RDF Shape Induction using Knowledge Base Profiling

Nandana Mihindukulasooriya
Universidad Politécnica de Madrid, Spain
nmihindu@fi.upm.es

Mohammad Rifat Ahmmad Rashid
Politecnico di Torino, Italy
mohammad.rashid@polito.it

Giuseppe Rizzo
Instituto Superiore Mario Boella, Italy
giuseppe.rizzo@ismb.it

Raúl García-Castro
Universidad Politécnica de Madrid, Spain
rgarcia@fi.upm.es

Oscar Corcho
Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es

Marco Torchiano
Politecnico di Torino, Italy
marco.torchiano@polito.it

ABSTRACT

Knowledge Graphs (KGs) are becoming the core of most artificial intelligent and cognitive applications. Popular KGs such as DBpedia and Wikidata have chosen the RDF data model to represent their data. Despite the advantages, there are challenges in using RDF data, for example, data validation. Ontologies for specifying domain conceptualizations in RDF data are designed for entailments rather than validation. Most ontologies lack the granular information needed for validating constraints. Recent work on RDF Shapes and standardization of languages such as SHACL and ShEX provide better mechanisms for representing integrity constraints for RDF data. However, manually creating constraints for large KGs is still a tedious task. In this paper, we present a data driven approach for inducing integrity constraints for RDF data using data profiling. Those constraints can be combined into RDF Shapes and can be used to validate RDF graphs. Our method is based on machine learning techniques to automatically generate RDF shapes using profiled RDF data as features. In the experiments, the proposed approach achieved 97% precision in deriving RDF Shapes with cardinality constraints for a subset of DBpedia data.

CCS CONCEPTS

• **Information systems → Data cleaning**; • **Computing methodologies → Knowledge representation and reasoning**;

KEYWORDS

RDF Shape, Knowledge Base, Data Quality, Machine Learning

1 INTRODUCTION

Knowledge Graphs (KGs) are becoming the core of most artificial intelligent applications driven by domain knowledge. Popular KGs such as DBpedia, YAGO2, and Wikidata have chosen the RDF data model for knowledge representation. RDF is a graph-based data model which is the *de facto* model in Semantic Web and Linked Data applications. RDF graphs can capture and represent domain information in a semantically rich manner using ontologies. As with any other data model, in order to specify the conditions that must be satisfied by an RDF graph, constraints need to be imposed. In practical settings, constraints are commonly used for three main tasks: (a) specifying the properties that data should hold; (b) handle contradictions within data or with respect to the domain under consideration; or (c) as a help for query optimization.

Ontology languages are designed for inferring new knowledge using known axioms rather than for validating data based on axioms. A reasoner and a validator have different functions, *i.e.*, a reasoner is used for inferring new knowledge while a validator is used for finding violations against a set of constraints. The underpinning principles used in OWL, such as the use of Open World Assumption (OWA) and Non-Unique Name Assumption, can lead to unexpected results in a validator. Thus, it is challenging to perform certain validation tasks using such languages.

Validation against a pre-defined schema is one of the key steps in most conventional data storage and publishing paradigms; for instance, validation against DDL constraints in SQL databases or XML Schema or RelaxNG in XML documents [27]. For RDF data, typically OWL or RDF Schema based ontologies are used for doing this kind of validations. However, the use of ontologies for validating RDF data leads to several problems: (a) the mismatch between intended purposes, inference (for which the ontologies are designed for) and validation, and (b) lack of information useful for validation in most of the ontologies.

In general, *data quality* remains a critical aspect to obtain trust by users. Data quality relates to the perception of the "fitness for use" in a given context [32]. Constraints are limitations incorporated on the data that are supposed to be satisfied all the time by instances of the database [2]. They are useful for users to understand data as they represent characteristics that data naturally exhibits [22]. Data profiling is defined as "the activity of creating small but informative summaries of a database" [17]. In this study, we present a data-driven approach by applying machine learning techniques on profiled RDF data to automatically generate integrity constraints. Then, a set of constraints related to a specific class are combined to generate an RDF Shape as a collection of constraints. These RDF shapes can be used to validate RDF graphs.

One of the main use cases for generating RDF Shapes from data is quality assessment. In this work, we use DBpedia [21] (version 2016-04) as our target KB and we experiment on assessing the quality of all *dbo:Person* instances in the DBpedia KB. The main requirement for such quality assessment would be to identify the validation rules that each person instance should conform to. For example, a person should have exactly one value for the "dbo:birthDate" property or the values of the "dbo:height" property should always be a positive number. The instances of the Person class have more than 13,000 associated properties (including dbo, DBpedia ontology properties and dbp, auto-generated properties from Wikipedia infobox

keys). Thus, it is a tedious, time-consuming, and error-prone task to generate such validation rules manually. Some of the validation rules can be encoded into the ontology, but it still requires a lot of manual effort. This work proposes an approach for inducing such validation rules in the form of RDF shapes by profiling the data and using inductive approaches to extract those rules. Other use cases for inducting shapes include describing the data (which is helpful in generating queries or creating dynamic user interfaces).

