Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL

Blerina Spahiu, Andrea Maurino, Matteo Palmonari

University of Milano-Bicocca
{blerina.spahiu|andrea.maurino|matteo.palmonari}@unimib.it

Abstract. As Linked Data available on the Web continue to grow, understanding their structure and assessing their quality remains a challenging task making such the bottleneck for their reuse. ABSTAT is an online semantic profiling tool which helps data consumers in better understanding of the data by extracting data-driven ontology patterns and statistics about the data. The SHACL Shapes Constraint Language helps users capturing quality issues in the data by means of constraints. In this paper we propose a methodology to improve the quality of different versions of the data by means of SHACL constraints learned from the semantic profiles produced by ABSTAT.

1 Introduction

Million of facts describing entities such as people, places, events, films, etc., are stored in knowledge repositories. In the Semantic Web, knowledge is encoded into graphs represented using the RDF data model[1]. Nodes in these graphs represent *entities* or literals, while arcs represent *relations* between entities and between entities and literals, whose semantics is specified by RDF properties. Entities and literals are usually associated a type, i.e., respectively a class or a datatype. The sets of possible types and properties are organized into *schemas* or *ontologies*, which defines the meaning of the terms used in the knowledge base through logical axioms. Often KGs are very large and are continuously evolving, as an example we can mention Linked Data which has evolved with roughly 1,184 data sets as of April 2018[2]. However, understanding and exploring the content of large and complex knowledge bases is very challenging for users [11, 21, 4, 19, 14].

As the number and size of published KGs is increasing, the need for methodologies and tools able to support the quality assessment of such datasets increase as well. Many data producers take care about the quality of the data they publish resulting in many data sets in the LD cloud that are of a high quality. However, there are also many data sets, which are extracted from unstructured

[1] https://www.w3.org/RDF/
[2] http://lod-cloud.net/

or semi-structured information being vulnerable for quality issues [6]. Many metrics, methodologies and approaches have been proposed to capture quality issues in linked data sets [20, 13, 10, 2]. In this context it happens that a new version of a dataset shows a reduced quality level with respect to a previous version. In addition, when an error is discovered in a given version of a dataset, there may be the need to check if such error is also found in previous versions of the dataset that are still in use.

In this paper we propose an approach for improving the quality of different versions of knowledge graphs by means of ABSTAT [15]. In practice, profiles computed with ABSTAT provide datadriven ontology patterns and related statistics. Given an RDF data set and, optionally, an ontology (used in the data set), ABSTAT computes a semantic profile which consists of a summary that provides an abstract but complete description of the data set content and statistics. ABSTAT's summary is a collection of patterns known as Abstract Knowledge Patterns (AKPs) of the form <`subjectType, pred, objectType`>, which represent the occurrence of triples <`sub, pred, obj`>in the data, such that `subjectType` is a minimal type of the subject and `objectType` is a minimal type of the object. With the term type we refer to either an ontology class (e.g., foaf:Person) or a datatype (e.g., xsd:DateTime). By considering only minimal types of resources, computed with the help of the data ontology, we exclude several redundant AKPs from the summary making them compact and complete. Summaries are published and made accessible via web interfaces, in such a way that the information that they contain can be consumed by users and machines (via APIs).

SHACL[3] (Shapes Constraint Language), is a highly expressive language used to write constraints for validating RDF graphs recommended by the W3C. Constraints are written in RDF. In this paper we show not only how ABSTAT patterns encode valuable information to evaluate and improve the quality of knowledge graphs (RDF data sets), but also how this information can be translated into SHACL profiles that can be validated against the data. More precisely, we introduce the first methodology to build SHACL profiles using data-driven ontology patterns extracted automatically from the data. These SHACL profiles can be derived automatically, refined by the users (also with the help of information provided by the semantic profiling tool) and then validated automatically using a SHACL validator. The overall idea is that ABSTAT profiles and SHACL profiles can be used in combination to improve the quality of RDF data sets along time.

Our contributions can be summarized as follows: (i) We provide a methodology for the quality assessment and validation among different versions of a dataset. In particular we propose an approach to detect quality issues in the data by means of cardinality statistics, (ii) Represent ABSTAT semantic profiles into SHACL language and (iii) Use SHACL profiles as templates for quality detection.

[3] https://www.w3.org/TR/shacl/

The rest of the paper is organized as follow: Related work are discussed in Section 2. In Section 3 we introduce the methodology for improving the general quality of the data sets. In Section 4.1 we describe how pattern-based profiles can support data quality assessment of the data. We first introduce ABSTAT and the generated profile consisting of data-driven ontology patterns. We then describe how to transform ABSTAT profiles into SHACL language in order to use them as templates for validation. An example of how such approach can be applied into DBpedia is given in Section 5 while conclusions end the paper in Section 6.

