UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Towards improving Wikidata reuse with emerging patterns

Carriero, V.A.; Groth, P.; Presutti, V.

[Link to publication]

Towards improving Wikidata reuse with emerging patterns

Valentina Anita Carriero[1], Paul Groth[2] and Valentina Presutti[1]

[1]*University of Bologna*
[2]*University of Amsterdam*

Abstract

The ontology underlying Wikidata has not been formalized. Instead, its semantics emerges from the use of its classes and properties. Flexible rules and suggestions 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. In this paper, we describe a method for extracting emerging patterns from (a domain-specific portion of) Wikidata, in the form of statistically frequent domain-property-range triplets. We show the results of our experiments on a Wikidata subset addressing the music domain, and compare them with the current support present in Wikidata. These patterns can provide guidance for the use of the Wikidata ontology and its potential improvement.

1. Introduction

Wikidata[1] is a collaboratively built knowledge graph (KG) that stores structured data for its Wikimedia sister projects, including Wikipedia and Wiktionary [15]. Wikidata is edited collaboratively on a daily basis, thus contains a rich set of factual statements about entities and events in the real world. Its underlying ontology is constantly subject to change due to its frequent updates by its contributors and the way they model data.

Due to this bottom-up definition and constant evolution, it can sometimes be challenging to effectively reuse the ontology [10, 4]. While Wikidata does provide some flexible guidelines around use (see Section 2), there still remains room to provide additional, more detailed, guidance on how to use the ontology based on its *actual usage*.

Hence, in this paper, we develop a method[2] for the extraction of what we term *emerging patterns* from Wikidata. These patterns are domain specific and consist of frequent domain-property-range triplets and their usage statistics. We show how these emergent patterns provide additional information not available from existing guidelines.

The rest of this paper is organized as follows: In Section 2, we discuss existing constructs and projects that support the reuse of Wikidata. Section 3 presents relevant related work focusing on the generation of shapes or data-driven patterns. Section 4 describes our method, while Section

---

[1]https://www.wikidata.org/
[2]Both code and results are available on GitHub: https://github.com/valecarriero/wikidata-emerging-patterns

5 shows the results of our experiments on a Wikidata subset on the music domain. Finally, Section 6 compares our results with the current support present in Wikidata, and Section 7 discusses future development.

2. Motivation

We now detail various approaches for recommending how to use the Wikidata ontology.

Property constraints. The Wikidata community has defined several types of property constraints[3]: property constraints are rules on properties that specify how they should be used, with possible exceptions. These rules are flexible, aiming at guiding the editor and providing useful suggestions while injecting/editing (new) statements; they are informally defined, with no explicit logical specification, thus can still be violated/ignored. Popular property constraint types include *type constraint* and *value-type constraint*, which specify that the domain or range of a property, respectively, should be one in a list of classes. However, unlike OWL property restrictions on classes, these do not limit the applicable classes. For example, a triple with an instance of *recurrent event edition* as subject, *part of the series* (wdt:P179) as predicate, and an instance of *collection of articles* as object would conform to the property constraints of wdt:P179, even if a more appropriate range in this case would be the class *recurring event*.

Properties for this type. The property *properties for this type* (wdt:P1963) specifies the properties suggested 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 are not specified.

Type of Wikidata property. The class *Type of Wikidata property* (wd:Q107649491) is a Wikidata metaclass, i.e. a class whose instances are classes that are related to a specific set of items, domain or topic; the relation with the topic is also expressed through the property *facet of* (wdt:P1269). These classes are organised in a hierarchy, and are populated by properties that can be declared as instances of (more than) one of these classes. For example, the property *Chessgames.com player ID* is an instance of the class *Wikidata property related to chess* that is a subclass of *Wikidata property related to sport*. However, (i) this classification is an ongoing activity, thus it is far from being complete for some domains; (ii) properties relevant to a certain type may be excluded from the metaclass specific to that set of items because they are relevant also in more general domains.