The main contribution of this paper is a novel approach for generating RDF Shapes based on inducing constraints using machine learning algorithms and using data profiling information as features. The proposed approach is defined in a generic way that applies to any type of constraint and is validated using two common constraint types: (a) cardinality constraints, and (b) range constraints.

The remainder of this paper is structured as follows: in Section 2 we define the problem setting; in Section 3 we provide references and a background on related research activities; in Section 4 we describe our approach and in Section we report the experimental results 5 achieved by our approach in a controlled setup (Section 6). We conclude in Section 7 providing insights on future research endeavours.

2 PROBLEM DEFINITION

A knowledge graph is defined to be consistent if it does not contain conflicting or contradictory data [16]. When a schema is available with the integrity constraints that the data should comply, the data usually goes through a validation process that verifies the compliance against those constraints. These integrity constraints encapsulate the consistency requirements for the data in order to be fit to a given set of use cases. For instance, in relational databases, integrity constraints are expressed in a data definition language (DDL), and the database management systems (DBMS) ensure that any data inserted into the database will not lead to any inconsistency in the entire database. However, validation of RDF data is not done in this manner due to several reasons such as the lack of a language for expressing constraints (RDFS and OWL are designed for entailment and not for validation as discussed in the introduction), or having generic models suitable for wider use and not for specific use cases.

When an ontology is available with TBox axioms that define the conceptualization of the domain, a reasoner can be used to verify whether the dataset is consistent with the domain by verifying the axioms defined in the ontology. However, most of the ontologies do not have rich axioms that could help to detect inconsistencies in data. Most of the large knowledge bases, such as DBpedia, lack a definition of integrity constraints and it is a tedious task to manually create these constraint definitions from scratch. Further, most of the schema information about RDF data is only available in the form of OWL ontologies that are most suited for entailment rather than validation. However, most of the practical use cases that utilize RDF data need validating according to integrity constraints.

In recent years, the W3C community has put an effort to standardize a language for describing structural integrity constraints and validation rules for RDF instance data through the RDF Data Shapes Working Group[1] and later on through the Shape Expression Community Group[2]. The outcomes of such efforts, languages such as W3C Shapes Constraint Language (SHACL) [19] and Shape Expressions Language [26], allow integrity constraints to be defined for RDF data and to validate data. However, for large RDF knowledge bases such as DBpedia, which contains more than 750 classes and 60,000 properties (*dbo:* and *dbp:*), it is still a challenge to generate integrity constraints manually. In this context, the main research problem addressed in the paper is: *How to automatically generate integrity constraints for RDF data?*

In this study, we propose a data-driven approach to automatically generate RDF shapes for a given dataset by applying machine learning techniques to RDF data profiling information extracted from the dataset.

2.1 Objective

Using the information extracted from statistical data profilers, our objective is to generate representative RDF Shapes for the data in a given dataset. We generate shapes at the class-level. An example excerpt of RDF Shape in SHACL for the *dbo:Person* class is illustrated in Listing 1.