2 Related Work

The identification of quality issues has been recently studied in RDF data sets. The most straightforward way to represent constraints is the use of OWL 2[4] by means of three axioms: owl:minCardinality, owl:maxCardinality, and owl:exactCardinality on ObjectProperty as well as on DatatypeProperty. However such kind of declaration is optional and consistency with the data is not ensure. Especially it is very difficult to evaluate consistency when relations are not explicitly stated such as in the case of lightweight ontologies. The expressiveness of OWL 2 is limited also to express integrity constraints [17].

The quality of DBpedia as one of the most known and used data set of the LOD cloud has been studied in different works such as [6, 20, 8, 18]. In [18] the authors study the property domain/range constraints and enrich the ontology with axioms from the field of Inductive Logical Programming (ILP) and use them to detect inconsistencies in DBpedia. The creation of suitable correction suggestions help users in identifying and correcting existing errors.

Authors in [3] present a study of the usability of 115 constraints on vocabularies commonly used in the social, behavioral, and economic sciences in order to asses the quality of the data. These constraints were validated in 15 694 data sets. Authors gain a better understanding about the role of certain constraint for assessing the quality of RDF data. The main findings refer to language that is used to formulate constraints, 1/2 of all constraints are informational, 1/3 are error, and 1/5 are warning constraints, etc.

The problem of mining cardinality bounds for properties in order to discover structural characteristics of knowledge bases and assess their completeness is introduced by [12]. The algorithm for mining cardinality patterns has two implementations; (1) based on SPARQL and (2) based on Apache Spark and is evaluated against five different real-world or synthetic datasets. The findings show that cardinality bounds can be mined efficiently and are useful to understand the structure of data. Such approach allows to analyze the completeness and the consistency of data. There are also different tools and frameworks for generic quality assessment such as [2, 5, 1]. [2] is a Linked Data quality assessment framework that helps non-programming experts in creating their quality

[4] https://www.w3.org/TR/owl2-syntax/

metrics either procedurally, through Java classes, or declaratively, through a quality metric. In [5] the definition of metrics for quality assessment task should be specified in an XML file. [1] harvests data dumps from the Web in various RDF format, cleans up quality issues relating to the serialization, and republishes the dumps in standards-conform formats.

With respect to the related work described in this section, this paper differs in several ways: i) it provides an iterative way for the improvement of quality over upcoming versions of the data; ii) it introduces the concept of SHACL profiles which serve as templates for validation of the data; and iii) proposes the use of minimal ontology-patterns to detect quality issues in the data.

3 The Proposed Methodology

In this section we introduce the methodology of the approach that can learn validation rules upon semantic profiles. For representing such rules we propose to use SHACL (Shapes Constraint Language), a versatile constraints language for validating RDF. Such rules are used as templates for detecting quality issues for the upcoming versions of data. The process of validating the quality of different versions of the data set through the use of SHACL profiles is given in Figure 1. Validation of the data set could be done in the following phases:

Phase I: As a first step, for a data set (D_1) semantic profiles (A_{p1}) are generated with the help of ABSTAT (Section 4.1). Such semantic profiles allow users to detect quality issues in the data [15].

Phase 2: In order to assess quality issues in the data set (D_1), a custom-built converter was implemented which transforms semantic profiles into SHACL mappings (S_{p1}) (Section 4.3). To the best of our knowledge in the state-of-the-art there are no converters to support such transformation.

Phase 3: SHACL profiles specify severities (sh:severity sh:Warning) to identify non-critical constraint violation for those properties for which the maximum cardinality is higher than twice the average (Section 4.4). We chose such threshold in our experiments because we wanted to verify as much as possible cases which violate the quality of the data. Such threshold is a tunable parameter.

Phase 4: The domain expert takes such report and verifies if the cases identified as Warning are real violations or not. Gradually checking such cases the user updates the SHACL profiles (S_{p1}'), if the violations are not real, or otherwise updates the data set. As a first step in validating our approach we check manually if violations are real or not. We are investigating methods how to capture such real violations automatically.

Phase 5: When a different version of the dataset is available (D_2) the domain expert can directly validate the new data against the constraints S_{p1}' and use semantic profiles for exploring the data.

Such process iterates any time a version of the data comes. In such a way the quality of the upcoming versions of the data improves over time as well as the quality of the data set itself is improved as in Figure 1.

