Inference of Latent Shape Expressions Associated to DBpedia Ontology

Daniel Fernández-Álvarez[1], Herminio García-González[1], Johannes Frey[2], Sebastian Hellmann[2], and José Emilio Labra Gayo[1]

[1] Department of Computer Science, University of Oviedo, Oviedo 33007, Spain
{danifdezalvarez,herminiogg}@gmail.com, labra@uniovi.es
[2] Agile Knowledge Engineering and Semantic Web, University of Leipzig, 04109 Leipzig, Germany
{frey,hellmann}@informatik.uni-leipzig.de

Abstract. In order to perform any operation in an RDF graph, it is recommendable to know the expected topology of the targeted information. Some technologies have been developed in the last years to describe the expected shapes in an RDF graph, such as ShEx or SHACL. In general, a domain expert can define the expected shapes in a graph, but there are some scenarios in which the schema cannot be predicted a priori, but it emerges at the same time that the graph is filled with new information (the shapes are latent). We have developed a prototype which is able to infer shapes of classes in a knowledge graph and used it with classes of DBpedia ontology. We serialize our results using ShEx.

Keywords: RDF · ShEx · Inference · DBpedia · Knowledge Graph · Schema

1 Introduction

The most common way to perform queries against any Resource Description Framework (RDF) store is using SPARQL. In order to perform an effective SPARQL query against a Knowledge Graph (KG), one may need to know the expected topology of the KG. A wrong pick of properties, data-types or classes may cause a certain query to ignore relevant information or to update data in a way that does not fit with the current KG's topology.

Ontologies define the meaning and the correct usage of properties and classes, but they are not intended to specify the expected shape of a group of nodes in the context of a specific KG. In the RDF world, Shape Expressions (ShEx) [6] and Shapes Constraint Language (SHACL)[4] have been proposed for describing and validating RDF.

Usually, the topology of a KG can be designed or predicted by domain experts in controlled scenarios. However, there are situations in which a KG does not have a planned schema, but the shapes emerge while the content keeps growing. Insightful examples of that are community-driven approaches such as DBpedia

or Wikidata. In those cases, as suggested in [2], discovering the latent schemata associated to classes by applying inference can be useful in several ways:

- **Guideline for users**. Knowing the shape associated to a class is helpful to build queries about their instances.
- **Measure of data quality**. The process of inference may produce shapes with different levels of trustworthiness w.r.t how homogeneously the knowledge is represented. That trustworthiness may be used as a data quality measure or as an input for some methods of error detection [5], especially w.r.t mistypings or absence of typings in Assertion Box (A-Box) terms.

We implemented a prototype which is able to infer Shape Expressions associated to the classes in a KG and applied it on the English chapter of DBpedia[3]. Our prototype calculates a score of how trustworthy the constraints inferred in the shapes are, i.e., how many of the total of instances really conform to it. Then, it serializes the results using ShEx. Some other works have already studied emergent schemata in RDF sources[3] and serialization or visualization of this information[1]. The novelty of our approach consists of the usage of ShEx. Despite SHACL is a W3C recommendation, at this stage we have chosen to work with ShEx instead of SHACL because ShEx presents a more concise human-readable syntax. Nevertheless, we are presenting a work in progress. Future versions of our prototype will support both technologies.

2 Shape Inference

Our prototype receives as input a list of class URIs from a KG and infers a shape for each class. It works with the following workflow: 1) Find all the instances of the target classes; 2) For each class, find all the triples whose subject is one of its instances and use them all to build a profile of the class, consisting of a list of triple constraints[4]; and 3) turn each profile into a ShEx. Some configuration can be provided to filter some tripe constraints from the results.

Our prototype has a linear complexity related to the total number of triples of the target classes' instances. However, the underlying algorithm can be parallelized or even adapted to a server process, triggering some changes in a limited number of schemata when there is some update.

Listing 1.1 shows an example of a small RDF graph about countries, and Listing 1.2 presents the shape that our prototype infers from it[5]. Every triple constraint induced in a shape is associated with a percentage that indicates how many instances of the target class conform with the constraint. The common case

[3] The used source code as well as an extended explanation of our experiments is available at https://github.com/DaniFdezAlvarez/dbpedia-shexer

[4] Triple constraints are the basic building block in ShEx. They are composed of a property, a node constraint and a cardinality.

[5] The prefixes employed in this paper are the common ones that can be resolved by the service http://prefix.cc/

in real scenarios is that not all of the instances conform with a given constraint rule, with the exception of the constraint *rdf:type [:nameOfTheClass]*, which they all share.