Listing 1: A snippet from an example Person shape

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

ex:DBpediaPerson a sh:NodeShape;
 sh:targetClass dbo:Person;
 sh:property [sh:path foaf:name;
  sh:minCount 1;
  sh:nodeKind sh:Literal ];
 sh:property [ sh:path dbo:birthDate;
  sh:datatype xsd:date ;
  sh:minCount 1;
  sh:maxCount 1;
  sh:nodeKind sh:Literal ] ;
 sh:property [ sh:path dbo:birthPlace;
  sh:datatype dbo:Place;
  sh:nodeKind sh:BlankNodeOrIRI;
  sh:minCount 1;
  sh:maxCount 1 ] .
```

To this end, a large number of special purpose systems and techniques have been developed for solving such constraint-based mining and learning problems [10]. Constraints can be used both in a descriptive setting, such as discovering association rules [3], or in a predictive setting, such as rule learning [13]. The key contribution of this paper is that we study how profiled data can be applied to RDF shape induction, which, to the best of our knowledge, has not been addressed for RDF data validation problems.

Our main hypothesis is that machine learning techniques can be used to generate correct RDF Shapes by using data profiling information as features. In Section 4 we define an approach for generating RDF Shapes and validate our hypothesis by evaluating

[1] https://www.w3.org/2014/data-shapes/charter
[2] https://www.w3.org/community/shex/

the generated RDF Shapes to ensure that the constraints that are defined are accurate (using manually labelled test data).

3 RELATED WORK

Ontology Learning (OL) is commonly defined as a field that comprises techniques for automated acquisition of ontological knowledge from data. Thus, the paradigm has shifted such that many approaches do not aim to generate a full fledged, gold-standard ontology from data anymore, but they rather focus on acquiring axioms of certain shapes such as concept definitions, atomic subsumptions, disjointness axioms. There are several works done on induction of Description Logic axioms using methods, such as:

Association rule mining (ARM). Abedjan *et al.* [1] present rule-based approaches for predicate suggestion, data enrichment, ontology improvement, and query relaxation. They identified inconsistencies in the data through predicate suggestion, enrichment with missing facts, and alignment of the corresponding ontology. Also they allow users to handle inconsistencies during query formulation through predicate expansion techniques.

Probabilistic graphical models (PGMs). An approach of probabilistic graphical models (PGMs) allows to generate interpretable models that are constructed and then manipulated by reasoning algorithms [20]. These models can also be learned automatically from data, allowing the approach to be used in cases where the manual building of a model is difficult or even impossible.

Statistical Relational Learning (SRL). It is a branch of machine learning that tries to model a joint distribution over relational data [14]. SRL is a combination of statistical learning which addresses uncertainty in data and relational learning which deals with complex relational structures [18].

Inductive logic programming (ILP). Buhmann *et al.* [8] present an approach of inductive lexical learning of class expressions by combining an existing logical learning approach with statistical relevance measures applied on textual resources.

Pattern extraction. It is an area of work where RDF data is analyzed to extract common patterns, for example, in the form of Frequent Graph Patterns [4] or Statistical Knowledge Patterns [7]. These approaches analyze the underlying RDF data and extract the characteristics related to the ontological axioms based on most frequent patterns. This approach is closely related to the work presented in this paper.

Our final goal is different from these research approaches. Instead of inducting the ontological axioms, our goal is to induct validation rules. In recent years various RDF validation languages have been introduced based on expressing constraints.

- The Web Ontology Language (OWL) [23] is an expressive ontology language based on Description Logics (DL). The semantics of OWL addresses distributed knowledge representation scenarios where complete knowledge about the domain cannot be assumed. To address the above mismatch some approaches use OWL expressions with Closed World Assumption and a weak Unique Name Assumption so that OWL expressions can be used for validation purposes such as Stardog ICV [3] and the approach of Tao *et al.* [31].
- The W3C Shapes Constraint Language (SHACL) [19] is used for validating RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. In particular it helps to identify constraints by using SPARQL. Also, it provides a high level vocabulary to identify predicates and their associated cardinalities, and datatypes.
- The Shape Expressions (ShEx) [26] language describes RDF nodes and graph structures. A node constraint describes an RDF node (IRI, blank node or literal) and a shape describes the triples involving nodes in an RDF graph. These descriptions identify predicates and their associated cardinalities and datatypes.
- SPARQL Inferring Notation (SPIN)[4] constraints associate RDF types or nodes with validation rules. In particular, SPIN allows users to use SPARQL to specify rules and logical constraints.