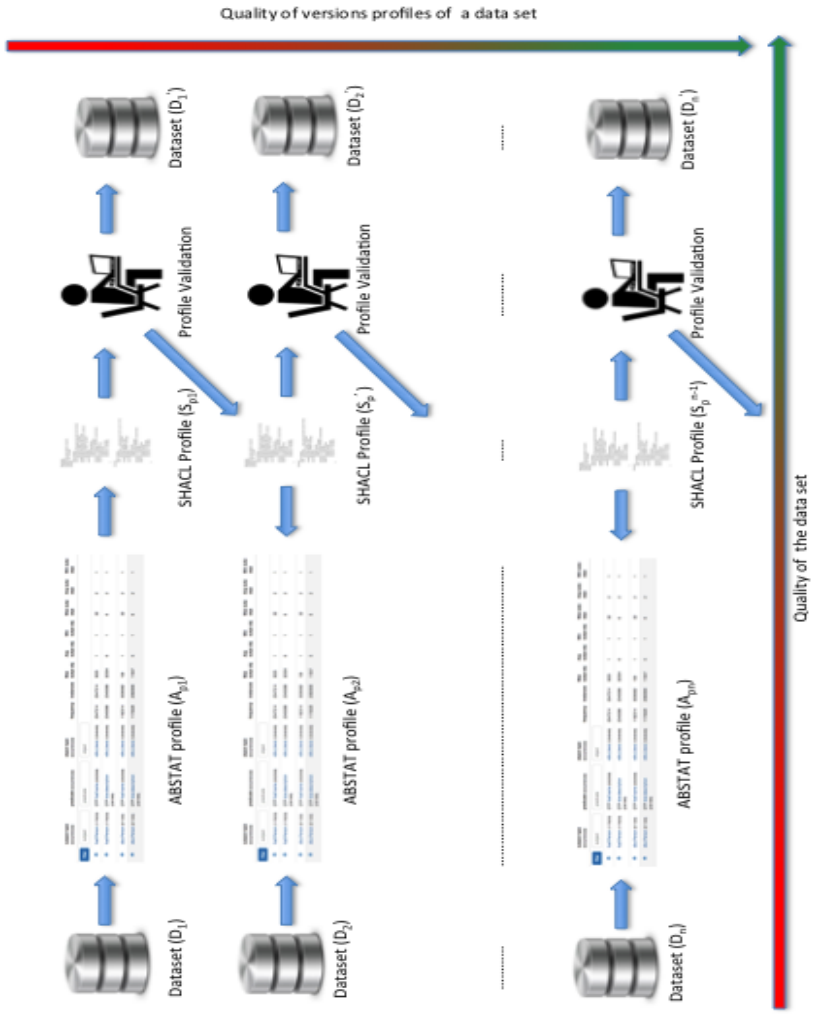

Fig. 1. Iterative quality improvement using SHACL profiles.

4 Data Quality Insights with Pattern-based Profiles

In this section we show how to apply the methodology by describing each phase and presenting a real example.

4.1 Pattern-based Data Summarization with ABSTAT

In the following we describe the main features of ABSTAT bringing the main definitions from our previous paper [15]. We borrow the definition of data set from the definition of Knowledge Base in Description Logics (DLs). In a Knowledge Base there are two components: a *terminology* defining the vocabulary of an

application domain, and a set of *assertions* describing RDF resources in terms of this vocabulary. A data set is defined as a couple $\Delta = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a set of terminological axioms, and $\mathcal{A}$ is a set of assertions. The domain vocabulary of a data set contains a set N^C of *types*, where with type we refer to either a named class or a datatype, a set N^P of named properties, a set of named individuals (resource identifiers) N^I and a set of literals L. In this paper we use symbols like C, C', ..., and D, D', ..., to denote types, symbols P, Q to denote properties, and symbols a, b to denote named individuals or literals. Types and properties are defined in the terminology and occur in assertions. Assertions in $\mathcal{A}$ are of two kinds: *typing assertions* of form $C(a)$, and *relational assertions* of form $P(a, b)$, where a is a named individual and b is either a named individual or a literal. We denote the sets of typing and relational assertions by $\mathcal{A}^C$ and $\mathcal{A}^P$ respectively. Assertions can be extracted directly from RDF data (even in absence of an input terminology). *Typing assertions* occur in a data set as RDF triples $< x, \texttt{rdf:type}, C >$ where x and C are URIs, or can be derived from triples $< x, P, y \hat{} \hat{} C >$ where y is a literal (in this case y is a typed literal), with C being its datatype. Without loss of generality, we say that x is an instance of a type C, denoted by $C(x)$, either x is a named individual or x is a typed literal. Every resource identifier that has no type is considered to be of type `owl:Thing` and every literal that has no type is considered to be of type `rdfs:Literal`. A literal occurring in a triple can have at most one type and at most one type assertion can be extracted for each triple. Conversely, an instance can be the subject of several typing assertions. A *relational assertion* $P(x, y)$ is any triple $< x, P, y >$ such that $P \neq Q*$, where $Q*$ is either `rdf:type`, or one of the properties used to model a terminology (e.g. `rdfs:subClassOf`).

Patterns are abstract representations of Knowledge Patterns, i.e., constraints over a piece of domain knowledge defined by axioms of a logical language, in the vein of Ontology Design Patterns [16]. A pattern is a triple (C, P, D) such that C and D are types and P is a property. In a pattern (C, P, D), we refer to C as the subject type and D to the object type. Intuitively, a pattern states that there are instances of type C that are linked to instances of a type D by a property P. Instead of representing every pattern occurring in the data set, ABSTAT summaries include only a base of minimal type patterns, i.e., a subset of the patterns such that every other pattern can be derived using a subtype graph.

