by probabilistic patterns/shapes, and explain which interpretation of probability we adopt (i.e. *frequentist* probability).

Moreover, while in [5] our experiments were only performed on a subset of Wikidata focused on the music domain, in this paper, besides re-running our method on the Wikidata music sub-KG, we also perform the same experiments on the *art, architecture, and archaeology* (AAA) domain. Besides providing additional, useful results, and comparing the results obtained from two different domains, this new contribution shows that our method is domain independent.

Finally, another novel contribution of this work is the comparison of the empirical patterns extracted from 3 different versions of Wikidata, with an interval of 15 months in between.

The remainder of the paper is organized as follows. In Section 2 we discuss related work that is relevant to the generation of shapes or data-driven patterns. Section 3 describes our method. Section 4 presents existing ways, constructs, and projects that support the reuse of the Wikidata ontology and KG, while Section 5 shows the results of our experiments on two Wikidata subsets in the music and AAA domains. In Section 6, we compare our results and current Wikidata functionality. Finally, in Section 7 we briefly discuss the potential in applying such method to KGs other than Wikidata. Section 8 concludes the paper and presents future work.

2. State of the art: shapes and data-driven patterns extraction

There exists many methods that generate constraints for concepts, in the form of shapes or patterns, mostly to provide support for validating knowledge graphs, but possibly useful also for its exploration.

Some of these approaches (like Astrea) are only based on ontologies. Astrea[4] [6] is a tool for automatically generating SHACL shapes from a set of ontologies: it executes a number of mappings between ontology constraint patterns and SHACL constraint patterns, which are encoded in the Astrea-KG. However, this tool does not consider the data level. In [7] various aspects of OWL and SHACL are compared, and useful mappings between the two languages are provided.

---

[2]What we term here empirical pattern corresponds to what we termed emerging pattern in [5].

[3]ShEx is one of the schema languages used to formalise shapes.

[4]https://astrea.linkeddata.es/

However, the majority of the methods build shapes, as collections of validation rules, from knowledge graphs. Shape Designer [8] is a graphical tool for automatically constructing valid SHACL or ShEx constraints that are satisfied from an RDF KG. The cardinality of the triple constraints (that is, *exactly one*, *optional*, *at least one*, *any number*) is derived from the data based on predefined rules. However, the tool cannot process large KGs, like Wikidata, without specifying a limit to the number of SPARQL query results. The experiments presented in [9] demonstrate that currently available methods cannot handle the scale of large knowledge graphs such as Wikidata: they crash even with KGs with a few millions triples[5]. sheXer [10] is a tool that automatically extracts shapes, serialising them in both ShEx and SHACL, by mining the graph structure and exploring the neighborhood of predefined target nodes. All constraints extracted with SheXer are linked to a trustworthiness score, which provides a means to filter constraints based on their frequency, and to sort and merge them in order to generate the final shapes.

Some methods are based on knowledge graph profiling techniques. These techniques generate concise and meaningful summaries from an RDF knowledge graph, which are then used as an input for constructing shapes. In [11] the authors present an approach that relies on machine learning techniques for automatically generating RDF shapes. Profiled RDF data are used as features, and the method uses the Loupe tool[6] [12], which provides information about the frequency of triple patterns (in the form $\langle subjectType, predicate, objectType \rangle$) that appear in a given dataset. Even if it is possible to extend this approach in order to generate other types of constraints, in this work the shapes are built at the class level, e.g. a shape for validating instances of the class `dbo:Person`. Moreover, they can generate only two constraint types, i.e. cardinality and range constraints, by analyzing data patterns and statistics. In [13], the profiles generated by ABSTAT are translated into SHACL shapes around a predefined target class. As a separate step, a human user can change and correct such shapes automatically generated. ABSTAT [14, 15] is a profiling tool that builds *semantic profiles* starting from a KG, also considering the possible ontology that the KG may be based on. The final profile is a set of what they name *Abstract Knowledge Patterns* (AKPs), where the pattern $\langle subjectType, predicate, objectType \rangle$ is associated with their occurrences. *subjectType* is the most specific type of the subject and *objectType* is the most specific type of the object; the hierarchy defined in the possible ontology is used to exclude more generic and redundant patterns.

As shown by [9], all existing approaches that automatically generate shapes include in such shapes a high number of constraints such that it is non-trivial for a human user to assess their correctness and validity. Moreover, in most cases, no constraint is produced for non-literal objects, i.e. most constraints do not indicate that the objects of a property should be of a specific type.

In [16, 17], a method for extracting what is termed Statistical Knowledge Patterns (SKPs) from a knowledge graph is presented. This method is based on statistical measures similar to the ones usually used for generating data-driven shapes, which rely on the frequency in the data. An SKP is expressed in OWL and is constructed around one of the *main* (i.e. with more instances) classes from an ontology: it reflects the usage of that class by enriching the class' properties and axioms from the ontology with properties and axioms that can be inferred thanks to statistical measures computed on the data. The most frequent (based on a threshold) properties in the data are selected, and the appropriate range axioms are introduced, unless they have been already explicitly asserted in the ontology. A catalogue containing 34 SKPs extracted from a version of DBpedia can be found online[7]. However, the code of the method presented in the paper is not publicly available, so it is not possible to reproduce the results, and the published SKPs do not include any metadata about the actual usage of the selected properties in the KGs, such as the number of occurrences in the data, and the provenance (i.e. the KG based on which they have been generated).

3. Method

Our method extracts the empirical patterns from a knowledge graph. At a fundamental level, the axioms and constraints extracted by our method are dependent on the KG being processed and its state at a given time. We *observe* the regularity of patterns and use that to assign a probability to each of the constructs we extract.

---

[5]The subKG of Wikidata on the music domain, one of the two subgraphs to which we apply our method, contains more than 5 millions triples.
[6]http://loupe.linkeddata.es/
[7]http://www.ontologydesignpatterns.org/skp/

We adopt the *frequentist* interpretation of probability[8]. The frequentist probability of an event is defined as the limit of its relative frequency in many trials. In our work, for *trials* we mean all the occurrences of an event (e.g. an instance being of a certain type) in the data. For example, let us consider a knowledge graph on the fishing domain, including 100 individuals. These 100 individuals will represent our trials. 55/100 individuals are instances of the class *Fishing Rod*, thus the frequentist probability that an instance in the fishing domain is a fishing rod is equal to 55/100, i.e. 55%. Again, this is true only in the context of the specific knowledge graph at a specific time. Indeed, each axiom/constraint needs to be also annotated with the KG which it is derived from.

We now will go into details about our method for extracting empirical patterns from a knowledge graph. An overview of the method can be seen in Figure 1. The method is also formally described in Algorithm 1.
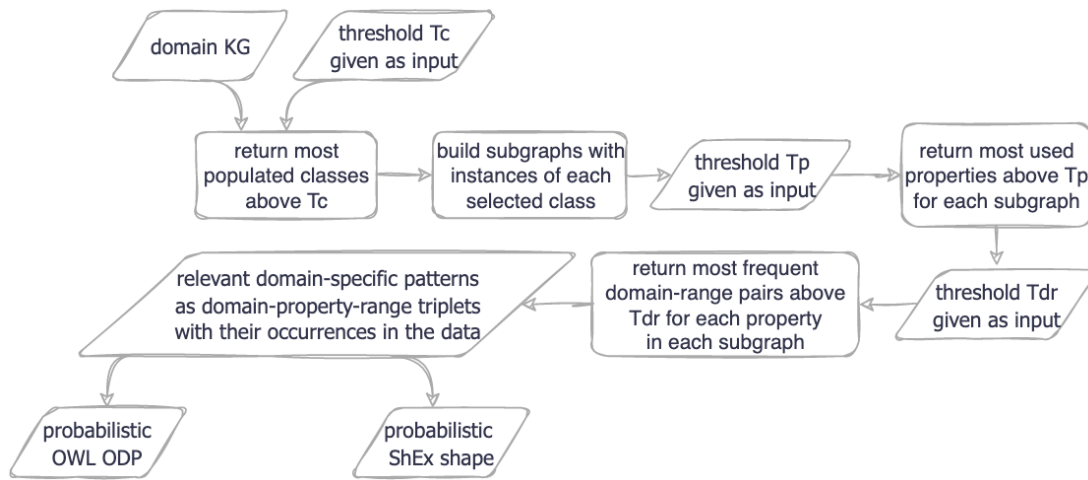

Fig. 1. Method for extracting empirical patterns from a KG.

Select relevant classes from the domain subgraph. The first step of the method takes as input the domain subgraph and returns the number of instances for each instantiated class of the graph. For each class, it also returns a percentage of coverage, which represents its frequentist probability, intended as a simple ratio between the number of instances of a class and the total number of distinct instances included in the graph. Subsequently, all classes that have a number of instances that fall below a given threshold are filtered out. This threshold needs to be given as input. The empirical patterns will be generated around the classes that have been selected based on this threshold, i.e. as a final output there will be a pattern corresponding to each filtered class.

This threshold is computed based on the absolute distance between the number of instances of a given class and the number of instances of the most populated class in the whole knowledge graph, i.e. the maximum value in the distribution. The distance is normalised, by dividing the result by the maximum value, such that this threshold falls within the [0, 1] range. If the threshold T_c is equal to 0, only the most instantiated class will be considered, since the distance between the count of the class and the maximum count must be smaller or equal to 0. On the contrary, if T_c is equal to 1, all classes instantiated by at least one individual will be taken into account, since the distance between the count of the given class and the maximum count must be smaller or equal to the maximum count. Note, that in our method we do not define the best thresholds to be used. The generated patterns incorporate information about their probabilities; however, it is the user that, based on her requirements and use cases, defines the most appropriate thresholds.

Extract a subgraph for each selected class. (lines 6-8 in Algorithm 1) Once the list of classes has been obtained, a subgraph for each class is constructed. This subgraph includes, from the domain subgraph, only the triples that have an instance of the given class as subject. For example, the subgraph corresponding to the class *album* will include all the triples where an instance of album is in the subject position.

Algorithm 1 Pseudocode of the algorithm for extracting empirical patterns from a KG.