These shape expression languages, namely, ShEx, SHACL, and SPIN, aim to validate RDF data and to communicate data semantics among users. They cover constraints such as keys and cardinality; however, their expressivity is limited and they require user intervention. In our approach we aim to automate the process of shape generation using machine learning. More specifically, various constraints can be envisioned in RDF based on their success in the relational model, such as cardinality and range constraints. Neuman and Moerkotte have proposed "characteristic sets" for performing cardinality estimations for RDF queries with multiple joins [25]. These works differ from the work presented in this paper on two axes. First, they are focused on determining the cardinalities of each value rather than the cardinality of the entity-value relation. Second, they are focused on query optimization than integrity constraint validation. However, we use the base of these works such as analysis of mean, variance, and other statistical features to derive an approach for cardinality estimation for integrity constraint validation.

4 APPROACH

As discussed above, RDF Shapes could help validating existing KGs; nevertheless, it is a tedious task to do manually. Thus, to address the challenge of automatically generating RDF Shapes, we propose a workflow for RDF Shape induction based on data profiling. We define a generic workflow so that it can be applied to any type of constraint. We provide examples for two types of constraints, namely, cardinality and range constraints. Similarly, it can be extended to other types of constraints such as value range constraints (min and max values), string constraints (minLength, maxLength, pattern, languagesIn, uniqueLanguage), or property pair constraints (lessThan, lessThanOrEquals, disjoint, equal) [19].

The goal of the workflow is to extract the constraints by analyzing data patterns and statistics and to combine the extracted constraints to generate RDF Shapes that can be used for validation.

[3] https://www.stardog.com/docs/
[4] http://spinrdf.org/

The main steps of the proposed workflow are illustrated in Figure 1. The input to the workflow is statistical profiling information coming from Loupe [24], a tool for profiling RDF data.

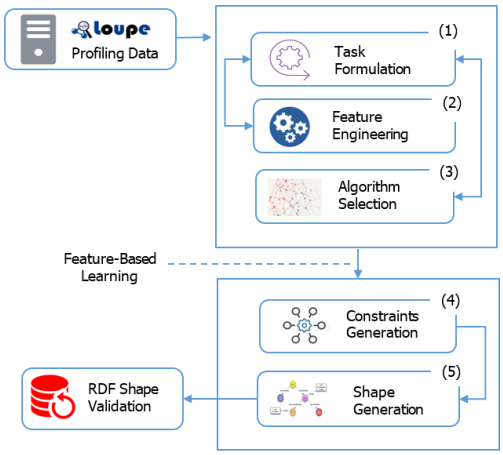

Figure 1: Workflow for RDF constraints generation.

4.1 Task formulation

The first step of the workflow is the formulation of the specific constraint extraction goal as a typical machine learning task.

Cardinally constraints: For the cardinality constraints, our goal is to generate two cardinality constraints when it is relevant, i.e., minimal cardinality and maximum cardinality. As the first step of understanding the task better, we gathered the expected cardinalities (*i.e.,* a gold standard) for a subset of 215 properties from the 3cixty knowledge base [33] and 166 properties of the Spanish National Library (BNE) dataset. When analyzing the dataset, we found only few common patterns repeated in both datasets. Table 1 shows the common cardinality patterns we found in the gold standard.

Table 1: Minimum and maximum cardinality levels.

Key	Description
MIN0	Minimum Cardinality = 0
MIN1	Minimum Cardinality = 1
MIN1+	Minimum Cardinality >1
MAX1	Maximum Cardinality = 1
MAX1+	Maximum Cardinality >1

We could observe the same trend in the vocabularies where the cardinality constraints are explicitly expressed. When we analyzed the 551 vocabularies in the *Linked Open Vocabularies (LOV)* catalogue for the values of cardinality restrictions, 96.91% (848 out of 875) of owl:maxCardinality constraints have value 1 and 93.76% (631 out of 673) of the owl:minCardinality values either 0 or 1.

Thus, in our work rather than trying to estimate the exact numbers for the minimum cardinality and maximum cardinality values, we find which cardinality category each property has with respect to a given class. By doing so, we reduce the problem from a regression problem to a classification problem.

Range constraints: For range constraints, our goals are twofold. First, we want to generate an object *node kind* constraint for each property associated with a given class. We use a subset of *node kinds* already identified in SHACL, i.e., *IRI*, *Literal*, *BlankNode*, and *BlankNodeOrIRI*. Once the node kind is determined, then more specific range constraints have to be decided. If the node kind is *Literal*, the corresponding datatype has to be determined. If the node kind is either *IRI*, *BlankNode*, or *BlankNodeOrIRI*, the class type of the object has to be determined.

We formulate this task also as a classification problem. We classify each property associated with instances of a given class to one of the aforementioned node types. The second task of assigning the corresponding datatype or class as the range of each property is done based on heuristics of datatypes or class type distributions among the set of objects associated with the property.

4.2 Feature engineering

Cardinally constraints: Loupe provides cardinality information for each property associated with instances of a given class. For example, by analyzing 1,767,272 *dbo:Person* instances in DBpedia, Loupe extracts the cardinality distribution for *dbo:Person-dbo:deathDate* combination as illustrated in Table 2.

Table 2: Cardinality counts for *dbo:Person-dbo:deathDate*

Cardinality	Instances	Percentage
0	1,355,038	76.67%
1	404,069	22.87%
2	8,165	00.46%

During the feature engineering step, this raw profiling data is used to derive a set of features that can be used for predicting the cardinality. Concretely, we have derived 30 features from this data including min-max cardinalities, mean, mode, standard deviation, variance, and other distribution-related metrics. The complete list can be found here[5].

Range constraints: Loupe provides statistics about the number of IRIs, literals, and blank nodes for each property associated with instances of a given class, as shown in Table 3. The blank node counts are also generated by Loupe but are not shown in the table because there were no blank nodes in this example.

Table 3: Object node type information

Class-property	IRIs		Literals	
	Total	Distinct	Total	Distinct
dbo:Person-dbp:birthPlace	89,355	21,845	44,639	20,405
dbo:Person-dbp:name	21,496	15,746	115,848	100,931
dbo:Person-dbp:deathDate	127	111	65,272	32,449
dbo:Person-dbp:religion	8,374	786	6,977	407