A piece of a subtype graph is shown in Figure 2. Two branches of the concept (classes) hierarchy are shown with respect to `Thing` class. The first branch has the subtype relation between `Agent, Person, Artist, MusicalArtist, Instrumentalist` and `Guitarist` while the second one has the subtype relation between `Place, PopulatedPlace` and `Settlement`. Consider in the data set the following triples: *<dbo:Kurt_Cobain dbo:birthPlace dbo:Washington>*, *<dbo:Kurt_Cobain rdf:type dbo:Guitarist>*, *<dbo:Kurt_Cobain rdf:type dbo:Person>*, *<dbo:Washington rdf:type dbo:Settlement>* and *<dbo:Washington rdf:type dbo:PopulatedPlace>*. From the triples above, in the data *Kurt Cobain* has two types, *Person* and *Guitarist*, while the resource of *Washington* has the types *PopulatedPlace* and *Settlement*. The approach of ABSTAT, which consid-

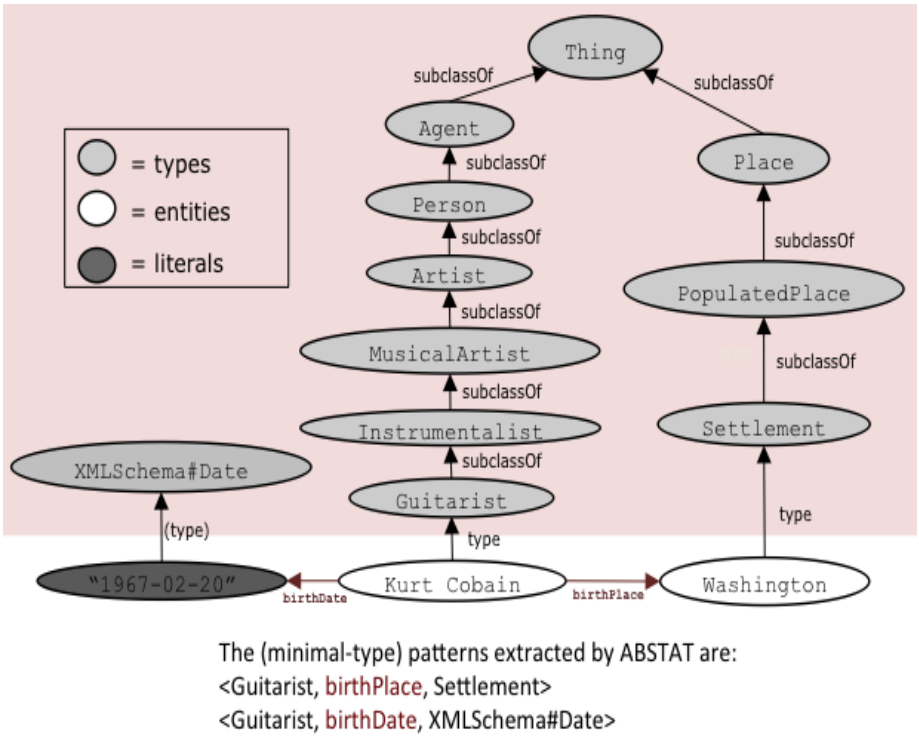

Fig. 2. A small graph representing the subtype relations and the different patterns extracted by ABSTAT. The transitive closure of type assertions is omitted.

ers only the minimal type, extracts the pattern <*dbo:Guitarist, dbo:birthPlace, dbo:Settlement*> as `Guitarist` is the minimal concept between `Person` and `Guitarist` for the resource of `Kurt Cobain`. While the concept `Settlement` is the minimal type because is a subtype of `PopulatedPlace`. For the triple with `birthDate` as predicate ABSTAT could extract the pattern <*dbo:Guitarist, dbo:birthDate, xmls:Date*>. Among patterns ABSTAT is able to extract also other statistical information such as:

- **frequency** which shows how many times this pattern occurs in the data set as minimal type pattern.
- **instances** which shows how many times this pattern occurs based on the ontology, regardless of pattern minimality. In other words, this number tells how many relation instances (i.e., relational assertions) are represented by the pattern, by counting also the relation instances that have a more specific pattern as minimal type pattern. For example, if we have two patterns (C, P, D) and (C, P, D') with D being a subtype of D', (C, P, D) is more specific than (C, P, D') (can be inferred using the subtype graph); thus, instances of (C, P, D') include occurrences of (C, P, D) and (C, P, D') itself.

– **Max (Min, Avg) subjs-obj** cardinality is the maximum (minimum, average) number of distinct subjects associated with a same object through the predicate `p`, with subjects and objects belonging to respectively the subject type and the object type.
– **Max (Min, Avg) subj-objs** is the maximum (minimum, average) number of distinct entities of type is subject position linked to a single entity of the type in the object position through the predicate `p`.