1: **Require:** Thresholds: T_c, T_p, T_{dr}
2: **Require:** Input Domain Knowledge Graph: *KG*

   ▷ We treat a *KG* as a graph consisting of a set of triples with a subject (*s*), predicate (*p*), and object (*o*)

3: **Require:** The function SUBJECTCLASSES(k) retrieves the set of classes the subjects in a *KG* are member of.
4: **Require:** The function PROPERTIES(k) retrieves the set of properties in a given *KG*.
5: **Require:** The function OBJECTCLASSES(k) retrieves the set of classes the objects in a *KG* are member of.
6: **for all** $C \in$ SUBJECTCLASSES(KG) **do**
7:    $ODP_c \leftarrow \emptyset$
8:    $KG_c \leftarrow \{(s, p, o) \in KG \text{ s.t. } s \in C \}$
9:    **if** $|KG_c| > T_c$ **then**
10:        **for all** $P \in$ PROPERTIES(KG_c) **do**
11:            $KG_p \leftarrow \{(s, p, o) \in KG_c \text{ s.t. } p = P \}$
12:            **if** $|KG_p| > T_p$ **then**
13:                **for all** $R \in$ OBJECTCLASSES(KG_c) **do**
14:                    $KG_{dr} \leftarrow \{(s, p, o) \in KG_p \text{ s.t. } o \in R \}$
15:                    **if** $|KG_{dr}| > T_{dr}$ **then**
16:                        add $\exists(C \sqcap P.R)$ to ODP_c with probability $\frac{|KG_{dr}|}{|KG_c|}$
17:                    **end if**
18:                **end for**
19:            **end if**
20:        **end for**
21:    **end if**
22: **end for**

Most frequent properties for each class. (lines 9-11 in Algorithm 1) At this point, iterating over all subgraphs, we count the number of distinct instances that have at least one triple involving each property in the subgraph, i.e. we compute their occurrences. The frequentist probability, expressed as a percentage, represents the ratio between the number of instances that have a given property, and the total number of instances of the specific class. Then, for each subgraph, we include in the final empirical pattern only the properties that are above another threshold T_p that is given as input. We discard from this computation both the property expressing the type of an instance (`rdf:type` for RDF KGs, `wdt:P31`[9] for the Wikidata KG) and the property used for expressing the hierarchy of classes (`rdfs:subClassOf` for RDF KGs, `wdt:P279` for the Wikidata KG).

Most frequent ranges for each frequent property. (lines 12-14 in Algorithm 1) For each subgraph, we build the domain-property-range triplets, where *domain* is the type of the subject and *range* corresponds to either the type of the object (when the object is a resource) or the data type (e.g. `rdfs:Literal` for RDF KGs). These triplets will include only the properties that have been selected in the previous step. We calculate the number of occurrences of each triplet in order to find the most common pairs of domain and range for each property considered. Again, a threshold T_{dr}, provided as input, allows to include in the final set of triplets only the most common domain-range pairs for any of the most common properties selected in the previous step.

Empirical patterns: from triplets to probabilistic ODPs. (lines 15-16 in Algorithm 1) After these steps, we have obtained the patterns in the form of sets of domain-property-range triplets associated with their occurrences and probability values. We transform each domain-property-range triplet into an OWL existential axiom, that will be part of an OWL ontology design pattern. Each axiom is annotated with its probability with respect to the specific pattern.

---

[9]`wdt`: http://www.wikidata.org/prop/direct/

This is the reason why we chose, in the implementation of our method, to formalise the patterns using RDF-star[10], which extends RDF with a convenient way to make statements about other statements, that is reification. Indeed, RDF-star syntax allows for a subject or an object of a triple to be a triple themselves. For instance, with a single RDF triple, you can say that the author of *book A* is *author B*. However, you may need to say that the author of *book A* is *author B* based on a *stylistic analysis*. This can be modeled as an RDF-star triple with *based on* as its predicate, *stylistic analysis* as its object, and another triple as its subject, namely the triple *<book A, has author, author B>*. Similarly, with RDF-star we can say that one of the axioms of the empirical ODP has a certain probability according to the specific KG it has been extracted from. We rely on the owl-star vocabulary[11] for encoding ontological assertions (i.e. axioms) as statements that can be annotated. Moreover, we also use the OPLaX ontology[12] [18], which is a language explicitly designed for annotating ontology design patterns. As a result, each OWL pattern will be included in a file containing all its probabilistic axioms.

Empirical patterns: from triplets to shapes. Additionally, each set of triplets is transformed into a shape, where each constraint is annotated with its probability value through comments. In implementing the method, we rely on the Shape Expressions (ShEx[13]) schema language for expressing the shape. SheX is a language that allows to describe an RDF graph structure as a set of conditions that the RDF data graph must satisfy in order to be considered compliant. For instance, you can create a ShEx shape stating that an instance of a *book* must have at least a triple with the property *has author*. Usually, ShEx schemas are used for validating instance data.

4. Experiments on Wikidata: Motivation

Before describing the results of using our method for Wikidata, we first provide motivation for its usage based on the resources that have been created and adopted by Wikidata for recommending how to use its underlying ontology.
Property constraints. Property constraints[14], as defined by the Wikidata community, are rules on a property that specify how the property should be used in the knowledge graph, with possible exceptions. Indeed, these rules have a degree of flexibility: they aim at guiding the Wikidata editor in injecting or editing (new) statements in the KG, providing possible useful suggestions. Their definition is informal, without an explicit logical specification, thus they can still be violated or ignored. There are several types of property constraints. Two popular property constraint types are the *subject type constraint* and the *value-type constraint*, which indicate that the domain or range of a property, respectively, should be one in a list of classes. However, unlike OWL property restrictions on classes, they do not limit the applicable classes.
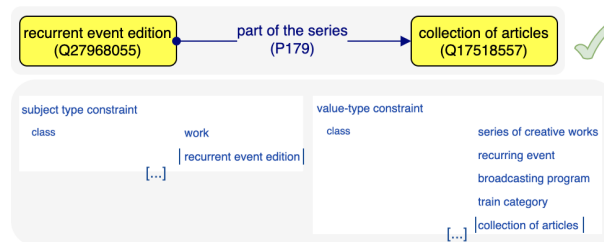

Fig. 2. Example of the limits of Wikidata property constraints.

For instance, as in Figure 2, a triple with an instance of *recurrent event edition* as subject, *part of the series* as predicate, and an instance of *collection of articles* as object would comply with the property constraints of the property, even if a more appropriate range in this case would be the class *recurring event* (`wd:Q15275719`).

---

[10]https://w3c.github.io/rdf-star/cg-spec/editors_draft.html
[11]https://github.com/cmungall/owlstar/blob/master/owlstar.ttl
[12]https://w3id.org/OPLaX
[13]https://shex.io/
[14]https://wikidata.org/wiki/Help:Property_constraints_portal

Properties for this type. The property *properties for this type* (`wdt:P1963`) allows to list the properties recommended to be used for instances of a certain type. For example, *part of the series* is one of the recommended properties for instances of the type *recurrent event edition*, however the appropriate range(s) to be paired with that specific type (e.g. *recurring event*) cannot be specified.

Type of Wikidata property. The class *Type of Wikidata property* (`wd:Q107649491`) is a Wikidata metaclass, i.e. the instances of this class are other classes that are related to a specific set of items, domain or topic; the property *facet of* (`wdt:P1269`) expresses the relation between the metaclass and its topic. These classes are placed into a hierarchy, and their instances are properties. For instance, as depicted in Figure 3, the property *Chessgames.com player ID* is member of the class *Wikidata property related to chess* that specialises *Wikidata property related to sport*. However, this classification of properties is activity in progress, thus it is not close to be complete for some domains; moreover, properties that are actually relevant to be used for instances of a certain type may be excluded from the metaclass specific to that set of items because they are declared as relevant only for more general domains.
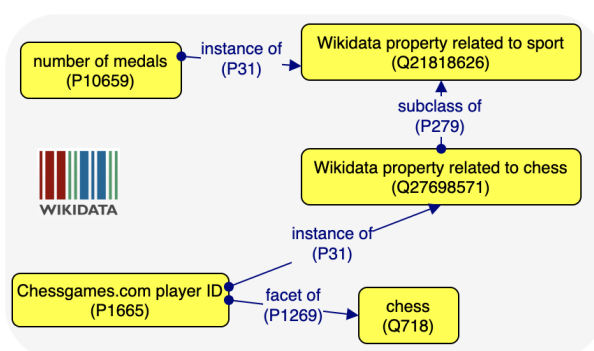

Fig. 3. Examples of Type of Wikidata property related to chess and sport.

Wikidata schemas. The Schemas Wikidata project[15] aims to define schemas, expressed in the Shape Expression language (ShEx) that can be used for the validation of subsets of items inside the Wikidata KG, in order to check whether they conform to a specific and *recommended* structure and possibly improve the quality of the data. Currently, more than 300 schemas have been manually defined by the Wikidata community[16], and these schemas vary considerably with respect to their size and granularity. For instance, the shape *E25* for actors[17] includes 4 constraints, and the only domain-specific constraint , i.e. a constraint that can be considered valid only for actors, not any human, specifies their occupation (*actor*). Instead, the shape *E42* for authors[18] is more detailed, and includes both constraints that are valid for all humans[19] and more *author-specific* constraints, such as *copyright status*. In any case, it is rare that these constraints express the suggested range; for example, the property *notable work* in the author shape has generically an IRI as recommended range, instead of possible specific classes.

Properties list in a WikiProject. In the context of domain-specific projects (e.g. music, astronomy, books, geology), the members of the community that are expert in that domain may define a list of properties that are recommended to be used for describing relevant entities of that domain. Each property, listed in a table, is usually associated with the data type of its range (that is, item, string, etc.), and a description of the usage of that property. This description, in some cases, also includes in plain text possible types for the range. For example, in the context of the WikiProject Books[20], the textual description of the property *inspired by*, recommended for *written work*s, recommends *artistic inspiration* as range. The whole process of defining relevant properties for a domain is performed manually, and possible ranges for properties are not always specified, and, in any case, not formally.

---

5. Experiments on Wikidata: Results

In this section, we discuss the results of applying our method to the Wikidata subgraphs on music and the 'art, architecture and archaeology' domains.

5.1. Input

In order to deal with the size of Wikidata, we used the Knowledge Graph Toolkit (KGTK)[21] [19]. KGTK is a framework developed as a Python library, whose aim is to support an easy manipulation of knowledge graphs, thanks to its scalability and speed. We downloaded a json dump of three versions of Wikidata[22]: on 04-04-2022 (*april 2022 version*), on 10-10-2022 (*october 2022 version*) and on 10-07-2023 (*july 2022 version*). The experiments that we present in Sections 5.2 and 5.3, and that we evaluate in Section 6, have been run on the *april 2022 version*. Section 6.4 presents a comparison of the patterns obtained from the three versions in order to show their evolution in 15 months.