[5]https://doi.org/10.6084/m9.figshare.5270710

Further, Loupe also extracts object type information by analyzing all the IRI and blank node objects. Table 4 shows an example of object type information by analyzing all the objects of the *dbo:Person-dbp:deathPlace* class-property combination. It contains the number of objects, the number of distinct objects of each class type and their respective percentages. As it can be seen, the objects of *dbo:Person-dbp:deathPlace* are typed with many different classes. In fact, most objects are typed with multiple classes (e.g., with equivalent classes, super classes). Also there are some objects that should not be associated (*i.e.*, inconsistent) with the *dbp:deathPlace* property, for example, a *Broadcaster* should not be a death place of a person. Further, there are some objects for which the type information is not available.

Table 4: Classes of dbo:Person-dbp:birthPlace objects

Object Class	Objects (89,355)		Distinct Objects (21,845)	
	Count	%	Count	%
schema:Place	71,748	80.29	16,502	75.54
dbo:Place	71,748	80.29	16,502	75.54
dbo:PopulatedPlace	71,542	80.07	16,353	74.86
dbo:Settlement	41,216	46.13	14,184	64.93
other rows are omitted for brevity				
schema:Product	2	00.00	2	00.01
dbo:Broadcaster	2	00.00	2	00.01
Unknown	9,790	10.95	2,888	13.22

Similarly, for literal objects Loupe extracts the information about their data types. Table 5 shows an example of extracted information for the class-property combination *dbp:Person-dbp:deathDate*. For each datatype, it shows the number of objects, number of distinct objects, and their corresponding percentages. This information provides heuristics about which should be the corresponding datatype.

Table 5: Datatypes of dbp:Person-dbp:deathDate literals

Datatype	Objects (65,272)		Distinct Objects (32,449)	
	Count	%	Count	%
xsd:date	39,761	60.92	26,726	82.36
xsd:integer	13,543	20.75	1,758	5.42
rdf:langString	6,388	9.79	3,512	10.82
xsd:gMonthDay	5,446	8.34	366	1.13
dt:second	113	0.17	66	0.20
xsd:double	20	0.03	20	0.06
dt:hour	1	0.00	1	0.00
Total	65,272	100	32,449	100

We use all the aforementioned information as features for the two tasks of detecting the object node kind and also detecting the class type for IRI objects and the datatype for literal objects.

4.3 Algorithm selection

In the algorithm selection step, the goal is to select the most appropriate algorithm for each of the tasks identified in the first step. As we have formulated both the cardinality prediction as well as the range prediction tasks as classification problems, we will describe the algorithm selection step in a generic way.

For selecting the algorithm, We look at the accuracy of that model as the number of correct predictions from all the predictions made. In classical machine learning approaches, it is known as classification accuracy. However, running an algorithm over a training dataset with different hyper-parameter settings will result in different models [6]. In particular, we are interested in selecting the best performing model. We used classifier accuracy to evaluate the predictive performance of various models. In detail, the steps for evaluating the predictive performance of a model are listed below:

(1) We want to increase the classifier accuracy by tweaking the learning algorithm and selecting the best performing model from a given hypothesis space. In our profile dataset we observed that rare events occur in the case of selected constraints as response variables where variations between two variables are less than 15%. We applied SMOTE (Synthetic Minority Over-sampling Technique) [9] for over-sampling the rare events. The SMOTE function over-samples response variables by using bootstrapping and k-nearest neighbor to synthetically create additional observations of that response variable.
(2) To reduce the variance of an accuracy score we need to ensure that each instance is used an equal number of times for training. We applied *k-fold cross validation*, where k is the number of splits to make in the dataset. In this approach we choose a value of k=10. This results in splitting the dataset into 10 parts (10 folds) and running the algorithm 10 times.
(3) It is important to establish the baseline performance on a predictive modeling problem. In this experimental analysis we applied the ZeroR classification method. ZeroR is a trivial classifier, but it gives a lower bound on the performance of a given dataset which should be significantly improved by more complex classifiers. As such, it is a reasonable test on how well the class can be predicted without considering the other attributes [34] .
(4) Finally, to identify the machine learning algorithm that is best-suited for the problem at hand, we performed a comparison between the selected algorithms. We chose 9 algorithms form 6 categories of machine learning approaches. Based on the highest value of classifier accuracy, we selected the best performing model from the hypothesis space.

4.4 Constraints Generation

Once the constraint prediction models are built, constraints can be generated.

Cardinally constraints: In the classification task for cardinality constraints, we identify five main types of cardinality classes: MIN0, MIN1, MIN1+, MAX1, and MAX1+. Out of these, MIN0 and MAX1+ do not put any constraints on the data, such that, any data will be valid for those cardinality types. Thus, if we detect those types we do not generate constraints. For other types, the corresponding SHACL property constraints are generated as illustrated by Listing 2.

Listing 2: Cardinality constraints

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

ex:DBpediaPerson a sh:NodeShape;
 sh:targetClass dbo:Person;
# for MIN1 and MIN1+
 sh:property [sh:path foaf:name;
  sh:minCount 1 ];

# for MAX1
 sh:property [ sh:path dbo:birthDate;
  sh:maxCount 1] .
```