Figure 3 shows the semantic profile generated by ABSTAT for the example above. The statistical information can be understood as below. Frequency shows that the pattern <*dbo:Guitarist, dbo:birthPlace, dbo:Settlement*> occurs 36 times in the data set. The number of instances shows that there are 50 relational assertions that are represented by this pattern, regardless of minimality (e.g., here we sum relational assertions that have (`Guitarist`, `birthPlace`, `City`) as minimal type pattern). Max (Min, Avg) subjs-obj cardinality is the maximum (minimum, average) number of distinct entities of type `Guitarist` linked to a same entity of type `Settlement` through the predicate `birthPlace`. There are at most 6 (minimum 1 and in average 1) distinct entities of type `Guitarist` linked to a single entity of type `Settlement`. Max (Min, Avg) subj-objs is the maximum (minimum, average) number of distinct entities of type `Settlement` linked to a single entity of type `Guitarist` through the predicate `birthPlace`. Notice that occurrences is given also for types and predicates. The type `Guitarist` occurs 151 times, the predicate `birthPlace` occurs 1 168 459 and the type `Settlement` occurs 238 436 times.

subject type (occurrences)	predicate (occurrences)	object type (occurrences)	frequency	instances	Max subjs-obj	Avg subjs-obj	Min subjs-obj	Max subj-objs	Avg subj-objs	Min subj-objs
dbo:Guitarist (151)	OP dbo:birthPlace (1168459)	dbo:Settlement (238436)	36	50	6	1	1	2	1	1
dbo:Guitarist (151)	DTP dbo:birthDate (1101418)	xmls:date (1884250)	88	88	1	1	1	2	1	1

Fig. 3. Patterns example from ABSTAT.

4.2 Shapes Constraint Language (SHACL)

Shapes Constraint Language (SHACL) since July 2017 is a W3C recommendation language for defining constraints on RDF graphs. The scope of SHACL is primarily validation but it can be extended to data integration, code generation and interface building. Validating RDF graphs is done against a set of shapes. A SHACL processor has two inputs: a data graph that contains the RDF data to validate and a shapes graph that contains the shapes [7]. RDF graphs containing conditions are called *shape graphs* while the RDF graphs that are validated

against them are called *data graphs*. A result of the validation process is a validation report. There are two types of shape; node shape that declare constraints directly on a node and property shape that declare constraints on the values associated with a node through a path. Node shapes declare constraints directly on a node e.g., node kind (IRI, literal or blank), IRI regex, etc. Property shapes declare constraints on the values associated with a node through a path, e.g., constraints about a certain ongoing or incoming property of a focus node; cardinality, datatype, numeric min/max, etc. SHACL shapes may define several target declarations. Target declarations specify the set of nodes that will be validated against a shape, e.g., directly pointing to a node, or all nodes that are subjects to a certain predicate, etc. The validation report produced by SHACL contains three different severity levels; Violation, Warning and Info. Severity does not matter to the validation result and can be specified by users while writing constraints. SHACL is divided into SHACL Core, which describes a core RDF vocabulary to define common shapes and constraints while SHACL-SPARQL describes an extension mechanism in terms of SPARQL. In SHACL-SPARQL, there are two types of validators: SELECT-based and ASK-based queries. While SELECT-based validators return no results to indicate conformance with the constraint and a non-empty set when violated. ASK-based validators return true to indicate conformance.

4.3 ABSTAT Profiles into SHACL

Figure 4 shows the syntax of the example pattern in Figure 3 into the SHACL language (Phase 2 of the methodology). Not all the information in the semantic profile produced by ABSTAT can be represented in SHACL. Only a subset of the information such as the minimum and maximum cardinality for both subj-objs and subjs-obj can be represented. We can not represent the information about the frequency of types, predicates, patterns, the occurrence and the average cardinality for both subj-objs and subjs-obj. For such statistics we will extend SHACL in the future as described in Section 6.

The semantic profile produced by ABSTAT can be transformed into SHACL by representing each pattern (C, P, D) (in parenthesis we map such process with the example in the Figure 4) as follows:

- For each type C (Guitarist) we extract all patterns with C (Guitarist) as the subject type. The subject type C (Guitarist) is mapped into a node shape (sh:NodeShape) targeting the class.
- The property P (birthPlace or birthDate) is mapped into a property shape (sh:property) with a value for sh:path equal to the property.
  - If for all patterns (C, P, Z), (Guitarist, birthPlace) Z (Settlement or City) is a class, then a "global" sh:path specification is added that specifies the sh:nodeKind as an sh:IRI. If for all patterns (C, P, Z), Z (xmls:date) is a datatype, then a "global" sh:path specification is added that specifies the sh:datatype as a rdfs:Literal. Observe that this step is executed only for the first pattern with C (Guitarist) as subject type and P (birthPlace, birthDate) as property.

```
Shape:Guitarist
a sh:NodeShape;
sh:targetClass class:Guitarist; #Applies to all guitarist.
sh:property [
        sh:name “birthPlace”;
        sh:path dbo:birthPlace;
        sh:nodeKind sh:IRI; #The birthplace is given in IRI link.
        sh:path [dbo:birthPlace
                sh:nodeKind dbo:Settlement;
        sh:minCardinality “0”^^xsd:integer; #birthPlace is not a required property.
        sh:maxCardinality “6”^^xsd:integer;
        sh:path [sh:inversePath sh:birthPlace;
                sh:nodeKind sh:IRI;
                sh:minCardinality “0"^^xsd:integer;
                sh:maxCardinality “2"^^xsd:integer; ]
        ];
        sh:path [dbo:birthPlace
                sh:nodeKind dbo:City;
        sh:minCardinality “0”^^xsd:integer; #birthPlace is not a required property.
        sh:maxCardinality “3”^^xsd:integer;
        sh:path [sh:inversePath sh:birthPlace;
                sh:nodeKind sh:IRI;
                sh:minCardinality “0"^^xsd:integer;
                sh:maxCardinality “1"^^xsd:integer; ]
        ];
sh:property [
        sh:name “birthDate”;
        sh:path    dbo:birthDate;        #This property shape applies to guitarist’s birthday.
        sh:datatype xmls:date; #birthDate is a date;
        sh:path [dbo:birthDate
                sh:datatype xmls:date;
        sh:minCardinality “0”^^xsd:integer; #birthDate is not  a required property.
        sh:maxCardinality “1”^^xsd:integer;
        sh:path [ sh:inversePath sh:birthDate;
                sh:datatype xmls:date;
                sh:minCardinality “0^^xsd:integer;
                sh:maxCardinality “2"^^xsd:integer; ];
        ];
```

Fig. 4. ABSTAT patterns into SHACL language.

- a ”local” sh:path specification is added to specify characteristics for each object type Z (Settlement, City) for the property P (birthPlace). For each path P (birthPlace), Z (Settlement, City), cardinality constraint components are set, namely sh:minCount, sh:maxCount.
- sh:inversePath property is used to describe the inverse cardinality for the properties.

4.4 Capturing Quality Issues by Applying Heuristics

In this section we describe how the domain expert can apply some heuristics in the semantic profiles and use SHACL to validate the data (Phase 3).

For all patterns for which the number of maximum cardinality is higher than twice the average we generate a severity report sh:Warning. We put such threshold in order to capture as much as possible cases where the quality might

```
Shape:Company
a sh:NodeShape;
sh:targetClass class:Company; #Applies to all companies.
sh:property [
        sh:name "keyPerson";
        sh:path dbo:keyPerson; #This property shape applies to companies key person.
        sh:nodeKind sh:IRI; #The keyperson is given in IRI link
        sh:path [dbo:keyPerson
                  sh:nodeKind owl:Thing;
        sh:minCardinality "0"^^xsd:integer; #keyPerson is a required property.
        sh:maxCardinality "5263"^^xsd:integer;
        sh:severity sh:Warning ;
        sh:sparql [
                sh:message "Triples that might violate quality";
                sh:prefixes dbo: ;
                sh:select """
                SELECT  ?s  <http://dbpedia.org/ontology/keyPerson> as  ?p   ?o
                where {
                ?s a <http://dbpedia.org/ontology/Company> .
                ?o a <http://dbpedia.org/ontology/keyPerson> .
                ?s <http://dbpedia.org/ontology/keyPerson> ?o .
                }
                """;
        ].
```

Fig. 5. SHACL validation of possible errors generated by ABSTAT.

be affected. For those patterns we use SHACL-SPARQL to generate the entities that violate such constraints as in the Figure 5.

5 DBpedia Case Study

In the following we consider the triple patterns extracted by ABSTAT in Figure 6, which contain descriptive information about companies in DBpedia 2015-10. Note that the triple patters in the figure are just a sample of the patterns containing `dbo:Company` in the subject type. In this example we also have fixed the predicate which is `dbo:keyPerson`. As described above apart from triple patterns, ABSTAT is able to produce also statistics that describe some characteristics about the data. The type Company is used in the data 51 898 times while the predicate `dbo:keyPerson` is used 31 078 times. The first triple pattern *<dbo:Company, dbo:keyPerson, owl:Thing>* occurs 18 710 times in the data. While the number of instances having such pattern including those for which the types `Company`, `Thing` and the predicate `keyPerson` can be inferred is 29 884. Moreover ABSTAT is able to describe some characteristics of the data (in Figure 6) as below:

R1. There are about 5 268 *distinct entities of type* `Company` *linked to a single entity of type* `Thing` *through the predicate* `keyPerson` *(Max subjs-obj).*

R2. For each entity of type `Company` *there exist at least one value for the predicate* `keyPerson` *(Min subjs-obj).*