In order to extract domain-specific patterns, and to handle a sub-graph of Wikidata with a more manageable size, we focus on specific domains represented in the Wikidata knowledge graph. While we work on the "music" and "art, architecture, and archaeology (AAA)" domains, the method can be applied to any domain. The extraction of instances related to both domains is based on a list of WordNet and BabelNet synsets identified as belonging to the respective domains, according to BabelDomains [20]. Then, the two Wikidata subgraphs are extracted by selecting each triple where the Wikidata domain-specific instance is in the subject position. The method we present in this paper does not focus on the extraction of domain-specific knowledge graphs from Wikidata (or any KG): by relying on BabelDomains we were able to easily obtain the two Wikidata subgraphs. However, a user that wants to use our method for extracting empirical patterns from a KG, could either run the method on the whole KG, or give as input a subgraph obtained with a method of their choice.

As explained in Section 3, each main step of our method takes as input a threshold, for filtering the list of classes (that is, the set of empirical patterns that will be generated), the list of properties to be included in each pattern, and the number of ranges that will define the final number of triplets/probabilistic axioms. These thresholds are to be defined by the user, based on her requirements. In this work, we do not present a method for finding the *best* thresholds to be chosen. However, for the purpose of presenting our results and comparing them with the current support in Wikidata, we have chosen reasonable defaults. For both the music and AAA patterns, the threshold T_c is equal to 0.95, the threshold T_p is equal to 0.85, and the threshold T_{dr} is 0.5[23].

The time required for producing all results with our method, with the chosen thresholds, is: about 8 hours for downloading the Wikidata dump; about 5 hours for processing such dump with KGTK and producing the necessary files for the extraction of the EODPs; about 1 hour for the actual extraction of the EODPs. The experiments have been executed on a RTX3090 with 128GB of RAM and Intel Core i9.

5.2. Wikidata emerging patterns on music

The Wikidata subgraph on the music domain contains 5,083,818 triples, and 226,989 distinct instances.

Most populated classes: music patterns. Having T_c equal to 0.95, we filter out all classes that have a number of instances that is lower than the 5% of the number of instances of the most populated class (from a total of 6,043 classes, ~6,000 of which have less than 200 instances). Note that the same entity can be an instance of more than one class.

In Table 1, the 7 classes around which we build music patterns are presented, with the respective number of instances and of triples that have an instance of the class as subject. The most relevant entities in this Wikidata music KG include both agents (such as human and musical group) and objects (namely, single, album, musical work, extended play, record label). Notice that *single* (wd:Q134556) and *extended play* (wd:Q169930) are not

---

Table 1

Most populated classes in the Wikidata music subKG.

Class	Instances	Triples
Q5 human	63,594	2,348,331
Q482994 album	63,213	723,722
Q215380 musical group	25,016	527,537
Q134556 single	20,977	253,201
Q105543609 musical work/composition	14,600	198,841
Q169930 extended play	3,816	33,725
Q18127 record label	3,640	35,118

subclasses of *musical work/composition* (`wd:Q105543609`) in the general Wikidata hierarchy (`wdt:P279*`). If we have a look at the ratio between the number of instances and the number of triples for each class, we notice at first sight that humans are more well *described* by facts than albums.

Recommended properties for each pattern. The average number of the most frequent properties selected for each pattern based on the T_p threshold is $\sim$21. In Table 2, we report the number of properties for each class, and the respective maximum and minimum number of their *occurrences*, intended as the number of instances that are subject of at least one triple including a specific property.

The number of selected properties is not directly proportional to the number of triples in the subgraph: for instance, for musical groups we recommend more properties that are frequently used (selected out of a total of 891 properties) than albums (369 properties in total). The most common properties across all patterns (without considering ID properties) are: `wdt:P136` *genre*, which is used with all 7 classes, and `wdt:P264` *record label*, present in all patterns except for record labels.

Table 2

Statistics of selected properties and triplets for each pattern.

Class	Properties	Occurrences		Triplets
		max	min	
Q5 human	48	63,583	9,543	63
Q482994 album	14	61,772	11,735	18
Q215380 musical group	33	22,423	3,474	38
Q134556 single	15	20,860	5,076	22
Q105543609 musical work/composition	17	13,916	2,204	29
Q169930 extended play	10	3,793	650	12
Q18127 record label	11	3,577	625	20

Recommended ranges for each property. Table 2 lists the number of triplets $\langle d, p, r \rangle$ – that is, the domain d and range r pairs for each recommended property p – generated for each pattern. Datatype properties will have only one range in any case, while for other properties the number of ranges depends on the $T_d r$ threshold. The average number of triplets across all patterns is $\sim$29. Since the same property can be involved in more than one pattern, for each pattern we may recommend different ranges for that property, except for datatype properties. That is, ranges recommendations are local to the individual pattern. For example, both the patterns for albums and singles include the property `wdt:P155` *follows*, with differnt ranges: *album* for albums, and *single* for singles, as expected.

Example: the album pattern. In Figure 4, we graphically represent the pattern for albums. Each domain-property-range triplet is accompanied by the number of instances in the Wikidata music subgraph that comply with that triplet (light blue rectangles). Based on the threshold we used, for most properties we recommend only one range. However, the *performer* property, when used with albums, can have either a human or a musical group as range
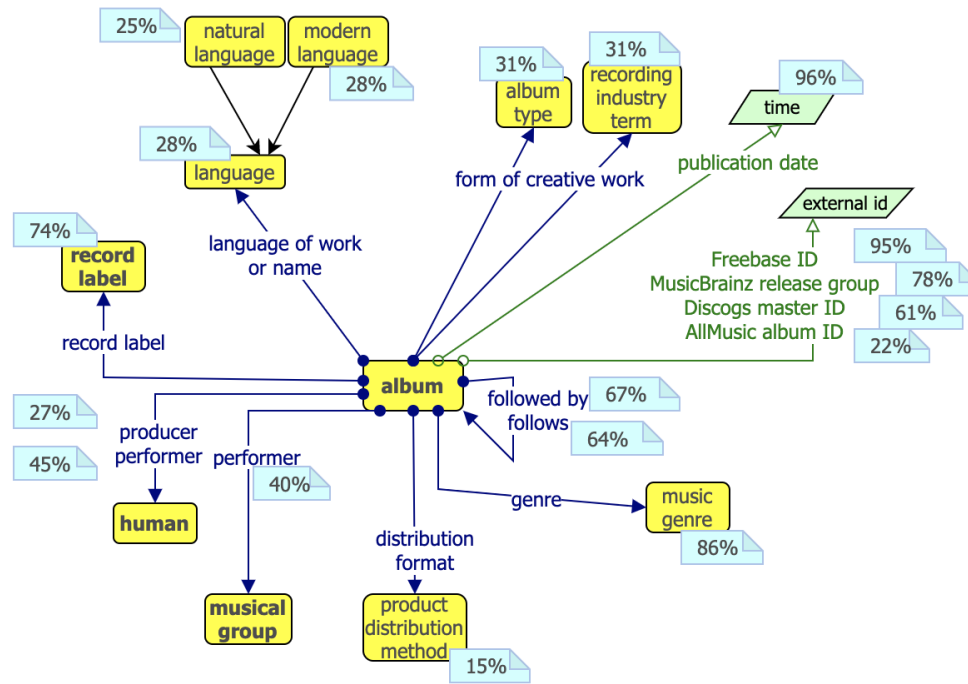
Fig. 4. The album pattern.

within the pattern, and the 3 recommended ranges for the property *language of work or name* have a subclass-of relation. It is interesting to notice that 4 recommended properties link to other empirical patterns as recommended ranges (*record label*, *human*, *musical group*).

Listing 1[24] contains a snapshot of the rdf-star – using the ttl-star syntax – probabilistic pattern about albums, derived from the sets of triplets selected from our method. The first 9 triples that can be found in 1, after the list of prefixes, aim at annotating the pattern itself and its related resources: the pattern (`weps:music_Q482994_09508505` is defined as a `oplax:ProbabilisticPattern`, is related to its source, i.e. the music Wikidata subgraph, and to the actual Wikidata class it is built around (`dcterms:references wd:Q482994`). Then, the music Wikidata subgraph, defined as a dataset, is linked to the whole Wikidata KG version it was derived from, and is part of (`weps:wikidata_20220404`). Finally, the whole version of Wikidata is annotated with the date on which the KG dump was downloaded ("2022-04-04"). The main properties and classes used for these annotations are also reported in Figure 5. The following statements are a subset of the probabilistic axioms we generate starting from the triplets of the pattern. `os:interpretation os:AllSomeInterpretation` indicates that these axioms are existential. The first one, `wd:Q482994 wdt:P175 wd:Q35120` (*album, performer, entity*) is the general probabilistic axiom about an album linked to some entity with the property *performer*. The frequentist probability of this axiom, annotated with `os:frequentistProbability`, is equal to 97.72%. However, as already explained, we also suggest

---

[24]@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix wd: <http://www.wikidata.org/entity/>
@prefix wdt: <http://www.wikidata.org/prop/direct/>
@prefix wikibase: <http://wikiba.se/ontology#>
@prefix dcterms: <http://purl.org/dc/terms/>
@prefix prov: <http://www.w3.org/ns/prov#>
@prefix os: <http://w3id.org/owlstar/>
@prefix oplax: <https://w3id.org/OPLaX/>
@prefix weps: <https://w3id.org/wikidata-eps/>

Listing 1 Snapshot of the album empirical ODP, expressed using rdf-star and owl-star.

```
weps:music_Q482994_09508505 rdf:type oplax:FrequentistProbabilisticPattern ;
 dcterms:source weps:wikidata_music_subkg_20220404 ; dcterms:references wd:Q482994 .

weps:wikidata_music_subkg_20220404 rdf:type dcterms:Dataset ;
  prov:derivedFrom weps:wikidata_20220404 ; dcterms:isPartOf weps:wikidata_20220404 .

weps:wikidata_20220404 rdf:type dcterms:Dataset ;
 dcterms:hasPart weps:wikidata_music_subkg_20220404 ; dcterms:date "2022-04-04"^^xsd:date .

# 'album' 'performer' 'entity'
<< << wd:Q482994 wdt:P175 wd:Q35120 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "97.72%" ; oplax:isNativeTo weps:music_Q482994_09508505 .

# 'album' 'performer' 'human'
<< << wd:Q482994 wdt:P175 wd:Q5 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "44.62%" ; oplax:isNativeTo weps:music_Q482994_09508505 .

# 'album' 'performer' 'musical group'
<< << wd:Q482994 wdt:P175 wd:Q215380 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "40.39%" ; oplax:isNativeTo weps:music_Q482994_09508505 .
```

more specific ranges for properties. In this case, two additional probabilistic axioms are generated: *album, performer, human* (probability 44.62%) and *album, performer, musical group* (probability 40.39%). The sum of the probabilities of such axioms is 85.01%: the remaining 12.71% corresponds to the sum of the probabilities of all the other possible, less frequent, ranges for this property when used with albums, that have been discarded based on the selected $T_d r$ threshold.
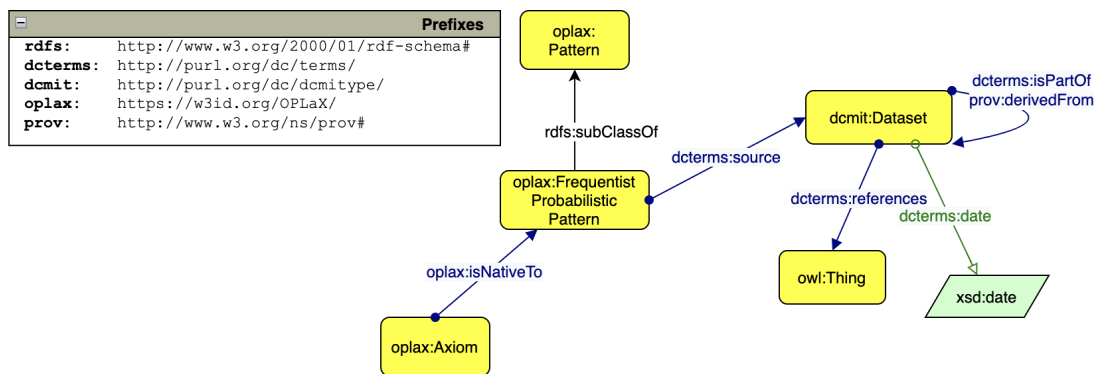

Fig. 5. Classes and properties used for annotating the probabilistic pattern.