Range constraints: For range constraints, we generate two types of constraints. First, we generate if the objects of a property associated with a given class are either *IRI*, *Literal*, *BlankNode*, and *BlankNodeOrIRI*. These nodetype constraints are represented as illustrated in Listing 3. Then, we generate the class type constrainsts for IRI objects and datatype constraints for literal objects.

Listing 3: Node type constraints

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

ex:DBpediaPerson a sh:NodeShape;
 sh:targetClass dbo:Person;
# node type IRI
 sh:property [sh:path dbp:birthPlace;
  sh:nodeKind sh:IRI;
  sh:or ( [sh:class schema:Place]
    [ sh:class dbo:Place ] )
  ];

# node type literal
 sh:property [ sh:path dbp:deathDate;
  sh:nodeKind sh:Literal;
  sh:datatype xsd:date ] .
```

4.5 Shape Generation

The goal of the final step is to combine all the constraints related to a given class and to generate an RDF Shape for the class. In the previous step, the constraints were generated for each class-property combination; for example, the cardinality constraints for *foaf:name* when associated with the *dbo:Person* class. Similarly, constraints are generated to all properties associated with instances of the *dbo:Person* class. All those constraints of a given class are combined together to generate the shape of the class.

5 EXPERIMENTAL SETUP

We performed an experimental analysis on the basis of the cardinality constraints of our approach on two KBs, namely DBpedia and 3cixty. It is interesting to see cardinality constraints in case of a given class, and property *i.e.* how many objects are linked to a subject. If we consider both the subject and the object based on a relation represented by a property, we can consider different cardinality patterns used by the Entity-Relationship (ER) models such as one to one (1-1), one to many (1-N), many to one (N-1), and many to many (N-N). Thus, we have formulated the cardinality prediction task as a classification problem. The goal of this experimental analysis is to evaluate our approach in the context of how many correct predictions were made. In particular, we evaluate the cardinality classifier performance compared to the baseline accuracy.

More specifically, one of the central problems of applying machine learning in KB graphs is to transform and identify features while retaining the subtleties of the information it contains. Our goal is to validate the prediction of two cardinality constraints, i.e., minimal and maximum cardinality from profiled data based on a gold standard. In this experiment, we created a gold standard with four human annotators labeling the data. The annotators mutually revised the annotations and a third annotator was called when there was no agreement.

For DBpedia, we created a gold standard of expected cardinalities for a subset of 174 properties. Similarly, for 3cixty we created one for a subset of 215 properties. In particular, the collected dataset presents cardinality information for each property associated with instances of a given class for 174 properties of DBpedia and 215 properties of 3cixty. The datasets can be accessed online[6]. We applied 10-fold cross validation for algorithm section tasks by splitting the dataset into 10 parts. For each algorithm training, it will be run on 90% of the data and testing on the 10%. In particular, it will use each data instance as a training instance exactly 9 times and each test instance 1 time. Our approach has been implemented with a prototype[7] written in R.

6 RESULT AND ANALYSIS

We perform model evaluation based on classifier performance in order to identify the best fitting model. To evaluate classifier performance, we used both the DBpedia and 3cixty datasets comparing with a baseline accuracy. In general, the accuracy score measures how many correct predictions are made compared to the gold standard. In our approach, we used the ZeroR [34] as baseline classification method for minimum and maximum cardinality. It is a simple classification method which relies on the target and ignores all predictors. ZeroR is based on predicting the most-frequent class in a target variable. For example, DBpedia minimum cardinality training set has two classes M0 and M1. As the frequency of M0 is 215, and that of M1 is 65, the ZeroR classifier will be based on M0. In the Table 6 we present the baseline accuracy for minimum and maximum cardinality for both datasets using ZeroR. Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods.

Table 6: Baseline accuracy for 3cixty KB and DBpedia KB

Knowledge base	Min. cardinality	Max. cardinality
3cixty	53.5%	80.5%
DBpedia	76.4%	52.9%

[6]https://github.com/rifat963/RDFShapeInduction/tree/master/dataset
[7]https://github.com/rifat963/RDFShapeInduction

We increased the predictive performance by applying SMOTE on the training dataset. The SMOTE [9] approach performs a combination of over-sampling the minority (abnormal) class and under-sampling the majority (normal) class. In our experiment, we applied an over-sampling value of 100 to double the quantity of positive cases, and an undersampling value of 200 to keep half of what was created as negative cases. It balances the classifier and achieves better performance than only under-sampling the majority class. The results of applying SMOTE on the training set are present in Table 7.