Wikidata schemas. The Schemas Wikidata project[4] aims at defining schemas, expressed in the Shape Expression language (ShEx) for validating subsets of items in Wikidata, to check whether they conform to a standardised structure. At present, the Wikidata community has manually defined more than 300 schemas, which may vary considerably in size and granularity. For example, the shape *E25* for actors includes 4 constraints, and the only domain-specific constraint is related to their occupation (*actor*). Instead, the shape *E42* for authors is much more detailed, including both constraints that are valid for all humans (shape *E10*) and author-specific constraints, such as *copyright status*. Anyway, constraints usually do not express the suggested range (e.g. *notable work* in the author shape has generically an IRI as recommended range).

---

[3]https://wikidata.org/wiki/Help:Property_constraints_portal
[4]https://wikidata.org/wiki/Wikidata:WikiProject_Schemas

Properties list in a WikiProject. In the context of domain-specific projects, the community expert in that domain defines a set of properties that can be used for describing relevant entities. Each recommended property, listed in a table, is usually accompanied by the data type of its range (e.g. item, string), a description of the (usage of the) property, which in some cases also includes in plain text possible types for the range (e.g. *artistic inspiration* as range of the property *inspired by* with *written work* as domain, in the WikiProject Books[5]). This process is performed manually, and possible ranges are not always specified.

3. Related work

To help address missing guidance, there exist many approaches to generate constraints/definitions (i.e. shapes) for concepts.

Some of them (like Astrea [5]) are only based on ontologies, and do not take into account the data level. However, most methods focus on generating shapes from a set of data. Shape Designer [3] is a graphical tool for automatically building valid SHACL or ShEx constraints that are satisfied from an RDF dataset. The cardinality of the triple constraints (*exactly one*, *optional*, *at least one*, *any number*) is inferred from the data. However, if working with large KGs such as Wikidata, there is a need to put a limit on the number of query results. Indeed, [11] shows that existing methods are not able to handle the scale of large KGs like Wikidata, crashing with KGs with a few millions triples[6]. sheXer [6] is an automatic shape extractor able to extract shapes – serialised in both ShEx and SHACL – by mining the graph structure and exploring the neighborhood of predefined target nodes. A trustworthiness score allows to filter infrequent constraints and sort/merge the inferred constraints for constructing the resulting shapes. Finally, some methods exploit knowledge graph profiling, which focuses on producing concise and meaningful summaries of RDF knowledge graphs, for building shapes. [9] presents a data-driven approach that, based on machine learning techniques, aims at automatically generating RDF shapes, as collections of validation rules. Profiled RDF data are used as features, exploiting the Loupe tool[7] [8], which provides information about the frequency of triple patterns (in the form $\langle subjectType, predicate, objectType \rangle$) that appear in a dataset. In [12], the profiles generated by ABSTAT are converted into SHACL shapes related to the instances of a specified target class, which can be updated and corrected by a human user. ABSTAT [13, 1] is a profiling tool that generates a *semantic profile*, starting from a knowledge graph and optionally an ontology used in the KG: this profile is composed of *Abstract Knowledge Patterns* (AKPs), associated with their occurrences, where *subjectType* is the most specific type of the subject and *objectType* is the most specific type of the object, excluding more generic redundant patterns by using the ontology, if any.

As highlighted in [11], all approaches supporting automatic generation of shapes produce a large number of shape constraints such that it is non-trivial to verify their validity. Moreover, in most cases no constraint is generated for non-literal objects (e.g. to indicate that objects for a property should be of a specific type). Some tools currently used by the Wikidata community for

---

inserting new data suffer from a similar problem: for instance, Recoin[8] recommends properties for a class based on their frequency in the data, and reports frequent properties that are missing for instances of a specific type, but lacks information on the appropriate ranges.