The axioms in 1 correspond to the subset of the ShEx shape we generate for this pattern, that can be found in Listing 2. The frequentist probabilities are reported via comments. Additionally, the shape also includes the constraint about the type (`wdt:P31`) of the instances that need to comply with the shape, i.e. `wd:Q482994`.

5.3. Wikidata emerging patterns on art, architecture and archaeology

The Wikidata subgraph on the art, architecture, and archaeology (AAA) domain includes 493,999 triples, and 26,380 distinct instances.

Most populated classes: AAA patterns. The threshold T_c we use for the Wikidata Art, architecture, and archaeology (AAA) subgraph is 0.95, thus we filter out all classes that have a number of instances lower than the 5% of the number of instances of the most instantiated class (from a total of 2184 classes, ∼2100 of which have less than 50 instances). Clearly, the same entity can be an instance of more than one class.

Listing 2 Snapshot of the album empirical shape, expressed as a shape using ShEx.

```
# shape extracted from dataset wikidata_music_subkg_20220404
which is derived from wikidata_20220404, dated 2022-04-04

start = @<album>
<album> { wdt:P31 [wd:Q482994] } ;

# 'album' 'performer' 'entity'
wdt:P175 { wdt:P31 [wd:Q35120] } ;   # probability: 97.72%

# 'album' 'performer' 'human'
wdt:P175 { wdt:P31 [wd:Q5] } ;   # probability: 44.62%

# 'album' 'performer' 'musical group'
wdt:P175 { wdt:P31 [wd:Q215380] } ;   # probability: 40.39%   }
```

Table 3

Most populated classes in the Wikidata art, architecture, and archaeology subKG.

Class	Instances	Triples
Q5 human	7,622	214,881
Q3947 house	2,333	21,433
Q41176 building	1,617	15,382
Q23413 castle	1,043	13,556
Q33506 museum	636	9,129
Q11303 skyscraper	601	8,011
Q1343246 English country house	596	7,240
Q3305213 painting	564	12,814
Q1307276 single-family detached home	459	4,922
Q27686 hotel	427	4,577
Q207694 art museum	413	7,994

Table 3 lists the 11 classes around which we build our patterns, along with their number of instances and the number of triples with an instance of the class as subject. The most relevant entities in the Wikidata AAA domain include both agents (human, museum intended as an institution) and objects (house, skyscraper, painting, etc.). As for hierarchical relations (wdt:P279*) between the classes of the patterns, *building* has 6 (indirect) subclasses (*house*, *castle*, *skyscraper*, *English country house*, *single-family detached home*, *art museum*); *English country house* and *single-family detached home* are subclasses of *house* (indirect and direct, respectively); *art museum* is indirect subclass of both *museum* and *building*. However, surprisingly, *museum*, as opposed to *art museum*, is not hierarchically related to *building*, as well as *hotel*. Finally, *human* and *painting* are not subclasses of any other empirically derived class. By looking at the ratio between the number of instances and the number of triples, we can observe that humans, painting and art museums are more well *described* with facts than the other entities.

Recommended properties for each pattern. The threshold T_p we use for selecting the most frequent properties for each pattern is 0.85. The average number of selected properties for each pattern is ∼16. In Table 4 you can find the number of selected properties for each pattern, and the maximum and minimum number of *occurrences* from this set of properties, defined as the number of instances that are subject of at least one triple involving a specific property.

Recommended ranges for each property. Table 4 reports the number of triplets $\langle d, p, r \rangle$ selected for each pattern, using the 0.5 threshold . The average number of triplets across all patterns is ∼23.

Table 4

Statistics of selected properties and triplets for each pattern.

| Class	Properties	Occurrences		Triplets
		max	min	
Q5 human	33	7,614	1,151	46
Q3947 house	9	2,331	494	15
Q41176 building	10	1,613	278	16
Q23413 castle	12	1,043	161	22
Q33506 museum	15	632	99	19
Q11303 skyscraper	19	600	106	28
Q1343246 English country house	11	596	110	17
Q3305213 painting	23	562	91	32
Q1307276 single-family detached home	11	459	108	17
Q27686 hotel	13	427	79	19
Q207694 art museum	21	413	65	26
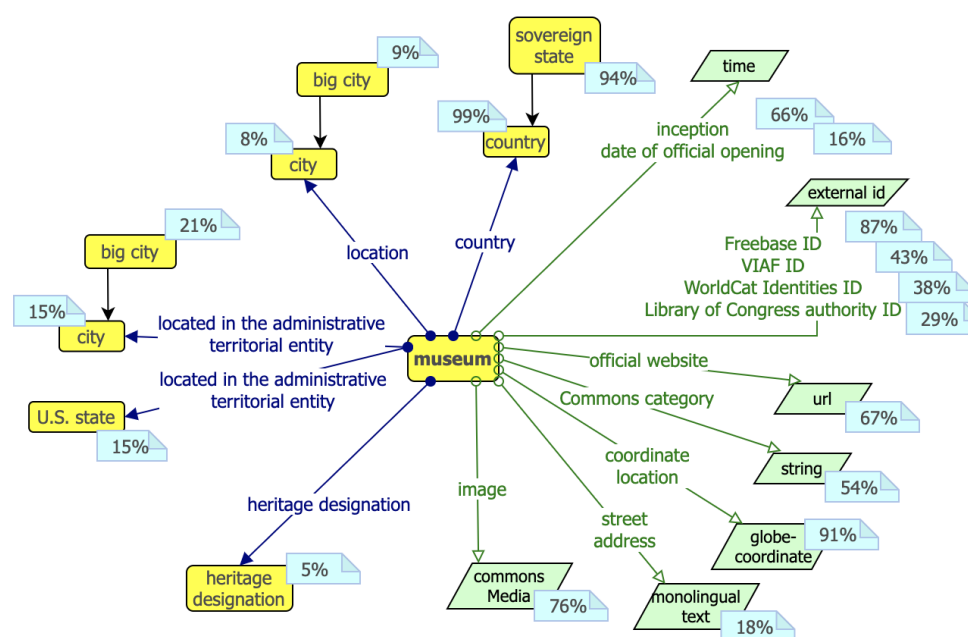
Fig. 6. The museum pattern.

Example: the museum pattern. In Figure 6 we provide a graphical representation of the pattern for museums. 11 out of 15 properties are datatype, and, as expected, have only one recommended range. Out of the remaining 4 object properties, 1 has one range, i.e. *heritage designation* for the homonymous property *heritage designation*, while the other 3 properties, which are all place-related, have multiple ranges: (i) *country* and *sovereign state* for the property *country*, the latter being a subclass of the former, with a very close percentage of coverage; (ii) *city* (8%) and its subclass *big city* (9%) for the property *location*; and (iii) again, *city* (15%) and its subclass *big city* (21%) for the property *location*, in addition to *U.S. state* (15%). As you can notice, between these 3 properties related to places, *country* is way more frequent, indeed almost all (99%) museums have this property.

Listing 3 includes a snapshot of the OWL probabilistic pattern for *museums*. Below the triples that annotate

Listing 3 Snapshot of the museum empirical ODP, expressed using rdf-star and owl-star.

```
weps:AAA_Q33506_09508505 rdf:type oplax:FrequentistProbabilisticPattern ;
 dcterms:source weps:wikidata_AAA_subkg_20220404 ; dcterms:references wd:Q33506 .

weps:wikidata_AAA_subkg_20220404 rdf:type dcterms:Dataset ;
 prov:wasDerivedFrom weps:wikidata_20220404 ; dcterms:isPartOf weps:wikidata_20220404 .

weps:wikidata_20220404 rdf:type dcterms:Dataset ;
 dcterms:hasPart weps:wikidata_AAA_subkg_20220404 ; dcterms:date "2022-04-04"^^xsd:date .

# 'museum' 'inception' time
<< << wd:Q33506 wdt:P571 wikibase:Time >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "66.35%" ; oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'entity'
<< << wd:Q33506 wdt:P17 wd:Q35120 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "99.37%" ; oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'country'
<< << wd:Q33506 wdt:P17 wd:Q6256 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "99.21%" ; oplax:isNativeTo weps:architecture_Q33506_09508505 .

# 'museum' 'country' 'sovereign state'
<< << wd:Q33506 wdt:P17 wd:Q3624078 >> os:interpretation os:AllSomeInterpretation . >>
os:frequentistProbability "94.5%" ; oplax:isNativeTo weps:architecture_Q33506_09508505 .
```

Listing 4 Snapshot of the museum empirical ODP, expressed as a shape using ShEx.

```
# shape extracted from dataset wikidata_AAA_subkg_20220404
which is derived from wikidata_20220404, dated 2022-04-04

start = @<museum>
<museum> { wdt:P31 [wd:Q33506] } ;

# 'museum' 'inception' time
wdt:P571 xsd:dateTime ;   # probability: 66.35%

# 'museum' 'country' 'entity'
wdt:P17 { wdt:P31 [wd:Q35120] } ;   # probability: 99.37%

# 'museum' 'country' 'country'
wdt:P17 { wdt:P31 [wd:Q6256] };   # probability: 99.21%

# 'museum' 'country' 'sovereign state'
wdt:P17 { wdt:P31 [wd:Q3624078] } ;   # probability: 94.5%   }
```

the pattern and its related sources, we include a subset of the probabilistic axioms that have been automatically generated. `wd:Q33506 wdt:P571 wikibase:Time` (*museum, inception, time*) is a probabilistic axiom that includes the datatype (*time*). This is, as expected, the only range for the property *inception*, and has frequentist probability equal to 66.35%. The next statements concern the property *country*; from the probability values, it can be noticed that *country* and *sovereign state* are used quite frequently as ranges for this property when the subject is a museum, and the majority of the instances in the object position for this property are of type both *country* and *sovereign state*. In Listing 4, we report the shape constraints that correspond to the axioms in Listing 3.

