Identification of SHACL Constraints from a Knowledge Graph

Erhard Eibl, Siyabend Sakik, Niels Schneider,
Oya Beyan, Nils Lukas

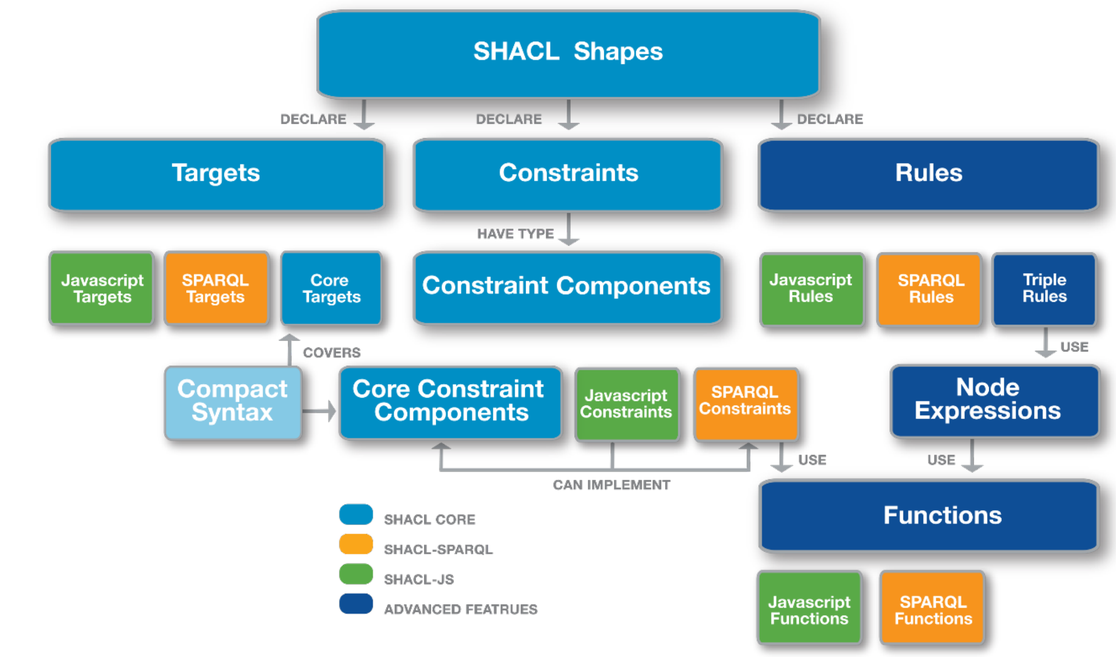
RWTH 5 | RWTH AACHEN UNIVERSITY

Overview

This project is about finding patterns in Knowledge Graphs and to generate SHACL constraints for entities. The goal is to maintain and improve consistency for instance in collaborative databases. Our premise is that entities and property patterns prevalent among them in a given graph can be used for generalization and categorization of new entities added into a graph. Therefore the algorithm is able to identify missing or false data with a certain confidence value. This is useful to automatically ensure compliance to a schema that is implicitly present in a database. For example, if every former president of the United States had a name, the algorithm assumes that having a name is a necessary condition for being president of the United States.

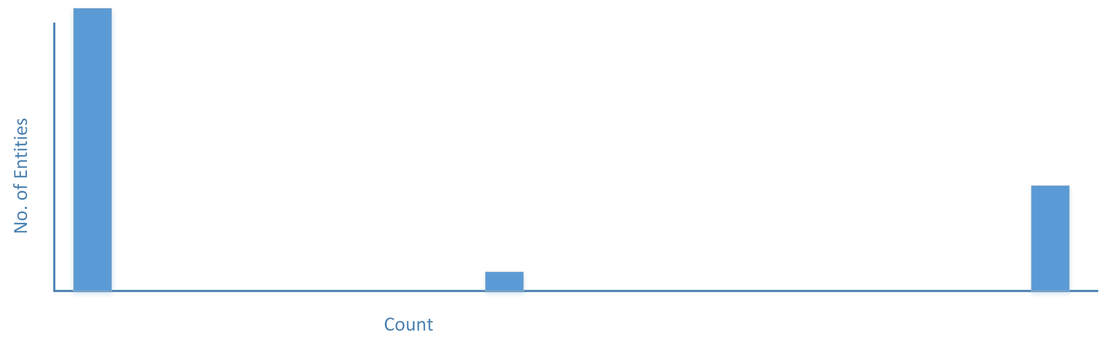
Why SHACL?