The closest work to ours, based on statistical measures similar to the ones used for generating data-driven shapes, is described in [16, 2]. The authors develop a method for extracting Statistical Knowledge Patterns (SKPs) from KGs. An SKP is expressed in OWL and constructed around one main class from an ontology: it enriches the properties and axioms involving the class from the ontology with properties and axioms that can be induced from statistical measures. The most frequent (based on a threshold) properties in the data are selected, and the appropriate range(s) is/are provided if they are not explicitly asserted in the ontology. A catalogue with 34 SKPs extracted from a version of DBpedia is online[9], but the method described in the paper has not been published, so it is not possible to reproduce their results. Moreover, no metadata about the actual usage of the selected properties is present in the SKPs.

4. Method

We now describe our method for extracting emerging patterns from Wikidata. An overview of the method is shown in Figure 1.
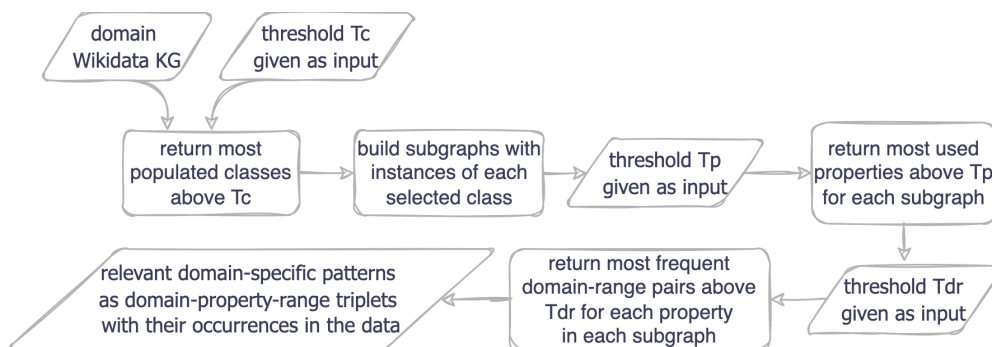
Figure 1: Method for extracting emerging patterns from a domain Wikidata KG.

Select relevant entities from the Wikidata subgraph. The first step of the method takes as input the domain subgraph and counts the number of instances for each instantiated class of the graph, i.e. it counts all the `wdt:P31` triples for each class. Then, a threshold is given as input, and allows to filter out all the classes whose instances fall below the given threshold. The selected classes are used to generate the emerging patterns.

The threshold is based on the absolute distance between the number of instances of a given class and the number of instances of the most instantiated class (i.e. the maximum number in the distribution of counts). This distance is then normalized by dividing the result by the maximum value, so that the threshold T_c falls within the range $[0, 1]$, such that the more the threshold is close to 1, the more classes will be selected: if T_c is equal to 0, it means that only

---

the most instantiated class will be selected (the distance between the count of a given class and the maximum count must be smaller or equal to 0); if T_c is equal to 1, it means that all classes will be selected (the distance between the count of a given class and the maximum count must be smaller or equal to the maximum count).

Extract a subgraph for each of those entities. Once the list of classes is obtained, we build a subgraph for each class, by selecting from the domain subgraph only the triples with an instance of the given class – or one of its subclasses – as subject. For example, for the class *album*, a subgraph containing all the triples about instances of album or subclasses of album is built.

Most frequent properties for each class. At this stage, the occurrences of all the properties instantiated in each subgraph is computed, i.e. the number of distinct instances that have at least one triple involving that property is counted. Then, for each subgraph, we select only the most common properties based on a threshold T_p given as input. Notice that in this step we discard the *instance of* wdt:P31 and *subclass of* wdt:P279 properties from the statistics.

Most frequent ranges for each frequent property. We compute all the domain-property-range triplets in each subgraph, where *domain* is the type (wdt:P31) of the subject and *range* is either the type of the object (when the object is a *wikibase-item*) or the wikidata data type (e.g. *time*, *monolingual text*).[10] The occurrences of each triplet are then counted to find the most common domain-range pairs for each property. Again, a threshold T_{dr} selects the most common domain-range pairs for each one of the most common properties selected in the previous step.

5. Results

In this section, we discuss the results of applying our method to the Wikidata subgraph on the music domain (see below)[2].