6. Experiments on Wikidata: Discussion

In this section, we discuss the results we obtain from both the music and AAA Wikidata knowledge graphs, and we perform an evaluation of the resulting empirical patterns through a comparison between them with the current support to reuse provided by Wikidata.

6.1. Music patterns

Patterns coverage. In order to observe how the extracted patterns are populated in the Wikidata subKG on music, we report in Table 5[25] the percentage of the total number of instances that cover different and increasing subsets of recommended properties. No pattern shows a 100% coverage even considering only the most frequent property. For 4/7 patterns, the instances that comply with the first half (1/2) of the recommended properties are between the 35 and ~58% of the total number of instances. On the contrary, musical work, human, and musical group patterns have already a very low percentage of coverage. The most populated pattern (including all properties), with respect to the total number of instances, is *extended play* (112/3,816), followed by *album* (845/63,213) and *musical group* (327/25,016). Instead, the pattern that has the lower percentage of coverage is *musical work*: only 1 out of a total of 14,600 musical works. In any case, all patterns have at least 1 representative instance. While the coverage percentages might seem very low, this is not *bad*, neither surprising: by applying the 0.85 threshold, as we did in our experiments, we include all properties that are used for at least 15% of the total number of instances. So, if we take the *record label* pattern as an example, considering that the least common property is used for 625/3,577 instances, it is not surprising that the intersection of instances with all the 11 recommended properties is equal to 28 instances.

Comparison with property constraints. If we take into account the most common properties across all patterns, that is *genre* (7/7 patterns) and *record label* (6/7), we can notice that the domains and ranges we suggest are all included in the *subject type* and *value-type* constraints of the two properties. In some cases, the Wikidata constraints list as ranges classes that are more general in the hierarchy with respect to the classes we suggest: for instance, they include work in place of the more specific musical work. However, as explained in Section 4, the correct pairs of domain and range cannot be specified inside Wikidata, thus our results integrate these constraints by suggesting that e.g. *music genre* is *more appropriate* as range of the property *genre* with *record label* as domain, than e.g. *criticism* – which is included in the value-type constraint of *genre*, and which never occurs in the data. Moreover, the *subject type* and *value-type* constraints are not available for all properties; for instance, *follows* (used in 4 out of 7 patterns) has no such constraints.

Table 5

Percentages of coverage of the patterns properties in the KG.

Class	1 prop	2 props	1/8	1/4	1/2	all
Q5 human	99.98	98.99	[8] 50.34	[12] 32.97	[24] 3.65	[48] 0.007 (5 instances)
Q482994 album	97.72	94.19	[2] 94.19	[4] 78.30	[7] 40.48	[14] 1.33 (845 instances)
Q215380 musical group	89.63	78.36	[4] 60.99	[8] 34.22	[16] 9.82	[33] 1.31 (327 instances)
Q134556 single	99.44	98.80	[2] 98.80	[4] 87.87	[7] 57.67	[15] 0.71 (151 instances)
Q105543609 musical work	95.31	76.36	[2] 76.36	[4] 39.69	[8] 6.34	[17] 0.006 (1 instance)
Q169930 extended play	99.39	97.95	[1] 99.39	[3] 92.29	[5] 56.70	[10] 2.93 (112 instances)
Q18127 record label	98.26	84.25	[1] 98.26	[3] 69.06	[5] 35.0	[11] 0.76 (28 instances)

Comparison with properties for this type. Taking into account our selected music patterns, we compared the properties we include and those included as value of the property *properties for this type* (`wdt:P1963`) for those classes. Based on a manual observation, we can report that some properties that are highly populated in the data are not suggested as properties for this type. Instead, all the properties listed as properties for the type, but not included in our patterns, are significantly less frequent. Let us take musical group (`wd:Q215380`) as an example[26]. As in Figure 7, IDs properties such as *Freebase ID* are widely used, but not listed as properties for this type. On the contrary, IDs that are less frequent in the data (e.g. *Apple Music artist ID (U.S. version)*), are filtered out from our

---

[25]Columns indicate the number/fraction of properties considered. The actual number of properties corresponding to the fraction is reported in square brackets. The number of instances covering the whole pattern is in round brackets. Example instances populating the whole patterns can be found here: https://github.com/valecarriero/wikidata-emerging-patterns/tree/main/results/supplementary_materials/example_instances

[26]https://github.com/valecarriero/wikidata-empirical-patterns/blob/main/results/music/supplementary_materials/properties_forthis_type/ Q215380_properties_comparison.tsv

pattern, but are recommended as properties for this type. That being said, 10 out of the 18 properties recommended as properties for this type are also included in our pattern.

Comparison with type of wikidata property. wd:Q27525351, a subclass of *Type of Wikidata property*, includes as instances *properties related to music*, such as music-related IDs (such as *YouTube playlist ID*) and other relations (e.g. *composer*, *performed at*, *discography*). However, it is not indicated, for each property, their possible domain(s), so, if a user needs to model a specific musical entity, e.g. a musical group, would have no support for understanding which of those properties to use for musical groups. *Wikidata property related to music* has 24 subclasses that are specific to some musical entities (e.g. music genres, songs, instruments). However, 14/24 include only identifiers, e.g. IDs for musical works, songs and bands. Even considering just the identifiers, our patterns are more complete and representative. For example, as depicted in Figure 7, the class *Wikidata property to identify bands* has as instance only the property *Encyclopaedia Metallum band ID*. The pattern we extracted for the class *musical group* contains 33 properties, including the most common IDs, while excluding wdt:P1952, which is used with only 8% of musical groups. Moreover, some relevant properties that we are able to include in the patterns cannot be identified based on the *Wikidata property* classes: for instance, *genre*, which is widely used in the data for describing instances of musicians (about 50%) and musical works (about 60%), is recommended for both humans musical works/compositions, while it can only be found as instance of the more general classes *Wikidata property for items about people* and *for items about works*.
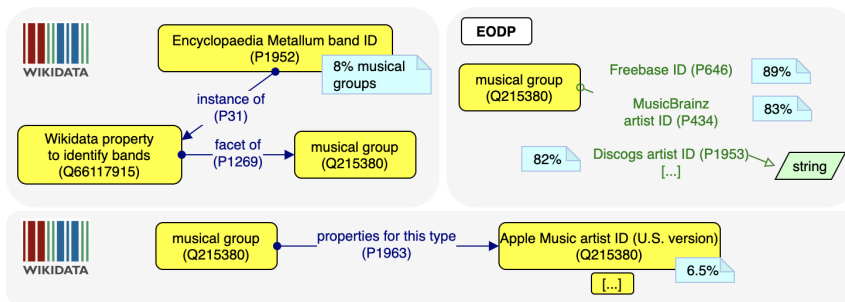

Fig. 7. Example of comparison between our pattern and the Wikidata support for describing musical groups.

Comparison with properties listed in the WikiProject Music. The WikiProject Music[27] (WPMusic hereinafter) identifies 6 relevant entities in the domain, and defines a set of recommended properties for each of them: human, musical ensemble, musical work, track, release, record label. Apart from *human* and *record label*, our patterns do not perfectly overlap with these 6 classes: musical ensemble *vs* musical group (the latter is the most populated subclass of the former); musical work *vs* musical work/composition (musical work has very few direct instances, while many of its subclasses are widely populated e.g. song); release, which groups together its subclasses album, single and extended play. However, it is still possible and useful to make a comparison between them. Let us consider the class *record label*: the WPMusic recommends 4 properties[28] in addition to 13 IDs. Our pattern contains 6 properties, in addition to 5 identifiers. We report in Table 6 a comparison between the properties recommended by WPMusic and those selected by our method (EP), except for IDs and wdt:P31[29]. It can be observed that our pattern is more inclusive (6 *vs* 3 properties) with the threshold we have chosen. More importantly, we include all properties recommended by WPMusic, while WPMusic does not recommend the second most frequent property *inception*. In our pattern, *country* (that is the range of the property *country* recommended by WPMusic) is indeed the most frequent range, but we also suggest 6 additional and more specific classes, such as *sovereign state*.

Now, let us compare the properties recommended for instances of subclasses of Release by WPMusic and our relevant patterns album (A), single (S) and extended play (P) (Table 7). 6 properties recommended by WPMusic are

---

Table 6

Comparison between properties recommended by WikiProject Music and properties included in our pattern for record labels.

Property	Occurrences	WPM	EP
P17 country	3,123	Y	Y
P571 inception	2,905	N	Y
P856 official website	1,833	Y	Y
P159 headquarters location	1,023	Y	Y
P136 genre	972	N	Y
P112 founded by	714	N	Y

Table 7

Comparison between properties recommended by WikiProject Music and properties included in our patterns for releases.

WPM Property	EP	WPM Property	EP	WPM Property	EP
P577 publication date	A, S, P	P136 genre	A, S, P	P156 followed by	A, S, P
P155 follows	A, S, P	P264 record label	A, S, P	P175 performer	A, S, P
P162 producer	A, S	P407 language of work	P	P361 part of	S
P1303 instrument	none	P483 recorded at studio	none	P676 lyrics by	none
P86 composer	none	P658 tracklist	none	P736 cover art by	none
P2291 charted in	none	P9237 reissue of	none	P1638 working title	none

included in all our 3 patterns, 3 properties are included in some of our patterns, while 9 properties are not included. However, for instance, the property *composer* is used only 6, 186 and 1602 times for instances of extended play, album and single, respectively; *instrument* is never used for any of these entities (instead, it is frequently used, and is included, in the *human* pattern); *working title* is used only twice for albums, while the property *title* (wdt:P1476) is much more frequent (9,007 occurrences).

Comparison with music-related shapes. We manually identified from the complete list of Wikidata entity schemas[30] only two music-related shapes: *music composition by W.A.Mozart* (E66), and *album* (E248), so it is strongly relevant to try to increase the coverage at least of the music domain. The shape for albums[31] (E248) recommends 18 properties as obligatory (with *exactly one/at least one* constraints): 7 of these properties are also included also in our pattern; while some recommended properties can actually be considered statistically relevant (such as the property *title*: 9,007/63,213 occurrences) and would have been included in our pattern with a little higher threshold, other properties included in the shape have very few occurrences that, at least, do not seem to justify their mandatory use (e.g. *review score*, with 722 occurrences, and *distributed by*, with 299 occurrences). Instead, for instance, the *producer* property (with *any number* as cardinality constraint in the E248 shape) is much more used (18,362), and is indeed recommended in our pattern.

