Fixing the Domain and Range of Properties in Linked Data by Context Disambiguation

Alberto Tonon
eXascale Infolab
University of Fribourg
Switzerland
alberto@exascale.info

Michele Catasta
EPFL
Lausanne
Switzerland
michele.catasta@epfl.ch

Gianluca Demartini
Information School
University of Sheffield
United Kingdom
g.demartini@sheffield.ac.uk

Philippe Cudré-Mauroux
eXascale Infolab
University of Fribourg
Switzerland
phil@exascale.info

ABSTRACT

The amount of Linked Open Data available on the Web is rapidly growing. The quality of the provided data, however, is generally-speaking not fundamentally improving, hampering its wide-scale deployment for many real-world applications. A key data quality aspect for Linked Open Data can be expressed in terms of its adherence to an underlying well-defined schema or ontology, which serves both as a documentation for the end-users as well as a fixed reference for automated processing over the data. In this paper, we first report on an analysis of the schema adherence of domains and ranges for Linked Open Data. We then propose new techniques to improve the correctness of domains and ranges by i) identifying the cases in which a property is used in the data with several different semantics, and ii) resolving them by updating the underlying schema and/or by modifying the data without compromising its retro-compatibility. We experimentally show the validity of our methods through an empirical evaluation over DBpedia by creating expert judgements of the proposed fixes over a sample of the data.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous

General Terms

Experimentation, Algorithms

Keywords

Linked Open Data, Schema adherence, Data quality

1. INTRODUCTION

Linked Open Data (LOD) is rapidly growing in terms of the number of available datasets moving from 295 available datasets in 2011 to 1'014 datasets in 2014[1]. As we report in Section 2, LOD quality has already been analyzed from different angles; one key LOD quality issue is the fact that the data does not always adhere to its corresponding schema, as we discuss in more detail in Section 3. That is, factual statements (i.e., RDF triples) do not always follow the definitions given in the related RDF Schemas or ontologies. Having a schema which the published data adheres to allows for better parsing, automated processing, reasoning, or anomaly detection over the data. Also, it serves as a de facto documentation for the end-users querying the LOD datasets, fostering an easier deployment of Linked Data in practice.

To mitigate the issues related to the non-conformity of the data, statistical methods for inducing the schema over the data have been proposed. Voelker et al. [9], for example, extract OWL-EL axioms from the data and use statistics to compute confidence values on the axioms. Similar statistics were also used in order to detect inconsistencies in the data [8].

In this work, we focus on one particular issue of LOD schema adherence: the proper definition of the properties' domains and ranges in LOD. More precisely, we propose (see Section 4) a new data-driven technique that amends both the schema and the instance data in order to assign better domains and ranges to properties; this goal is achieved by detecting the cases through which a property is used for different purposes (i.e., with different semantics) and by disambiguating its different uses by dynamically creating new sub-properties extending the original property. Thus, our approach modifies both the schema (new sub-properties are created) and the data (occurrences of the original property that were used with some given semantics are replaced with the newly created sub-property). One of the interesting properties of our approach is that the modified data is retro-compatible, that is, a query made over the original version of the data can be posed *as is* over the amended version.

We evaluate our methods in Section 5 by first comparing how much data it can fix by adjusting different parameters, and then by asking Semantic Web experts to judge the quality of the modifications suggested by our approach.

2. RELATED WORK

One of the most comprehensive piece of work describing LOD is the article by Schmachtenberg et al. [7] in which the adoption of best practices for various aspects, from creation

[1] `http://lod-cloud.net`

WWW2015 Workshop: Linked Data on the Web (LDOW2015).

to publication, of the 2014 LOD are analyzed. Such practices, ultimately, are meant to preserve the quality of a large body of data as LOD—a task that is even more daunting, considering the inherently distributed nature of LOD.

Data quality is a thoroughly-studied area in the context of companies [6], because of its importance in economic terms. Recently, LOD did also undergo a similar scrutiny: in [4], the authors show that the Web of Data is by no means a perfect world of consistent and valid facts. Linked Data has multiple dimensions of shortcomings ranging from simple syntactical errors over logical inconsistencies to complex semantic errors and wrong facts. For instance, Töpper et al. [8] statistically infer the domain and range of properties in order to detect inconsistencies in DBpedia. Similarly, Bizer et al. in [5] propose a data-driven approach that exploits statistical distributions of properties and types for enhancing the quality of incomplete and noisy Linked Data sets, specifically for adding missing type statements, and identifying faulty statements. Differently from us, they leverage the number of instances of a certain type appearing in the property's subject and object position in order to infer the type of an entity, while we use data as evidence to detect properties used with different semantics.

There is also a vast literature ([9, 3, 2, 1]) that introduces statistical schema induction and enrichment (based on association rule mining, logic programming, etc.) as a means to generate ontologies from RDF data. Such methods can for example extract OWL axioms and then use probabilities to come up with confidence scores, thus building what can be considered a "probabilistic ontology" that can emerge from the messiness and dynamicity of Linked Data. In this work, we focus on the analysis of property usage with the goal of fixing Linked Data and improve its quality.