5.1. Input

In order to deal with the size of Wikidata, we use the recently developed tool Knowledge Graph Toolkit (KGTK)[11] [7]. KGTK is a Python library for easy manipulation of KGs, a comprehensive framework designed for ease of use, scalability, and speed. This tool allows us to avoid reaching the query timeout limit on the SPARQL endpoint for some of our queries. We work with a json dump of Wikidata[12], downloaded on 04-04-2022. We focus on a specific domain represented in Wikidata in order to extract domain-dependent patterns, and to handle a more manageable subgraph of Wikidata. While we choose to work on the music domain, the method can be applied to any domain. The extraction of instances related to the music domain is based on a list of WordNet and BabelNet synsets identified as belonging to the music domain, according to BabelDomains [14]. Then, the Wikidata subgraph on music is extracted by selecting each triple where the Wikidata music instance is in the subject position. The different thresholds have been chosen after running some experiments, in order to extract reasonably representative patterns from the music domain.

---

[10]See https://www.wikidata.org/wiki/Special:ListDatatypes
[11]https://kgtk.readthedocs.io/en/latest/
[12]https://dumps.wikimedia.org/wikidatawiki/entities/

5.2. Wikidata emerging patterns on music

Most populated classes: music patterns. The threshold T_c we use for the Wikidata music 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 6,043 classes, ~6,000 of which have less than 200 instances). Clearly, the same entity can be an instance of more than one class.

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

Table 1 lists the 7 classes around which we build our patterns, along with their number of instances and the number of triples with an instance of the class (or one of its subclasses) as subject. The most relevant entities in the Wikidata music domain include both agents (human, musical group) and objects (single, album, musical work, extended play, record label). Notice that wd:Q134556 *single* and wd:Q169930 *extended play* are not subclasses of wd:Q105543609 *musical work/composition* in the Wikidata hierarchy (wdt:P279*). By looking at the ratio between the number of instances and the number of triples, at first sight, we can observe that e.g. humans are more well *described* with facts than albums, considering that the number of respective instances is roughly equal.

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 ~21. In Table 2 you can find the actual 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. Notice, the number of recommended properties is not directly proportional to the number of triples in the subKG: for instance, musical groups have 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 (except for IDs) are: wdt:P136 *genre*, which is recommended for all patterns, and wdt:P264 *record label*, present in all patterns but record label.

Recommended ranges for each property. For selecting the most frequent ranges for each recommended property, we set the threshold to 0.5. Datatype properties will have only one range in any case. Table 2 reports the number of triplets $\langle d, p, r \rangle$ – that is, the domain d and range r pairs for each recommended property p – selected for each pattern. The average number of triplets across all patterns is ~29. Since the same property can be involved in more than

Table 2

Statistics of selected properties and triplets for each pattern.

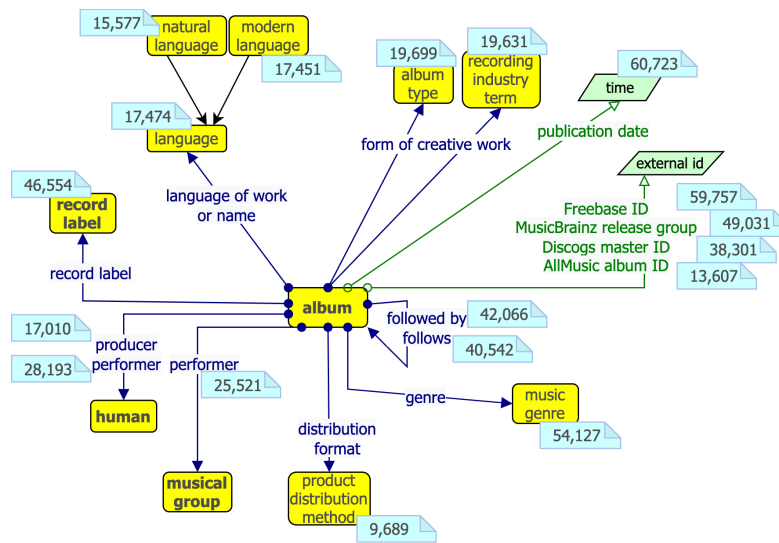| Class	Properties	Occurrences		Triplets
		max	min	
Q5 human	48	63,583	9,543	63
Q482994 album	14	61,772	11,735	18
Q215380 musical group	33	22,423	3,474	38
Q134556 single	15	20,860	5,076	22
Q105543609 musical work/composition	17	13,916	2,204	29
Q169930 extended play	10	3,793	650	12
Q18127 record label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