6.2. Art, architecture and archaeology patterns

Patterns coverage. Table 8 reports the percentage of the total instances that covers increasing subsets of the recommended properties for each class. 3 out of 11 AAA patterns have 100% coverage considering only the most frequent property: *castle*, *English country house*, and *art museum*. However, while in the case of the *castle* and *English country house* patterns, this percentage of coverage with the most frequent property is associated with a high (wrt the other patterns) percentage of coverage also considering all properties of the pattern – 2.78% and 9.89%, respectively, being *English country house* the most populated pattern out of all 11 patterns –, *art museum* has only one instance that complies with the whole pattern (0.24%). Other patterns widely populated (considering all properties), wrt the total number of instances, are *house* (4.88%) and *single-family detached home* (5.88%). All patterns but

---

painting have at least one representative entity for the whole set of properties: the pattern *painting*, which has a very high coverage (higher than other patterns) considering the first sets of properties (e.g. 97.34% with the first three properties), has a 0.70 percentage of coverage with 20/23 selected properties, while has no instances covering all the most frequent 21, 22, and 23 (i.e. all) properties. Other patterns with a lower percentage of coverage considering the whole set of properties are: *human* (8 individuals complying with the whole pattern out of 7,622), *museum* (1/636), and *hotel* (1/427).

Table 8

Percentages of coverage of the AAA patterns properties in the KG.

Class	1 prop	2 props	1/8	1/4	1/2	all
Q5 human	99.89	99.39	[4] 90.12	[8] 39.71	[16] 2.91	[33] 0.10 (8 instances)
Q3947 house	99.91	98.92	[1] 99.91	[2] 98.92	[4] 96.39	[9] 4.88 (114 instances)
Q41176 building	99.75	97.71	[1] 99.75	[3] 94.61	[5] 77.79	[10] 0.37 (6 instances)
Q23413 castle	100	99.04	[2] 99.04	[3] 93.76	[6] 60.88	[12] 2.78 (29 instances)
Q33506 museum	99.37	90.72	[2] 90.72	[4] 70.12	[7] 33.96	[15] 0.15 (1 instance)
Q11303 skyscraper	99.83	97.17	[2] 97.17	[5] 71.38	[9] 33.44	[19] 0.33 (2 instances)
Q1343246 English country house	100	93.95	[1] 100	[3] 92.44	[5] 73.15	[11] 9.89 (59 instances)
Q3305213 painting	99.64	98.93	[3] 97.34	[6] 72.51	[11] 51.59	[23] 0 (0 instances)
Q1307276 single-family detached home	99.78	99.78	[1] 100	[3] 99.12	[5] 97.16	[11] 5.88 (27 instances)
Q27686 hotel	100	94.14	[2] 94.14	[3] 81.49	[6] 20.84	[13] 0.23 (1 instance)
Q207694 art museum	100	94.67	[3] 86.44	[5] 75.54	[10] 41.64	[21] 0.24 (1 instance)

Comparison with property constraints. The most common properties across all patterns are `wdt:P17` *country* (recommended for 9/11 patterns) and `wdt:P131` *located in the administrative territorial entity* (9/11 patterns). Neither properties have any suggested domain in the *subject type* constraint, thus we could integrate the constraints by suggesting possible classes the subject can be type of (like *building*) in the AAA domain. All ranges we suggest for the `wdt:P131` property have a `wdt:P279*` hierarchical relation with one of the classes in the *value-type* constraint (namely `wd:Q56061` *administrative territorial entity*), excluding *village* (`wd:Q532`). 3/11 ranges suggested for `wdt:P17` are directly included in the *value-type* constraint, 5/11 are subclasses of classes listed in the constraint, while 3/11 (namely *democratic republic*, *constitutional republic* and *superpower*) are absent from the *value-type* constraint list, however, for instance, more than 80% buildings have a *constitutional republic* as type of the object of the property *country*, so this range is not explicitly recommended but is actually very common in the data.

Comparison with properties for this type. 8 out of the 11 classes we build our patterns around have values for the property *properties for this type* (`wdt:P1963`). Many frequently used properties are left out from the *properties for this type*. For example, 30 properties are recommended by Wikidata for the type *painting*. 19 of these properties (like *creator*, *collection*, *height*, *width*, etc.) are also included in our *painting* pattern, and the remaining 11 properties have a number of occurrences that ranges between 63 (11% of all paintings) and 0. Instead, our pattern also recommends other 4 properties not included in the *properties for this type*: *Freebase ID* (99.65%), *Commons category* (40.78%), *BabelNet ID* (40.07%) and *Google Arts & Culture asset ID* (17.3%). Another example is the type *skyscraper*: out of the 16 *properties for this type*, 7 are also mentioned in our pattern, and the remaining 9 have been excluded based on their low frequency (from 8% to 0.17%), while very frequent properties such as *coordinate location* (97.17%) and *image* (83.86%) are not listed as *properties for this type*.

Comparison with type of wikidata property. *Wikidata property to identify artworks* links only to IDs related to specific museums, while the only IDs we select for *paintings*, based on their usage, are quite general (see Figure 8).

Wikidata property related to architecture, apart from IDs, recommends the properties *architect*[32] and *architectural style*, which are included in some of our patterns, as depicted in Figure 8. *Wikidata property for items about buildings* only mentions the property *has certification*, which is never used for buildings.
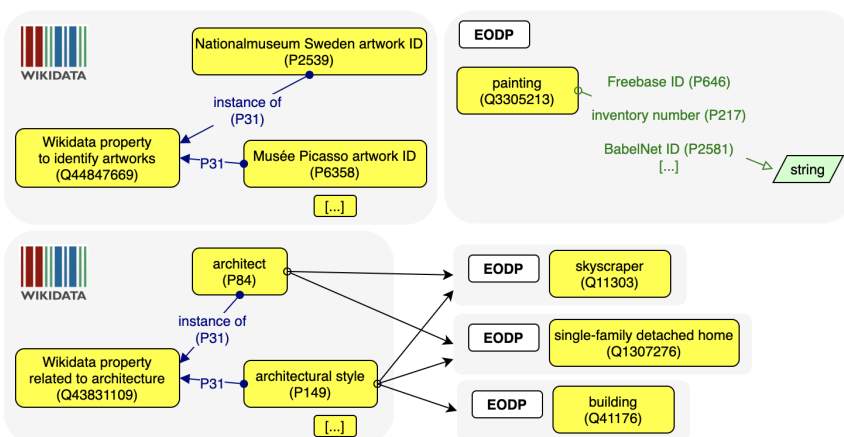

Fig. 8. Example of comparison between our patterns and the Wikidata support for describing artworks and architecture.

Comparison with properties listed in the WikiProjects. It does not exist a WikiProject addressing the Art, architecture, and archaeology (AAA) domain. The WikiProject *Archaeology*[33] is related to archeology in general, however the list of properties, which is not split based on the different classes of the subject, contains only IDs, other than the property *director of archaeological fieldwork*. The WikiProject *sum of all paintings* lists 6 important properties for paintings[34], which are all included in our pattern *painting*. There is also a list of properties related to *structures*[35], but it is not completely clear which structures it addresses; by looking at the example triples we can find hotels, bridges, skyscrapers, airports, so we could probably associate these properties with instances of *building* and subclasses. Apart from IDs, the properties that we can attribute to buildings are very specific (e.g. *structure replaced by*, which is never used neither for buildings nor hotels in our subgraph), so most of them are absent from our *structure-related* patterns, apart from *floors above ground* (included in the *skyscraper* and *hotel* patterns) and *number of elevators*, included in the *skyscraper* pattern.

The WikiProject *Museums*[36] (WPMuseum hereinafter) defines a set of properties relevant for museums, split in different topics (location, practical information, communications, etc.). We report in Table 9 a comparison between the properties recommended by WPMuseum and our method (EP *museum*), except for IDs. Our pattern includes 7 properties recommended by WPMuseum, whose percentages of occurrence range from 99% to 18%, while WPMuseum does not recommend 3 our properties, one of which is very frequently used: *image* is used for 75.79% museums, while WPMuseum recommends a more specific property, *logo image*, which is only used by 1.57% museums. In addition to *logo image*, there are other 17 properties recommended by WPMuseum but excluded from our pattern; however, even if we used a lower threshold in order to include more properties, many of them would be discarded anyway: *closed on*, *open period from* and *open period to* have 0 occurrences in the data, and other 9 properties have a percentage of occurrence between 0 and 3.

Comparison with AAA-related shapes. We could identify 10 shapes related to the architecture, archaeology and art domain from the list of Wikidata entity schemas[37]: *statue* (E99), *museum* (E125), *painting* (E130), *UNESCO*

---

[32]And the more specific *landscape architect*.

[33]https://www.wikidata.org/wiki/Wikidata:WikiProject_Archaeology

[34]https://www.wikidata.org/wiki/Wikidata:WikiProject_sum_of_all_paintings#Item_structure_to_describe_paintings_on_Wikidata

[35]https://www.wikidata.org/wiki/Wikidata:List_of_properties/work#Wikidata_property_related_to_structures

[36]https://www.wikidata.org/wiki/Wikidata:WikiProject_Museums

[37]https://wikidata.org/wiki/User:HakanIST/EntitySchemaList

world heritage site (E142), *hospital* (E187), *public artwork* (E216), *building* (E270), *runic inscription* (E276), and *street* (E317). Let us have a look at the shapes that correspond to two patterns we extract, i.e. *museum* and *painting*. The shape for museums[38] includes 33 properties: 30/33 are part of optional constraints (*zero or more*, *zero or one*), while 3/33 are declared as obligatory: *country* (wdt:P17), *located in the administrative territorial entity* (wdt:P131), *coordinate location* (wdt:P625). All these 3 obligatory properties are also recommended by our *museum* pattern. As for the optional properties, we include 2 IDs and 5 other properties that are also in the shape. Some of the identifiers included as optional in the shape – like *ISNI* and *GND ID* – would have not been filtered out with a slightly more inclusive threshold. On the contrary, other identifiers are quite uncommon, like *Museofile* (0.47%) and *GitHub username* (0.31%), or they are not present in the data, like *SUDOC authorities ID*. Finally, the shape does not include some frequent properties reported in our pattern, such as *Freebase ID* (87.74%) and *location* (34.12%).