SHACL (Shapes Constraint Language) is designed to specify schemas for entities more conveniently than for example with OWL. A shape specifies targets, which correspond to the nodes to be constrained and the constraints make up a basic schema that these nodes should satisfy. Using conformance checking tools you can validate a graph against a schema. A graph passes a validation if and only if all instances conform to all set constraints. We are able to generate shape files automatically based on a statistical data mining approach and we define targets using the SHACL-SPARQL extension, which allows us to constrain arbitrary groups of instances.



Limitations

The main limitation of a statistical approach concerns the distinction between valid and invalid outliers. For instance, in a scenario where most people have exactly two names (e.g. first and last), but one person incorrectly has 100 names. Naively counting "min" and "max" without outlier detection will not constrain the names to exactly two attributes, but instead to a range between 2 and 100 attributes. This could be mitigated by pruning the data on both sides but this gets incresingly difficult with higher variance data or multimodal distrubtions.



Our implementation is agnostic to the type of the data, meaning that a categorization into cities with a certain number of inhabitants will not be discovered. That would be useful to have, but it requires a deeper analysis of the involved literals.

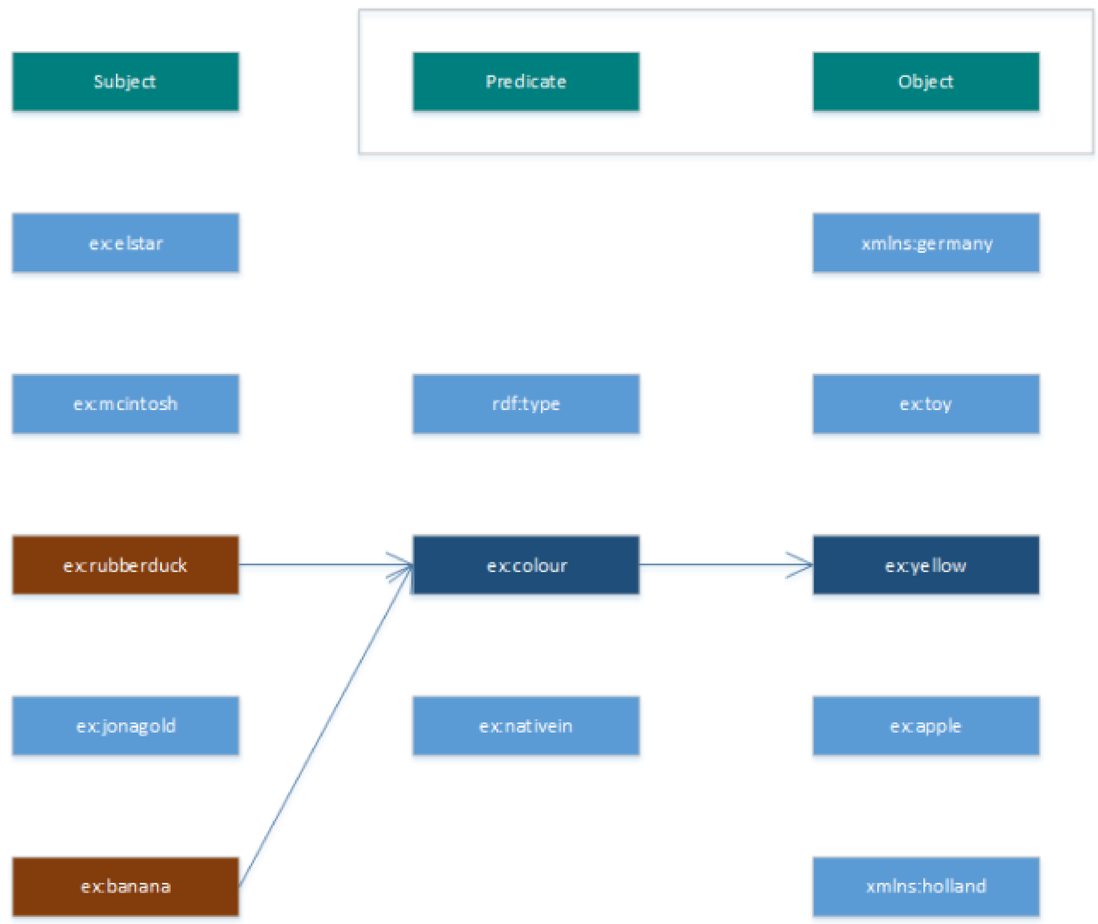
Acknowledgement

This work is conducted as part of the Knowledge Graphs Praktikum offered by RWTH Aachen University Informatik 5 department in collaboration with OSTHUS. We thank OSTHUS for defining research goals and requirements from the industry perspective, and providing student travel grants.

OSTHUS

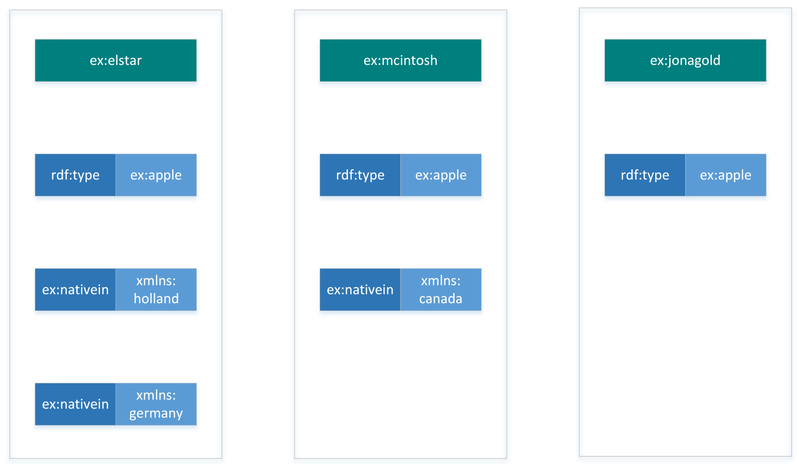
Category discovery