Figure 2: The album pattern.

one pattern, a property can have different recommended ranges based on the specific pattern, except for datatype properties. That is, ranges recommendations are local to the pattern. For instance, both album and single patterns include the property wdt:P155 *follows*, with *album* and *single* as range, respectively.

Example: the album pattern. In Figure 2 we provide a graphical representation of the pattern for albums. Each domain-property-range triplet is associated with the number of instances in the Wikidata subKG that comply with that triplet. Based on the 0.5 threshold, most properties have only one recommended range. However, the *performer* can be both a human and a musical group, and the 3 selected ranges of *language of work or name* have a subclass-of relation. As you can notice, 4 recommended properties link to other frequent patterns as recommended ranges (*record label*, *human*, *musical group*).

6. Discussion

Patterns coverage. In order to understand how the extracted patterns are populated in the Wikidata subKG, we report in Table 3[13] the percentage of the total instances covering different (increasing) subsets of recommended properties. No pattern has a 100% coverage even considering only the most frequent property; in some cases, the set of the two most common properties has a percentage of coverage very close to the first property (e.g. *human*), while in others (e.g. *musical work*) it decreases significantly. In 4 out of 7 patterns, the instances populating the first half (1/2) of the recommended properties are between the 35 and ∼58% of the total number of instances; instead, humans musical works and groups have already a very low coverage. The most populated pattern (considering all properties), wrt the total number of instances, is *extended play* (112/3,816), followed by *album* (845/63,213) and *musical group* (327/25,016). The pattern with the lower percentage of coverage is *musical work* (1/14,600). The coverage percentages might appear very low, however this is not surprising: by using the 0.85 threshold, we include all properties that are used by at least 15% of the total number of instances. If the least common property is used for e.g. 625/3,577 instances (see *record label*), it is not surprising that the intersection of instances with all 11 properties is equal to 28 instances.

Comparison with property constraints. Let us consider the most common properties across all patterns. The domains and ranges we suggest for the properties *genre* (7/7 patterns) and *record label* (6/7) are all included in the *type* and *value-type* constraints of the two properties – still, in some cases, the constraints suggest a superclass as range, e.g. work in place of the more specific musical work. However, as we explained in Sec. 2, the correct pairs of domain and range cannot be specified, thus our method can integrate these constraints by suggesting that e.g. *music genre* is *more correct* as range of *genre* with *record label* as domain, than e.g. *criticism* (included in the value-type constraint of *genre*), which never occurs in the data. Moreover, not all properties define these constraints: e.g. *follows* (4/7 patterns) has no *type/value-type* constraints.

Table 3
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 the 7 most populated classes in the music Wikidata subgraph, we performed a comparison between the properties included

---

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

in our patterns and the properties included as value of the property *properties for this type* (`wdt:P1963`) for those classes. We manually observed that some properties highly instantiated in the data are not listed as properties for this type, while all the properties suggested as properties for the type and excluded from our patterns are significantly less frequent, and sometimes have a very low number of occurrences. Take musical group (`wd:Q215380`) as an example[14]. Identifier properties such as *Freebase ID*, *MusicBrainz artist ID* and *Discogs artist ID* are widely used (about 81, 75 and 74 % respectively), but not included as properties for this type. Instead, IDs less frequently associated with musical groups in the data (e.g. *Apple Music artist ID (U.S. version)*, ∼6.5%), hence filtered out from our pattern, are recommended. As another example, properties such as *influenced by* and *award received* are recommended, while they are discarded in our pattern because of their very low frequency (less than 0.5 and 2 % respectively). 10 out of the 18 properties recommended as properties for this type are also included in our pattern.