The shape for paintings[39] has a significantly lower number of properties: it recommends 4 mandatory (*one or more*) properties – i.e. *title*, *location*, *collection*, *creator* – and one optional (*zero or one*) property, that is *inception*. All these 5 properties of the *painting* shape are also included in our pattern for *paintings*.

6.3. Comparison between the music and AAA patterns

The only class common to both music and AAA domains is wd:Q5 *human*. The music *human* pattern contains 48 properties, while the AAA *human* pattern includes 33 properties. 27 properties are common to both patterns, and they can be indeed mostly considered general, like *sex or gender*, *occupation*, *country of citizenship*, *given name*. The music *human* includes 21 additional properties. 11/21 are clearly music-specific, including properties such as *genre*, *instrument*, and *record label*, and many domain-specific identifiers, like *Spotify artist ID* and *MusicBrainz artist ID*. The AAA *human* pattern contains 7 properties in addition to those shared by humans from both domains: 4/7 are specific to the AAA domain, i.e. *RKDartists ID*, *Artnet artist ID*, *Invaluable.com person ID*, and *has works in the collection*. Instead, other properties included in either the music or the AAA *human* patterns seem to be applicable to both domains, such as *Facebook ID* and *official website* (music), *Union List of Artist Names ID* and *award received* (AAA), but they have a higher frequency in either domains.

Let us now take a look at the ranges recommended for some common properties. For the property *occupation*, in both patterns, the classes *occupation* and *profession* are recommended as ranges; however, the more specific range *artistic profession* is recommended for AAA *humans* (45.76%), while the domain-specific range *musical profession* is recommended for music *humans* (83.78%). The property *educated at* recommends as ranges, along with other general classes, *conservatory* in the music *human* pattern, and *art school* in the AAA *human* pattern. It is clear from these examples that, even in the case of general properties common to patterns from different domains, it is possible to have domain-specific recommendations of ranges.

6.4. Evolution of the music and AAA patterns across two versions of Wikidata

In this section, we report our analysis of the empirical ODPs extracted from both the music and the art, architecture, and archaeology sub-KGs, from three different versions of Wikidata: the release on which the experiments presented and discussed in the previous sections have been run (downloaded on 04-04-2022, *april 2022 version*), a release dated 6 months later (downloaded on 10-10-2022, *october 2022 version*), and a release date 9 months later than the october 2022 version (downloaded on 10-07-2023, *july 2023 version*). The classes that have been selected for building the patterns, using the same thresholds, are the same across the three releases, and the number of instances for each class is almost equal, or the difference is negligible.

Music. Out of the 7 music patterns extracted from all Wikidata versions, 2 include the same exact list of properties in the three versions, with the same – or slightly different – order of frequency. The remaining 5 patterns show small changes:

---

[38]https://www.wikidata.org/wiki/EntitySchema:E125
[39]https://www.wikidata.org/wiki/EntitySchema:E130

Table 9

Comparison between properties recommended by WikiProject Museums and properties included in our pattern for museums.

Property	Percentage of occurrence	WPM	EP
P17 country	99.37%	Y	Y
P625 coordinate location	91.35%	Y	Y
P131 located in the administrative territorial entity	87.11%	Y	Y
P856 official website	66.67%	Y	Y
P571 inception	66.35%	Y	Y
P276 location	34.12%	Y	Y
P6375 street address	18.24%	Y	Y
P18 image	75.79%	N	Y
P1435 heritage designation	16.51%	N	Y
P1619 date of official opening	15.57%	N	Y
P1612 Commons Institution page	8.33%	Y	N
P159 headquarters location	7.86%	Y	N
P1329 phone number	7.86%	Y	N
P968 email address	5.03%	Y	N
P1174 visitors per year	5.03%	Y	N
P669 located on street	4.87%	Y	N
P2851 payment types accepted	2.83%	Y	N
P576 dissolved, abolished or demolished date	2.2%	Y	N
P1037 director/manager	2.04%	Y	N
P2900 fax number	2.04%	Y	N
P154 logo image	1.57%	Y	N
P3025 open days	0.63%	Y	N
P1436 collection or exhibition size	0.31%	Y	N
P2555 fee	0.16%	Y	N
P2846 wheelchair accessibility	0.16%	Y	N
P3026 closed on	0%	Y	N
P3027 open period from	0%	Y	N
P3028 open period to	0%	Y	N

- the *human* EODP includes 2 additional identifier properties in the *october 2022 version*: *National Library of Israel J9U ID* (with an increase of 14.26% than the *april 2022 version*) and *Grove Music Online ID* (+15.47%), to which are added other 2 properties in the *july 2023 version*, that is *award received* (+1.14%) and *Film.ru person ID* (which is indeed a quite recent property that replaces the older *Film.ru actor ID*[40]).
- the *musical group* pattern also includes 2 more IDs in the october 2022 version: *Musik-Sammler.de artist ID* (+6.58%) and *Bibliothèque nationale de France ID* (+0.18%), which are present also in the july 2023 version.
- *album* recommends 2 properties (not identifiers) in addition to those from the april 2022 version: *place of publication* (+10.37%) and *title* (+1.2%), which are included in the july 2023 version too.
- the *musical work/composition* EODP is the only one that *looses* properties in the newer versions: one property is not included anymore in the october 2022 version, i.e. *followed by* (-0.31%); however, this is the first property in the list of those excluded based on the threshold; and other two properties are not included in the july 2023 version, i.e. *follows* (-0.32%) and *producer* (-0.2%).

---

[40]See https://www.wikidata.org/w/index.php?title=Property:P10302&action=history.

- the *record label* EODP includes 2 additional properties in the july 2023 version: *name* (+2.91%) and *Commons Category* (+3.92%); moreover, the newer pattern makes it evident, thanks to the change of label, that one property is being gradually superseded (*WorldCat Identities ID (superseded)*, for now -0.2%).

Across the april 2022 and the october 2022 versions, the properties that show a wider increase in their usage, due to the edits along the 6 months, are *National Library of Israel J9U ID* (*human* pattern), *Grove Music Online ID* (*human* pattern), and *place of publication* (*album*). The july version shows that the new property *Film.ru person ID* has been widely used from october 2022 to july 2023 (*human* pattern).

AAA. 4 patterns, out of the 11 patterns extracted on the art, architecture, and archaeology Wikidata subgraph, include the same set of properties across the two versions. As for the remaining 7 patterns:

- the *human* EODP includes 2 additional ID properties in the october 2022 version: *described by source* (+5.08%) and *National Library of Israel J9U ID* (+12.42%), to which are added other 2 properties in the *july 2023 version*, that is field of work (+3.2%) and AKL Online artist ID (1.84%).
- *castle* has 4 additional properties in the october 2022 release, that is: *TripAdvisor ID* (+16.36%), *described by source* (+15.67%), *official website* (+3.11%), and *Historic England research records ID* (this property was never used for castles in the april 2022 version); additionally, the july 2023 version includes also *Canmore ID* (+0.88%) and *GB1900 ID* (+12.31%).
- *hotel* recommends 4 other properties in the october 2022 version: *hotel rating* (+4.73%), *Skyscanner hotel ID* (+4.96%), *Agoda hotel numeric ID* (+3.78%), and *Booking.com numeric ID* (+4.72%); instead, no addition could be found in the july 2023 release.
- the *art museum* EODP has one additional property in the october 2022 release, that is *National Library of Israel J9U ID* (+15.23%), and other 2 properties in the july 2023 release: heritage designation (+0.83%) and Sotheby's Museum Network ID (+0.35%).
- the *museum* pattern includes two more properties in the july 2023 release, that is: *postal code* (+1.49%) and *OpenStreetMap node ID* (16.98%, never used in the two previous releases).
- the *English country house* includes also the *TripAdvisor ID* property in the latest version (+5.88%).
- the *painting* EODP recommends also the *depicts Iconclass notation* property in the latest version (+7.95%).

We notice that, in the human EODP, from both the AAA domain and the music domain, one of the properties that most increased in their usage is *National Library of Israel J9U ID*. In the AAA domain, the same property is more used also with *art museums*. Other properties that have undergone a relevant increase are *TripAdvisor ID* and *described by source* (*castle*).

7. Applications to knowledge graphs other than Wikidata

In the previous sections, we presented how empirical patterns can be extracted from the Wikidata knowledge graph, which lacks an explicit and formal ontology, but provides some constraints and guidelines for its usage. Here, we briefly discuss how such patterns could benefit knowledge graphs other than Wikidata.

In the case of a knowledge graph that is completely devoid of an ontology, with no rules or constraints on the use of properties and classes, our EODPs may play a fundamental role in guiding a user when trying to navigate/reuse the knowledge graph, and may be a useful starting point to the development of a formalised ontology from the knowledge graph.

Our method extracts patterns from the data level without taking into account an ontology, however, it can be useful even when a strongly formalised ontology does exist. Indeed, it may suggest possibly missing axioms in the input ontology, or *wrong/irrelevant* axioms that do not reflect the actual usage in the data, giving insights on how the data (the empirical ODPs) match the actual intent of the ontology designers.

Let us take the ontology network and knowledge graph ArCo [21], which we contributed to, as an example. The ArCo ontology network has been developed following the pattern-based eXtreme Design methodology [22]. One of the ODPs modelled within the ontology is built around the class `a-cd:Acquisition`[41]: a situation in which

---

[41] @prefix a-cd: <https://w3id.org/arco/ontology/context-description/>

a cultural property is acquired, passing from an owner to another. On the left side of Figure 9, you can find the graphical diagram the ontology designers generated for documentation purposes, and the axioms that are actually declared within the ontology. On the right side, the empirical ODP, that we would extract with our method from the ArCo knowledge graph version 1.0[42], is depicted.
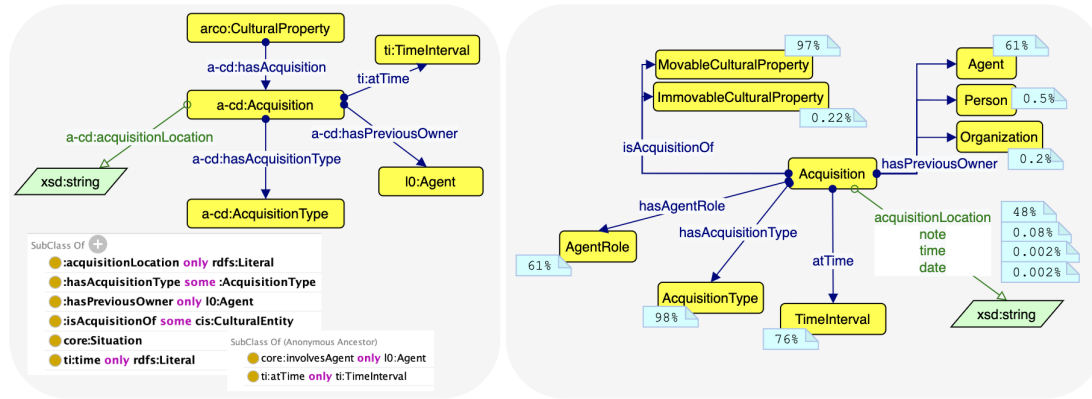

Fig. 9. Comparison between ArCo's top-down ODP for acquisitions and the EODP extracted from ArCo's KG.