3. MOTIVATION AND BASIC IDEAS

The motivation that led us to the research we are presenting is summarized in Table 1. Its upper part reports the top-5 properties in DBpedia[2] and Freebase[3] The table reports on the number of times the properties appear with a wrong domain, together with their Wrong Domain Rate (WDR), that is, the ratio between the number of times the property is used with a wrong domain to its total number of uses. Analogously, the lower part of the table reports on the top-5 properties by number of range violations and their Wrong Range Rate (WRR).[4] We observe that the absolute number of occurrences of wrong domains/ranges in Freebase is two orders of magnitude greater than that of DBpedia. This cannot be explained only by the different number of entities contained in the two knowledge bases since the number of topics covered by Freebase is only one order of magnitude greater than that of DBpedia (approximately 47.43 and 4.58 million topics, respectively, according to their Web-pages). We deduce that in Freebase the data adheres to the schema less than in DBpedia. This is also suggested by the fact that the top-3 most frequent properties defined in the DBpedia ontology, namely `dpo:birthPlace`, `dpo:birthYear`, and `birthDate`, have WDR and WRR smaller than 0.01, while the top-3 most used property in Freebase, namely `fb:type.object.type`, `fb:type.type.instance`, and `fb:type.object.key`, have an average WDR of 0.30 and an average WRR of 0.87. This disparity can in part be explained by the fact that the Freebase ontology is a forest of trees rather than a tree with a single root note (as in DBpedia). Thus, while one could expect that each entity in the dataset should descend from 'object', this is not the case when looking at the data. In addition, we noticed that in DBpedia, out of the 1'368 properties actually used in the data, 1'109 have a domain declaration in the ontology and 1'181 have a range declaration. Conversely, Freebase specifies domain and range of 65'019 properties but only 18'841 properties are used in the data.

In this paper we argue that a number of occurrences of wrong domains or ranges are due to the fact that the same property is used in different contexts, thus with different semantics. The property `dpo:gender`, for example, whose domain is not specified in the DBpedia ontology, is used both to indicate the gender of a given person and the gender of a school (that is, if it accepts only boys, girls or both). Hence, `dpo:gender` appears both in the context of `dpo:GivenName` and of `dpo:School`. While this can make sense in spoken language, we believe that the two cases should be distinct in a knowledge base. However, we cannot make a general rule out of this sole example as, for instance, we have that `foaf:name` (whose domain is not defined in the DBpedia ontology) is attached to 25 direct subtypes of `owl:Thing` out of 33; these types include `dpo:Agent` (the parent of `dpo:Person` and `dpo:Organization`), `dpo:Event`, and `dpo:Place`. In this case, it does not make sense to claim that all these occurrences represent different contexts in which the property appears, since the right domain for this case is indeed `owl:Thing`, as specified by the FOAF Vocabulary Specification.[5] Moreover, in this case creating a new property for each subtype would lead to an overcomplicated schema. Finally, the fact that `dpo:name` is not attached to all the subtypes of `owl:Thing` suggests that the property is optional.

What follows describes the intuition given by this example in terms of statistics computed on the knowledge base. In addition, we also present algorithms to identify the use of properties in different contexts.

4. DETECTING AND CORRECTING MULTI-CONTEXT PROPERTIES

In this section, we describe in detail the algorithm we propose, namely, LeRiXt (LEft and RIght conteXT). For the sake of presentation, we first describe a simpler version of the method we call LeXt (LEft conteXT) that uses the types of the entities appearing as subjects of the property in order to identify properties that are used in different contexts (*multi-context properties*). We then present the full algorithm as an extension of this simpler version. For the description of the algorithm, we make use of the notation defined in Table 2.

4.1 Statistical Tools

[2]We used the English version of DBpedia 2014 (`http://dbpedia.org/Downloads2014`).

[3] We used a dump downloaded on March 30th 2014 (`http://freebase.com`).

[4]When computing WDR and WRR we do take into account the type hierarchy for computing the violation rate. That is, if a property has 'Actor' as range and is used in a RDF triple where the object is an 'American Actor' we consider it as correct as 'American Actor' is a subtype of 'Actor'.

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

Table 1: Top-5 properties by absolute number of domain violations (top), and range violations (bottom), with their domain/range violation rate (the truncated properties are `fb:dataworld.gardening_hint.last_referenced_by` and `fb:common.topic.topic_equivalent_webpage`).

DBpedia property	#Wrong Dom.	WDR	Freebase Property	#Wrong Dom.	WDR
dpo:years	641'528	1.00	fb:type.object.type	99'119'559	0.61
dpo:currentMember	260'412	1.00	fb:type.object.name	41'708'548	1.00
dpo:class	255'280	0.95	fb:type.object.key	35'276'872	0.29
dpo:managerClub	47'324	1.00	fb:type.object.permission	7'816'632	1.00
dpo:address	36'449	0.90	fb:[...].last_referenced_by	3'371'713	1.00
DBpedia property	**#Wrong Rng.**	**WRR**	**Freebase Property**	**#Wrong Rng.**	**WRR**
dpo:starring	298'713	0.95	fb:type.type.instance	96'764'915	0.61
dpo:associatedMusicalArtist	70'307	0.64	fb:[...]topic_equivalent_webpage	53'338'833	1.00
dpo:instrument	60'385	1.00	fb:type.permission.controls	7'816'632	1.00
dpo:city	55'697	0.55	fb:common.document.source_uri	4'578'671	1.00
dpo:hometown	47'165	0.52	fb:[...].last_referenced_by	3'342'789	0.99

Table 2: Notation used for describing LeRiXt.