Comparison with type of wikidata property. A subclass of *Type of Wikidata property* is specifically dedicated to *properties related to music* (`wd:Q27525351`) and includes as instances properties such as music-related IDs (e.g. *YouTube playlist ID*) and other specific relations (e.g. *composer*, *performed at*). However, it is not specified which are the possible domains of such properties, so it is difficult to understand which properties to use for a user that needs to model a specific musical entity. 24 subclasses of *Wikidata property related to music* are specific to some musical entities (e.g. music genres, songs, instruments). However, 14 of them group only identifiers, e.g. for songs and bands. Even considering just the IDs, our patterns are more complete and representative. For instance, the class *Wikidata property to identify bands*, *facet of musical group*, includes only the property *Encyclopaedia Metallum band ID* (`wdt:P1952`). 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 are difficult to identify for reuse based on the *Wikidata property* classes: for instance, *genre*, which is widely used for musicians (about 50%) and musical works (about 60%), is included in both the human and musical work/composition patterns, while it can only be found under the more general classes *Wikidata property for items about people* and *for items about works*.

Comparison with properties listed in the WikiProject Music. The WikiProject Music[15] (WPM hereinafter) defines a set of properties for 6 relevant entities in the domain: human, musical ensemble, musical work, track, release, record label. Apart from *human* and *record label*, our patterns do not perfectly overlap: musical ensemble *vs* musical group (the latter being the most populated subclass of the former); musical work *vs* musical work/composition (musical work has very few direct instances, while being a class with plenty instantiated subclasses e.g. song); release, which groups together its subclasses album, single and extended play. However, it is still useful to try to compare them. Let us take *record label* as an example: the WPM recommends 4 properties in addition to 13 identifiers. Our pattern contains 11 properties, 6 of which are identifiers. Apart from *instance of* `wdt:P31`, which we always exclude from our

---

[14]https://github.com/valecarriero/wikidata-music-odp/blob/main/results/supplementary_materials/properties_forthis_type/Q215380_properties_comparison.tsv

[15]https://wikidata.org/wiki/Wikidata:WikiProject_Music

patterns, and is included by WPM, we report in Table 4 a comparison between the properties recommended by WPM and our method (EP), except for IDs. It can be observed that our pattern is more inclusive (6 *vs* 3 properties). We can detect all properties recommended by WPM, while WPM does not include *inception*, even if it is the second most frequent property. In our pattern *country* (recommended as range of the property *country* by WPM) is the most frequent range, but we also suggest 6 more specific classes such as *sovereign state*.

Table 4
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 5
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

Now, let us have a look at the Release properties in WPM and our patterns album (A), single (S) and extended play (P) (Table 5). 6 properties recommended by WPM for releases are included in all our patterns, 3 properties are included in a subset of our patterns, while 9 properties are not included. However, e.g. *composer* is used only 6, 186 and 1602 times for extended plays, albums and singles, respectively; *instrument* is never used for these entities (instead, it is included in the *human* pattern); *working title* is used only twice for albums, while the property *title* (wdt:P1476) is much more used (9,007 occurrences).

Comparison with music-related shapes. The only music-related shapes we were able to manually identify from the list of Wikidata entity schemas[16] are: E66 *music composition by W.A.Mozart*, and E248 *album*, so there is room for improvement wrt coverage of the music domain. The album shape recommends 18 properties as mandatory (*exactly one/at least one*): 7/18 are included also in our pattern; while some recommended properties may be statistically relevant (e.g. *title*: 9,007/63,213 occurrences) and would have been included in our pattern with a little higher threshold, other properties have very few occurrences that do not justify their obligatory use (e.g. *review score* 722 and *distributed by* 299). Instead, e.g. the *producer* property

---

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

(with *any number* as cardinality constraint) is much more used (18,362) and is included in our pattern.