First of all, the EODP provides a useful *prioritization* of the properties needed to describe an acquisition, based on the specific knowledge graph taken into account. For instance, the link to an `a-cd:AcquisitionType` (98%) seems to be more necessary than the link to the time interval when the acquisition happened (76%). Moreover, it includes more precise suggestions for the range of a property according to the usage, while the ontology is rightfully kept at a more general level. For example, the ArCo ontology contains an existential restriction stating that an instance of the class `a-cd:Acquisition` should be linked to at least one `cis:CulturalEntity`[43] with the property `a-cd:isAcquisitionOf`. In the data, `cis:CulturalEntity` is never instantiated, while the EODP highlights that the type of cultural property that is usually (97%) involved in a change of ownership is movable (that is, an object that can be moved in various ways), as opposed to immovable (an object fastened and/or incorporated into the ground, like a building). The EODP can also point out that an axiom of the ontology may be irrelevant: `ti:time`[44] is almost never used, while `ti:atTime`, included in a restriction on the ancestor class `core:Situation`[45] is the preferred property for assigning a time interval to the acquisition. Two properties are missing in the restriction of the ontology, namely `core:note` and `dc:date`[46], but their usage in the data is, indeed, very scarce. Finally, `a-cd:acquisitionLocation` is confirmed as a relevant property (48%), however, in the data, the datatype `xsd:string` takes the place of `rdfs:Literal` of the restriction.

8. Conclusion and Future Work

In this paper, we presented a method for extracting empirical patterns from a knowledge graph, without relying on an ontology. The method takes as input a KG and builds empirical ontology design patterns (EODPs) around the classes instantiated in the KG, by formalising axioms and constraints that involve those classes. Such EODPs incorporate information about the probability of each axiom or constraint to *happen*, based on their occurrences in the KG. Data about the probability is used as a filter to define how *likely* a class or axiom should be in order to be considered relevant. When run on a knowledge graph, the extracted EODPs allow a user to *observe* the main

---

(wrt their frequency) EODPs in a (domain-specific) knowledge graph, and how the core classes of the EODPs are *described* in the actual usage through relations, and to which entities they are linked.

Experiments on two subgraphs extracted from Wikidata, on the music and the art, architecture and archaeology domains, demonstrated how these patterns can support the reuse of the Wikidata ontology, and how they can be an access point to understand the content of a knowledge graph, and possibly compare different KGs.

As future work, we would like to investigate how the empirical ODPs we can extract from a KG can be mapped to state-of-the-art ODPs, like those published on the ODP Portal[47]. Trying to develop a method that automatically identifies domain-specific properties, keeping them separate from properties associated with entities relevant to multiple domains is also an important direction forward. Moreover, we would like to test the method on knowledge graphs other than Wikidata, and possibly compare the results on the same domains of knowledge.

References

[1] D. Vrandečić and M. Krötzsch, Wikidata: A Free Collaborative Knowledgebase, *Commun. ACM* **57**(10) (2014), 78–85.

[2] A. Piscopo and E. Simperl, Who Models the World? Collaborative Ontology Creation and User Roles in Wikidata, *Proc. ACM Hum.Comput.Interact.* **2**(CSCW) (2018).

[3] F. Brasileiro, J.a.P.A. Almeida, V.A. Carvalho and G. Guizzardi, Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata, in: *WWW '16 Companion*, 2016, pp. 975–980.

[4] M. Lissandrini, T.B. Pedersen, K. Hose and D. Mottin, Knowledge graph exploration: where are we and where are we going?, *ACM SIGWEB Newsletter* (2020), 1–8.

[5] V.A. Carriero, P. Groth and V. Presutti, Towards improving Wikidata reuse with emerging patterns, in: *Proceedings of the 3rd Wikidata Workshop 2022 co-located with the 21st International Semantic Web Conference (ISWC2022), Virtual Event, Hanghzou, China, October 2022*, CEUR Workshop Proceedings, Vol. 3262, CEUR-WS.org, 2022.

[6] A. Cimmino, A. Fernández-Izquierdo and R. García-Castro, Astrea: Automatic Generation of SHACL Shapes from Ontologies, in: *ESWC*, Lecture Notes in Computer Science, Vol. 12123, Springer, 2020, pp. 497–513.

[7] H. Knublauch, SHACL and OWL Compared, 2017.

[8] I. Boneva, J. Dusart, D. Fernández-Álvarez and J.E.L. Gayo, Shape Designer for ShEx and SHACL constraints, in: *Proceedings of Posters & Demonstrations, Industry, and Outrageous Ideas - ISWC 2019*, M.C. Suárez-Figueroa, G. Cheng, A.L. Gentile, C. Guéret, C.M. Keet and A. Bernstein, eds, CEUR Workshop Proceedings, Vol. 2456, CEUR-WS.org, 2019, pp. 269–272.

[9] K. Rabbani, M. Lissandrini and K. Hose, SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption, in: *Companion of The Web Conference 2022*, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman and L. Médini, eds, CEUR Workshop Proceedings, Vol. 2043, CEUR-WS.org, 2022.

[10] D. Fernandez-Álvarez, J.E. Labra-Gayo and D. Gayo-Avello, Automatic extraction of shapes using sheXer, *Knowledge-Based Systems* (2021), 107975. doi:10.1016/j.knosys.2021.107975.

[11] N. Mihindukulasooriya, M.R.A. Rashid, G. Rizzo, R. García-Castro, O. Corcho and M. Torchiano, RDF shape induction using knowledge base profiling, in: *33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1952–1959.

[12] N. Mihindukulasooriya, M. Poveda-Villalón, R. García-Castro and A. Gómez-Pérez, Loupe - An Online Tool for Inspecting Datasets in the Linked Data Cloud, in: *Proceedings of the ISWC 2015 Posters & Demonstrations Track*, CEUR Workshop Proceedings, Vol. 1486, CEUR-WS.org, 2015.

[13] B. Spahiu, A. Maurino and M. Palmonari, Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL, in: *Proceedings of the 9th Workshop of Ontology Design and Patterns (WOP 2018) co-located with the 17th International Semantic Web Conference (ISWC 2018)*, M.G. Skjæveland, Y. Hu, K. Hammar, V. Svátek and A. Lawrynowicz, eds, CEUR Workshop Proceedings, Vol. 2195, CEUR-WS.org, 2018, pp. 52–66.

[14] B. Spahiu, R. Porrini, M. Palmonari, A. Rula and A. Maurino, ABSTAT: Ontology-driven Linked Data Summaries with Pattern Minimalization, in: *SumPre co-located with ESWC*, CEUR Workshop Proceedings, Vol. 1605, CEUR-WS.org, 2016.

[15] R.A. Alva Principe, A. Maurino, M. Palmonari, M. Ciavotta and B. Spahiu, ABSTAT-HD: a scalable tool for profiling very large knowledge graphs, *VLDB Journal* (2021), 1–26.

[16] Z. Zhang, A.L. Gentile, E. Blomqvist, I. Augenstein and F. Ciravegna, Statistical knowledge patterns: Identifying synonymous relations in large linked datasets, in: *ISWC*, Springer, 2013, pp. 703–719.

[17] E. Blomqvist, Z. Zhang, A.L. Gentile, I. Augenstein and F. Ciravegna, Statistical Knowledge Patterns for Characterising Linked Data, in: *WOP co-located with ISWC*, CEUR Workshop Proceedings, Vol. 1188, CEUR-WS.org, 2013.

---

[47]http://www.ontologydesignpatterns.org

[18] L. Asprino, V.A. Carriero, C. Colonna and V. Presutti, OPLaX: annotating ontology design patterns at conceptual and instance level, in: *Proceedings of the 12th Workshop on Ontology Design and Patterns (WOP 2021) co-located with 20th International Semantic Web Conference (ISWC 2021)*, K. Hammar, C. Shimizu, H. Küçük-McGinty, L. Asprino and V.A. Carriero, eds, CEUR Workshop Proceedings, Vol. 3011, CEUR-WS.org, 2021, pp. 1–13.

[19] F. Ilievski, D. Garijo, H. Chalupsky, N.T. Divvala, Y. Yao, C. Rogers, R. Li, J. Liu, A. Singh, D. Schwabe and P. Szekely, KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis, in: *ISWC*, Springer, 2020, pp. 278–293.

[20] R. Tripodi, E. Marzi, A. Poltronieri, V. Presutti, A. Pompilio, A. Luporini, P. "van Kranenburg" and E. Daga, Plurilingual corpora containing source texts in English, French, Spanish and German (v1.0), Deliverable, 4.1, Polifonia Grant 101004746, 2021.

[21] V.A. Carriero, A. Gangemi, M.L. Mancinelli, L. Marinucci, A.G. Nuzzolese, V. Presutti and C. Veninata, ArCo: The Italian Cultural Heritage Knowledge Graph, in: *Proceedings of the 18th International Semantic Web Conference ISWC 2019, Part II*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I.F. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Lecture Notes in Computer Science, Vol. 11779, Springer, 2019, pp. 36–52. doi:10.1007/978-3-030-30796-7_3.

[22] V.A. Carriero, A. Gangemi, M.L. Mancinelli, A.G. Nuzzolese, V. Presutti and C. Veninata, Pattern-based design applied to cultural heritage knowledge graphs, *Semantic Web* **12**(2) (2021), 313–357. doi:10.3233/SW-200422.

[23] H. Hanke and D. Knees, A phase-field damage model based on evolving microstructure, *Asymptotic Analysis* **101** (2017), 149–180.

[24] E. Lefever, A hybrid approach to domain-independent taxonomy learning, *Applied Ontology* **11**(3) (2016), 255–278.

[25] P.S. Meltzer, A. Kallioniemi and J.M. Trent, Chromosome alterations in human solid tumors, in: *The Genetic Basis of Human Cancer*, B. Vogelstein and K.W. Kinzler, eds, McGraw-Hill, New York, 2002, pp. 93–113.

[26] P.R. Murray, K.S. Rosenthal, G.S. Kobayashi and M.A. Pfaller, *Medical Microbiology*, 4th edn, Mosby, St. Louis, 2002.

[27] E. Wilson, Active vibration analysis of thin-walled beams, PhD thesis, University of Virginia, 1991.