In order to extract meaningful and implicit information about similar entities in a given Knowledge Graph, the available data has to be clustered into categories in a fitting way. Simply using OWL classes would be an intuitive starting point, but such a categorization is unlikely to be optimal because classes are managed by the database provider making information to be extracted very dependant on the resources invested. Furthermore, it is unlikely that a given Knowledge Graph contains classes for every imaginable category meaning various implicit and unobvious categories remain undiscovered. Inspired by natural language, we decided to use an approach based on predicate-object pairs, so called "descriptions". Compare the sentences "Apple is a type of Fruit" and "Banana is a type of Fruit." Obviously "Apple" and "Banana" are similar in some regards even if we have no information what an "Apple" might be. In this particular example, "Apple" and "Banana" would probably fall into the same OWL class "Fruit" but with this approach we are able to find other, less obviously similar entities such as things that are of color yellow.



Feature mining

In our implementation, the actual category discovery occurs by considering each predicate-object pair that is prevalent in the database, make it into a category and prune this set heuristically. Categories that contain fewer than 50 instances are ignored. This is an arbitrary threshold. Subsequently, for each discovered category all contained instances are compared by counting which other predicates are set. For example, every Person that has been a president of the United States is queried for a wife, a dog, a date of death and so forth. This information is stored in n-dimensional feature vectors, where n is the number of predicates in the database. In our current implementation we track how many times properties are set at least (min) and at most (max). Furthermore, a confidence value (conf) is computed stating the relative probability that a property is set at all. The confidence value is a smoothed percentage value that includes a potential bias when there is too little data. For the sake of illustration it is represented by a purely relative amount.



category type-apple	min	max	conf
rdf:type	1	1	1
ex:nativein	0	2	⅔

References

[1] Top Quadrant `https://www.topquadrant.com/2017/09/13/shacl/`

[2] Melo, André, and Heiko Paulheim. "Learning SHACL Constraints for Validation of Relation Assertions in Knowledge Graphs.