R3. The number of distinct entities of type `Thing` *linked to a single entity of type* `Company` *through the predicate* `keyPerson` *is* 23 *(Max sub-objs).*

R4. For each entity of type `Thing` *there exist at least one entity of type* `Company` *for the predicate* `keyPerson` *(Min sub-objs).*

R5. The predicate `keyPerson` *is an object property thus all triples which have this as predicate take an entity in the object (the OP symbol in the predicate field of the pattern).*

R6. The domain and range of the property `keyPerson` *is the union of the types in the subject position and the union of the types in the object position of triple patterns that have* `keyPerson` *as predicate.*

subject type (occurrences)	predicate (occurrences)	object type (occurrences)	frequency	instances	Max subjs-obj	Avg subjs-obj	Min subjs-obj	Max subj-objs	Avg subj-objs	Min subj-objs
dbo:Compan	dbo:keyPerso	object								
dbo:Company (51898)	OP dbo:keyPerson (31078)	owl:Thing	18710	29884	5263	3	1	23	2	1
dbo:Company (51898)	OP dbo:keyPerson (31078)	dbo:Person (611330)	6808	9049	14	1	1	12	1	1
dbo:Company (51898)	OP dbo:keyPerson (31078)	dbo:Office-Holder (60424)	361	404	4	1	1	4	1	1
dbo:Company (51898)	OP dbo:keyPerson (31078)	dbo:Company (51898)	236	254	2	1	1	2	1	1

Fig. 6. Detecting errors in the data by exploring triple patterns.

By looking at the descriptive information gained from ABSTAT in the example above, we can see that they reveal quality issues in the data. R1, R2, R3 and R4 are known as cardinality constraints (used to specify a minimum and maximum bound for relationships with properties that an entity can have [9]) while, constraints of the same form as R5 are known as integrity constraints (used to ensure accuracy and consistency of data). Constraints such as R6 are domain and range constraints, respectively; they restrict the types that entities (in the domain) and values (in the range) of relations for a given property can have. The statistics about the average cardinality are indicators that can help users identify violations of such constraints. In the example considered it is very strange that while the average number of distinct entities of type Company linked with one entity of a given type (which is categorized as Thing) is 3, while the maximum is 5 268. It means that from 18 710 times that this pattern occurs in the data, more than 25% (5 268) of the patterns are created by these distinct entities of type `Company` that are linked with the same entity of type `Thing`. The high imbalance between the number of average and maximal cardinality subjs-obj incite us on making further exploration and checks. Querying the endpoint of DBpedia we could identify the entity of

type `Thing` which is linked with 5 268 different entities of type `Company`. The entity is <http://dbpedia.org/resource/Chief_executive_officer>which in DBpedia does not have a type. This means that a Chief Executive Officer (represented as an entity) might be the CEO of 5 268 distinct companies. Examples of triples of entities which type is Company linked with the entity CEO are:

<dbr:Kodak dbo:keyPerson dbr:Chief_executive_officer>

<dbr:Telefnica dbo:keyPerson dbr:Chief_executive_officer>

<dbr:Allianz dbo:keyPerson dbr:Chief_executive_officer>

Similarly, we could identify such quality issue for the triple pattern *<dbo:Company, foaf:homePage, owl:Thing >*. The max cardinality subjs-obj is 27 while the minimum and the average cardinality is 1. Looking at the triples from which ABSTAT derived the above pattern, there are different entities of the type `Company` linked with the predicate homepage of the FOAF vocabulary to the `http://www.centurylink.com` webpage which are linked with instances of type `Thing`. Some example of such kind of triples are the following:

<dbr:Embarq_Florida foaf:homepage http://www.centurylink.com/>

<dbr:Central_Telephone foaf:homepage http://www.centurylink.com>

<dbr:United_Telephone_Company_of_Kansas foaf:homepage http://www.centurylink.com>

In the version of DBpedia considered in this paper we did not find any violation for the R5. Such violations are easy to be identified in the version of DBpedia 3.9 with Infoboxes e.g., birthDate is used as an object property (with the `dbp` namespace) also as a datatype property (with the `dbo` namespace).

The violation of the R6 is caused by triple patterns of the form *<dbo:Company, dbo:keyPerson, dbo:Company>*. In the ontology of DBpedia, the range of `dbo:keyPerson` is `Person` but in the data such property is used also with instances of the type `Company` (254 patterns), `Organization` (308 patterns), `University` (15 patterns), `Bank` (13 patterns) or `Software` (7 patterns) etc. In the other example the semantics of the predicate `foaf:homepage` implies the domain to be a `Thing` while the range should be a Document[5]. Such restriction is violated by the use of such predicate in DBpedia 2015-10.

In this work, we limit ourselves to mainly study the violation of cardinality constraints and integrity constrains described above because of the impact that they have in the quality analysis of the data. Most of the Linked Data quality assessment, require users to have knowledge about the structure of the data whereas we aim to provide users a method to capture quality issues by means of identifying violated constraints against semantic profiles produced by ABSTAT.