Table 7: DBpedia and 3cixty distribution of cardinality constraints

Distribution	Minimum Cardinality		Maximum Cardinality	
	MIN0	MIN1	MAX1	ANY
3cixty KB				
Without SMOTE	47.2%	52.8%	79.2%	20.8%
With SMOTE(100,200)	50%	50%	50%	50%
DBpedia KB				
Without SMOTE	76.5%	23.5%	53%	47%
With SMOTE(100,200)	50%	50%	50%	50%

In order to evaluate cardinality classifier accuracy, we choose the most common machine learning algorithms considering a wide range of categories. Classifier accuracy gives an insight to performance based on the ratio of the number of correct predictions out of all predictions made and presented as a percentage where 100% is the best an algorithm can achieve. We estimated classifier accuracy on nine algorithms covering seven categories of machine learning approaches. We applied 10-fold cross-validation together with parameter optimization to reduce the problem of choosing a set of optimal hyperparameters for a learning algorithm. In Table 8 we present the percentage of cardinality classifier accuracy for DBpedia KB and 3cixty KB. From Table 8, most of the machine learning algorithm has a high percentage of accuracy compared to baseline algorithm. In case of DBpedia KB, both minimum and maximum cardinality classifier have high percentage of accuracy such as Random Forest model has 97.6% for minimum cardinality and 97.03% for maximum cardianality. However, 3cixty KB minimum and maximum cardinality have significant variation in classifier performance. Such as, Naive Bayesian (58.3%) and Decision Tree algorithms C4.5 (55.56%) has near to baseline accuracy of 53.5% for minimum cardinality. However, Random forest model has a higher percentage of accuracy of 74.8%.

We further look into Kappa metric [5] which presents the same information but taking the class balance into account. Details results of Kappa value can be found online[8]. Kappa value also gave similar results for both use cases. For example, in Figure 2 we present 3cixty KB minimum cardinality Kappa value comparison with other models. From the Kappa value, random forest model has higher value of 0.65 compared to other model. Based on the classifier performance ensemble model, *Random Forest* has the highest percentage of accuracy for both the DBpedia and 3cixty datasets.

[8]https://rifat963.github.io/kappa/kapp.html

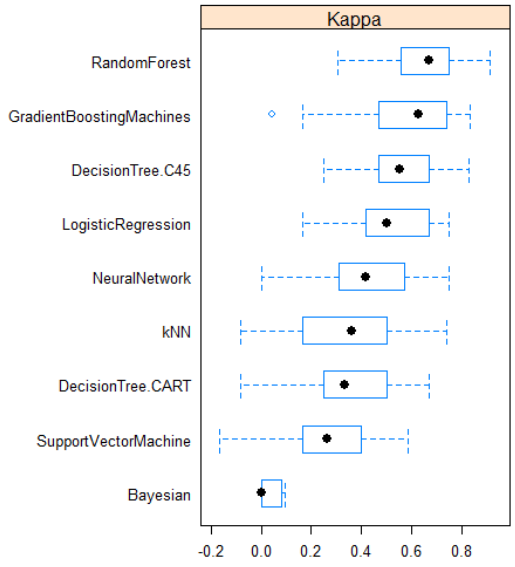

Figure 2: Box plot comparing model results using Kappa Value for 3cixty KB Minimum Cardinality.

Finally, we generated the cardinality constraints using the *Random Forest* model for DBpedia. The 174 property cardinality constraints generated using the Random Forest model can be found online for minimum cardinality[9] and for maximum cardinality[10].

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a KG data profiling-based RDF shape induction approach by using predictive modeling. The main motivation of the work presented in this paper is rooted in the concepts of RDF data validation and KB quality assessment. We tested our approach on two KBs, DBpedia, and 3cixty for generating cardinality constraints. Additionally, we explored the generation process of range constraints. We observed that our approach is extremely effective and was able to achieve 97.61% accuracy for DBpedia datasets for cardinality constraints by using a Random Forest model. Nevertheless, the proposed approach is defined in a generic and extensible manner which can be extended to other types of constraints such as string-based constraints.

So for further future work, we plan to use this approach for extracting other constraints which can be expressed through the W3C SHACL language. In particular, we will extend our implementation to other constraints such as string-based constraints, property pair constraints, and value range constraints. Furthermore, we would like to facilitate more efficient feature vectors for feature engineering by proposing improved classification strategies. We perceive that the scalability of our approach can be increased by using incremental methods, as for example, monitoring the changes in the underlying KB. We assume that if the changes remain uniform

[9]https://github.com/rifat963/RDFShapeInduction/blob/master/dataset/dbp-min-training.csv
[10]https://github.com/rifat963/RDFShapeInduction/blob/master/dataset/dbp-max-training.csv

Table 8: Cardinality Classifier Performance for DBpedia KB and 3cixty KB

Machine Learning Algorithm		DBpedia KB Classifier Accuracy		3cixty KB Classifier Accuracy	
		Minimum Cardinality	Maximum Cardinality	Minimum Cardinality	Maximum Cardinality
Baseline	ZeroR[34]	76.4%	52.9%	53.5%	80.5%
Regression	Logistic Regression [11]	92.2%	84.74%	66.67%	78.4%
Bayesian	Naive Bayes [5]	87.83%	84.04%	58.3%	69.44%
Support Vector Machine	Least Squares Support Vector Machine[30]	94.84%	88.67%	63.89%	79.31%
Instance Based	k-Nearest Neighbour[5]	92.64%	85.5%	63.89%	79.31%
Decision Tree	Classification and Regression Tree (CART)[29]	96.63%	87.29%	66.88%	76.29%
	C4.5[28]	95.64%	93.84%	55.56%	79.34%
Ensemble	Random Forest[15]	97.61%	97.03%	74.89%	81.52%
	Gradient Boosting Machines[12]	96.17%	94.86%	55.56%	79.89%
Neural Network	Perceptron [5]	95.1%	89.6%	61.11%	79.31%