Listing 1.1. RDF example graph

```
dbr:Spain rdf:type dbo:Country ;
   dbp:capital dbr:Madrid;
   rdfs:label "Spain" ;
   rdfs:label "Kingdom of Spain" .

dbr:France rdf:type dbo:Country ;
   dbp:capital dbr:Paris ;
   rdfs:label "France" .
```

Listing 1.2. Example Country Shape

```
:Country
{
  rdf:type [dbo:Country] ;  # 100%
  dbp:capital IRI ;         # 100%
  rdfs:label xsd:string +   # 100%
       # 50% have cardinality {1}
}
```

Listing 1.3 shows an example of Country Shape inferred by analyzing the actual content of DBpedia.

Listing 1.3. Country Shape (trustworthiness of 80%)

```
:Country {
        rdf:type [dbo:Country] ;          # 100.0 %
        dbo:wikiPageID xsd:integer ;      # 97.108 %
        owl:sameAs IRI +;                 # 96.933 %
        foaf:name xsd:string +;           # 96.758 %
        dcterms:subject IRI +;            # 96.028 %
        dbo:dissolutionYear xsd:gYear +;  # 83.148 %
                        # 82.593 % have cardinality {1}
        dbo:foundingYear xsd:gYear +;     # 82.009 %
                        # 81.454 % have cardinality {1}
        dbp:continent rdf:langString +    # 80.607 %
                        # 80.344 % have cardinality {1}
}
```

The main features of our prototype are the following:

Trustworthiness score. Every triple constraint inferred is associated with the relative amount of instances that fit with it. We provide that information in a comment. That allows for sorting the constraint w.r.t. its trustworthiness, as well as filtering constraints that are not frequent enough. The threshold to accept or reject a constraint w.r.t how trustworthy it is can be configured.

Literals and IRIs recognition. All kinds of literals are recognized and treated separately when inferring the constraints. In case a literal is not explicitly associated with a type in the original KG, *xsd:string* is assumed. If the object of a triple is an IRI, the macro *IRI* is used to represent it in the inferred constraints.

Special treatment of rdf:type*.* The only exception to the previous feature happens when analyzing triples whose predicate is *rdf:type*. In those cases, we create a triple constraint whose object is a value set containing a single element, which is the actual object of the original triple. We introduced this exception just for *rdf:type* because we consider that the information related to typing is specially relevant for the context. Two classes with a high number of overlapped instances may have also a highly overlapped schema. Future versions of this prototype will allow to customize which properties point to value sets.

Cardinality management. Some of the triples of a given instance may fit in an infinite number of constraint triples with the same predicate and object but different cardinalities. For example, if an instance has a single label, that makes it fit with infinite triple constraints of the form {*rdfs:label xsd:string C*}, where C can be {1}, +, {1,2}, {1,3},... At this stage, our prototype considers rules with exact cardinality or + closure. We avoid using rules with * closure because every triple constraint with that cardinality will match with 100% of the instances, not mattering the predicate or the object of the constraint.

When serializing the shapes, our prototype can be configured to prioritize the least specific cardinality or the most specific one if its trustworthiness is high enough. Information about cardinality which is not given in the constraint itself is provided through comments.

3 Conclusions and Future Work

We have applied automatic inference over DBpedia to discover latent Shapes associated to classes of the DBpedia ontology using a statistical approach. We have serialized the latent shapes using ShEx, which can be useful as a guideline on how to manipulate the data. Our approach associates a score of trustworthiness to each rule, so it can also be used as a metric of homogeneity of the dataset.

We are presenting a work in progress research. The algorithm underlying our prototype can be extended with extra features, including more complex inferences, such as inter-shape referencing or more precise cardinalities; regular expressions for some literals; or generation of serializations different to ShEx, such as SHACL or example SPARQL queries associated to each class.

Acknowledgments. This work is partially funded by the Spanish Ministry of Economy and Competitiveness (Society challenges: TIN2017-88877-R)

References

1. Dudáš, M., Svátek, V., Mynarz, J.: Dataset summary visualization with lodsight. In: International Semantic Web Conference. pp. 36–40. Springer (2015)
2. Fernández-Alvarez, D., Labra-Gayo, J.E., Garcıa-González, H.: Inference and serialization of latent graph schemata using shex. In: SEMAPRO 2016, The Tenth International Conference on Advances in Semantic Processing. IARIA
3. González, L., Hogan, A.: Modelling dynamics in semantic web knowledge graphs with formal concept analysis. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web. pp. 1175–1184. International World Wide Web Conferences Steering Committee (2018)
4. Knublauch, H., TopQuadrant, Inc., Kontokostas, D., University of Leipzig: Shapes constraint language (shacl). W3C Recommendation **11**, 8 (2017)
5. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2017)
6. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: an rdf validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems. pp. 32–40. ACM (2014)