In our experiments we use DBpedia as one of the most used data set covering different domains and being one of the most curated data set in the LOD cloud. However other data sets being a single-topic data set such as media, life science, etc. data sets are planed to be considered in our future experiments.

[5] `http://xmlns.com/foaf/spec/#term_homepage`

6 Conclusions and Future Work

In this paper we propose a methodology for assessing the quality of data sets and their versions by means of an ontology pattern profiling tool. For each data set, ABSTAT generates a semantic profile that describes some characteristics about the data. The semantic profiles are then transformed into SHACL for the validation of constraints. Cases where the cardinality between max and average have a high imbalance are reported as warnings. A user verifies further such warnings and consequently updates the data set and the SHACL profile. Although this research is in the beginning, samples extracted from DBpedia have shown that it is an effective way to capture and improve quality issues over time.

As a future direction we plan to implement the full pipeline for the generation of SHACL profiles and SHACL validators and make the code available. We plan to extend SHACL by including also other statistical information produced by ABSTAT in order to apply sophisticated heuristics to capture quality issues such as including frequency and occurrence of a pattern. Moreover, we plan to run the experiment in large scale, thus including more data sets that belong to different topical domains and make an analysis of the most frequent quality issues.

Another interesting future direction is the investigation of inconsistencies or quality issues over time for those data sets that have versioned ontologies. The semantic profiles produced by different versions of the data and the ontology can be compared by exploring summaries while SHACL profiles can be used as templates for the verification of errors found in the other version of the data.

Acknowledgements

This research has been supported in part by EU H2020 projects EW-Shopp - Grant n. 732590, and EuBusinessGraph - Grant n. 732003. This work will be published as part of the book "Emerging Topics in Semantic Technologies. ISWC 2018 Satellite Events. E. Demidova, A.J. Zaveri, E. Simperl (Eds.), ISBN: 978-3-89838-736-1, 2018, AKA Verlag Berlin".

References

[1] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. Lod laundromat: a uniform way of publishing other peoples dirty data. In *International Semantic Web Conference*, pages 213–228. Springer, 2014.

[2] J. Debattista, S. Auer, and C. Lange. Luzzu–a framework for linked data quality assessment. In *Semantic Computing (ICSC), 2016 IEEE Tenth International Conference on*, pages 124–131. IEEE, 2016.

[3] T. Hartmann, B. Zapilko, J. Wackerow, and K. Eckert. Directing the development of constraint languages by checking constraints on rdf data. *International Journal of Semantic Computing*, 10(02):193–217, 2016.

[4] S. Khatchadourian and M. P. Consens. Explod: summary-based exploration of interlinking and rdf usage in the linked open data cloud. In *Extended Semantic Web Conference*, pages 272–287. Springer, 2010.

[5] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd international conference on World Wide Web*, pages 747–758. ACM, 2014.

[6] D. Kontokostas, A. Zaveri, S. Auer, and J. Lehmann. Triplecheckmate: A tool for crowdsourcing the quality assessment of linked data. In *International Conference on Knowledge Engineering and the Semantic Web*, pages 265–272. Springer, 2013.

[7] J. E. Labra Gayo, E. Prud'hommeaux, I. Boneva, and D. Kontokostas. *Validating RDF Data*, volume 7 of *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool Publishers LLC, sep 2017.

[8] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[9] S. W. Liddle, D. W. Embley, and S. N. Woodfield. Cardinality constraints in semantic data models. *Data & Knowledge Engineering*, 11(3):235–270, 1993.

[10] P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: linked data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123. ACM, 2012.

[11] 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 *International Semantic Web Conference (Posters & Demos)*, 2015.

[12] E. Muñoz and M. Nickles. Mining cardinalities from knowledge bases. In *International Conference on Database and Expert Systems Applications*, pages 447–462. Springer, 2017.

[13] H. Paulheim and C. Bizer. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):63–86, 2014.

[14] S. Peroni, E. Motta, and M. dAquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *Asian Semantic Web Conference*, pages 242–256. Springer, 2008.

[15] B. Spahiu, R. Porrini, M. Palmonari, A. Rula, and A. Maurino. Abstat: Ontology-driven linked data summaries with pattern minimalization. In *International Semantic Web Conference*, pages 381–395. Springer, 2016.

[16] S. Staab and R. Studer. *Handbook on ontologies*. Springer Science & Business Media, 2010.

[17] J. Tao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity constraints in OWL. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[18] G. Töpper, M. Knuth, and H. Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 33–40. ACM, 2012.

[19] G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. Rdf digest: Efficient summarization of rdf/s kbs. In *European Semantic Web Conference*, pages 119–134. Springer, 2015.

[20] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

[21] X. Zhang, G. Cheng, and Y. Qu. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th international conference on World Wide Web*, pages 707–716. ACM, 2007.