in the underlying KB, then the constraint shapes generated can predict consistency issues by using our approach. This will help to detect quality issues which are automatically present in dynamic knowledge bases.

ACKNOWLEDGMENTS

This work was funded by the Spanish government with the BES-2014-068449 grant under the 4V project (TIN2013-46238-C4-2-R).

REFERENCES

[1] Ziawasch Abedjan and Felix Naumann. 2013. Improving RDF Data Through Association Rule Mining. *Datenbank-Spektrum* 13, 2 (01 Jul 2013), 111–120. https://doi.org/10.1007/s13222-013-0126-x

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. Foundations of databases: the logical level. (1995).

[3] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. 1996. Fast discovery of association rules. *Advances in knowledge discovery and data mining* 12, 1 (1996), 307–328.

[4] Adrien Basse, Fabien Gandon, Isabelle Mirbel, and Moussa Lo. 2010. DFS-based frequent graph pattern extraction to characterize the content of RDF Triple Stores. In *Web Science Conference 2010 (WebSci10)*.

[5] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[6] Peter Bloem and Gerben K. D. De Vries. 2014. Machine Learning on Linked Data, a Position Paper. In *Proceedings of the 1st International Conference on Linked Data for Knowledge Discovery - Volume 1232 (LD4KD'14)*. CEUR-WS.org, Aachen, Germany, Germany, 64–68. http://dl.acm.org/citation.cfm?id=3053827.3053834

[7] Eva Blomqvist, Ziqi Zhang, Anna Lisa Gentile, Isabelle Augenstein, and Fabio Ciravegna. 2013. Statistical knowledge patterns for characterising linked data. In *Proceedings of the 4th International Conference on Ontology and Semantic Web Patterns-Volume 1188*. CEUR-WS. org, 1–13.

[8] Lorenz Bühmann, Daniel Fleischhacker, Jens Lehmann, Andre Melo, and Johanna Völker. 2014. Inductive lexical learning of class expressions. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 42–53.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[10] Luc De Raedt, Tias Guns, and Siegfried Nijssen. 2010. Constraint programming for data mining and machine learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*. 1671–1675.

[11] David A Freedman. 2009. *Statistical models: theory and practice*. cambridge university press.

[12] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[13] Johannes Fürnkranz and Peter A Flach. 2005. Roc âĂŸnâĂŹrule learningâĂŤtowards a better understanding of covering algorithms. *Machine Learning* 58, 1 (2005), 39–77.

[14] Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. MIT press.

[15] Tin Kam Ho. 1995. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, Vol. 1. IEEE, 278–282.

[16] Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. 2010. Weaving the Pedantic Web. In *Proceedings of the Linked Data on the Web (LDOW 2010)*, Vol. 628. CEUR Workshop Proceedings.

[17] Theodore Johnson. 2009. Data Profiling. In *Encyclopedia of Database Systems*, LING LIU and M. TAMER ÖZSU (Eds.). Springer US, Boston, MA, 604–608. https://doi.org/10.1007/978-0-387-39940-9_601

[18] Hassan Khosravi and Bahareh Bina. 2010. A Survey on Statistical Relational Learning.. In *Canadian Conference on AI*. Springer, 256–268.

[19] Holger Knublauch and Dimitris Kontokostas. 2017. W3C Shapes Constraint Language (SHACL). (July 2017). https://www.w3.org/TR/shacl/

[20] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

[21] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[22] Stephen W Liddle, David W Embley, and Scott N Woodfield. 1993. Cardinality constraints in semantic data models. *Data & Knowledge Engineering* 11, 3 (1993), 235–270.

[23] Deborah L McGuinness, Frank Van Harmelen, et al. 2004. OWL web ontology language overview. *W3C recommendation* 10, 10 (2004), 2004.

[24] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. 2015. Loupe-An Online Tool for Inspecting Datasets in the Linked Data Cloud. In *Demo at the 14th International Semantic Web Conference*. Bethlehem, USA.

[25] Thomas Neumann and Guido Moerkotte. 2011. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 984–994.

[26] Eric Prud'hommeaux, Iovka Boneva, Jose Emilio Labra-Gayo, and Gregg Kellogg. 2017. Shape Expressions Language 2.0. (July 2017). http://shex.io/shex-semantics/

[27] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems*. ACM, 32–40.

[28] J Ross Quinlan. 2014. C4. 5: programs for machine learning. (2014).

[29] Dan Steinberg and Phillip Colla. 2009. CART: classification and regression trees. *The top ten algorithms in data mining* 9 (2009), 179.

[30] Johan AK Suykens, Tony Van Gestel, and Jos De Brabanter. 2002. *Least squares support vector machines*. World Scientific.

[31] Jiao Tao, Evren Sirin, Jie Bao, and Deborah L McGuinness. 2010. Extending OWL with Integrity Constraints. *Description Logics* 573 (2010).

[32] Giri Kumar Tayi and Donald P Ballou. 1998. Examining Data Quality. *Commun. ACM* 41, 2 (1998), 54–57.

[33] Raphael Troncy and Giuseppe Rizzo et al. 2017. 3cixty: Building Comprehensive Knowledge Bases for City Exploration. *Web Semantics: Science, Services and Agents on the World Wide Web* 46-47, Supplement C (2017), 2 – 13. https://doi.org/10.1016/j.websem.2017.07.002

[34] WEKA. 2013. *Weka Manual for Version 3-7-8*. Technical Report. WEKA. https://pdfs.semanticscholar.org/d617/d41097bdf97d994d1481adbcfe0c05a51696.pdf