7. Conclusion and future work

In this paper, we presented a method for extracting emerging patterns from Wikidata, in the form of statistically frequent domain-property-range triplets. Experiments on the music domain, demonstrated how these patterns can support the reuse of the Wikidata ontology. These patterns can also support current WikiProjects aiming at defining properties that can be used by domain-specific infoboxes (as shown for the WikiProject Music), and could work as an input for new WikiProjects on under-documented subject areas.

As future work, we would like to transform these patterns into OWL ontology design patterns, by defining the appropriate axioms for each relevant triplet; in this way, our Wikidata patterns could be mapped to state-of-the-art ODPs (e.g. from http://www.ontologydesignpatterns.org/). Moreover, we would like to test the method with domains other than music. Such analysis could suggest ways to identify domain-specific properties and keep them separate from properties associated with entities relevant to multiple domains. Extending this method to KGs other than Wikidata is also an important direction forward.

References

[1] Renzo Arturo Alva Principe et al. "ABSTAT-HD: a scalable tool for profiling very large knowledge graphs". In: *VLDB Journal* (2021), pp. 1–26.

[2] Eva Blomqvist et al. "Statistical Knowledge Patterns for Characterising Linked Data". In: *WOP co-located with ISWC*. Vol. 1188. CEUR-WS.org.

[3] Iovka Boneva et al. "Shape Designer for ShEx and SHACL constraints". In: *ISWC (Posters & Demonstrations, Industry, and Outrageous Ideas)*. Vol. 2456. CEUR-WS.org, 2019, pp. 269–272.

[4] Freddy Brasileiro et al. "Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata". In: *WWW '16 Companion*. 2016, pp. 975–980.

[5] Andrea Cimmino, Alba Fernández-Izquierdo, and Raúl García-Castro. "Astrea: Automatic Generation of SHACL Shapes from Ontologies". In: *ESWC*. Vol. 12123. 2020, pp. 497–513.

[6] Daniel Fernandez-Álvarez, Jose Emilio Labra-Gayo, and Daniel Gayo-Avello. "Automatic extraction of shapes using sheXer". In: *Knowledge-Based Systems* (2021), p. 107975.

[7] Filip Ilievski et al. "KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis". In: *ISWC*. Springer. 2020, pp. 278–293.

[8] Nandana Mihindukulasooriya et al. "Loupe-An Online Tool for Inspecting Datasets in the Linked Data Cloud." In: *ISWC (Posters & Demos)*. Vol. 1. 1. 2015, p. 2.

[9] Nandana Mihindukulasooriya et al. "RDF shape induction using knowledge base profiling". In: *33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 1952–1959.

[10] Alessandro Piscopo and Elena Simperl. "Who Models the World? Collaborative Ontology Creation and User Roles in Wikidata". In: *Proc. ACM Hum.Comput.Interact.* 2.CSCW (Nov. 2018).

[11] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. "SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption". In: *The Web Conference*. 2022.

[12] Blerina Spahiu, Andrea Maurino, and Matteo Palmonari. "Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL". In: *WOP co-located with ISWC*. Vol. 2195. CEUR-WS.org, 2018, pp. 52–66.

[13] Blerina Spahiu et al. "ABSTAT: Ontology-driven Linked Data Summaries with Pattern Minimalization". In: *SumPre co-located with ESWC*. Ed. by Andreas Thalhammer, Gong Cheng, and Kalpa Gunaratna. Vol. 1605. CEUR Workshop Proceedings. 2016.

[14] Rocco Tripodi et al. *Plurilingual corpora containing source texts in English, French, Spanish and German (v1.0)*. Deliverable 4.1. Polifonia Grant 101004746, 2021.

[15] Denny Vrandečić and Markus Krötzsch. "Wikidata: A Free Collaborative Knowledgebase". In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85.

[16] Ziqi Zhang et al. "Statistical knowledge patterns: Identifying synonymous relations in large linked datasets". In: *ISWC*. Springer. 2013, pp. 703–719.