Symbol	Meaning
KB	the knowledge base composed of triples (s,p,o) s.t. $s \in E \cup T$, $p \in P$, $o \in E \cup L \cup T$ with E set of all entities, P set of all properties, T set of all entity types, and L set of all literals.
$\top$	the root of the type hierarchy.
e, t	an entity and an entity type, respectively.
$e\,\mathrm{a}\,t$	$(e, \mathrm{a}, t) \in KB$, that is, e is an instance of t
p	a property.
t^L	an entity type t on the left side of a property.
t^R	an entity type t on the right side of a property.
$\mathrm{Par}(t)$	the parent of a type t in the type hierarchy.
$\mathrm{Ch}(t)$	the set of children of a type t in the type hierarchy.
$\mathrm{Cov}(p')$	the coverage of a sub-property p' of a property p, that is, the rate of occurrences of p covered by p'.

LeXt makes use of two main statistics: $\Pr(t^L \mid p)$, that is, the conditional probability of finding an entity of type t as the subject of a triple having p as predicate (i.e., finding t "to the Left" of p), and the probability $\Pr(p \mid t^L)$, that is, the probability of seeing a property p given a triple whose subject is an instance of t. Equation 1 formally defines those two probabilities.

$$\begin{aligned}\Pr(t^L \mid p) &= \frac{|\{(s,p',o) \in KB \mid s\,\mathrm{a}\,t,\, p = p'\}|}{|\{(s,p',o) \in KB \mid p = p'\}|}\\ \Pr(p \mid t^L) &= \frac{|\{(s,p',o) \in KB \mid s\,\mathrm{a}\,t,\, p = p'\}|}{|\{(s,p',o) \in KB \mid s \in t\}|}\end{aligned} \tag{1}$$

As one can imagine, $\Pr(t^L \mid p) = 1$ indicates that t is a suitable domain for p, however, t can be very generic. In particular $\Pr(\top^L \mid p) = 1$ for every property p where $\top$ is the root of the type hierarchy. Conversely, $\Pr(p \mid t^L)$ measures how common a property is among the instances of a certain type. $\Pr(p \mid t^L) = 1$ suggests that the property is mandatory for t's instances. In addition, whenever we have strong indicators that a property is mandatory for many children t_i of a given type t, that is, $\Pr(p \mid t_i^L)$ is close to 1 for all t_is, we can deduce that t is a reasonable domain for p and that all the t_i are using p as an inherited (possibly optional) property. For example, if in DBpedia we consider the property `foaf:name` and we analyze $\Pr(p \mid t_i^L)$ for all $t_i \in \mathrm{Ch}($`owl:Thing`$)$ we see that the probability is greater than 0 in 25 cases out of 33 and is greater than 0.50 in 18 cases, suggesting that all the t_is do not constitute uses of the properties in other contexts but rather that the properties are used in the more general context identified by `owl:Thing`.

Computationally, we only need to maintain one value for each property p and for each type t, that is the number $\#(p \wedge t^L)$ of triples having as subject an instance of t and p as predicate. In fact, if we assume that whenever there is a triple stating that $(e, \mathrm{a}, t) \in KB$ there is also a triple $(e, \mathrm{a}, t') \in KB$ for each ancestor t' of t in the type hierarchy, we have that

$$\forall p \in P. |\{(s,p',o) \in KB \mid p = p'\}| = \#(p \wedge \top^L),$$
$$\forall p \in P. |\{(s,p',o) \in KB \mid s \in t\}| = \sum_{p' \in P} \#(p' \wedge t^L).$$

The computation of all the $\#(p \wedge t^L)$ can be done with one map/reduce job similar to the well-known word-count example often used to show how the paradigm works, thus, it can be efficiently computed in a distributed environment allowing the algorithms we propose to scale to large amounts of data. Another interesting property implied by the type subsumptions of the underlying type hierarchy is that if $t_1 \in \mathrm{Ch}(t_0)$ then $\Pr(t_1^L \mid p) \leq \Pr(t_0^L \mid p)$. Assuming the same premises, however, nothing can be said about $\Pr(p \mid t_0^L)$ and $\Pr(p \mid t_1^L)$.

4.2 LeXt

As previously anticipated, LeXt detects multi-context properties by exploiting the types of the entities found on the left-hand side of the property taken into consideration. Specifically, given a property p, the algorithm makes a depth-first search of the type hierarchy starting from the root to find all cases for which there is enough evidence that the property is used with a different context. Practically, at each step, a type t—the current root of the tree—is analyzed and all the $t_i \in \mathrm{Ch}(t)$ having $\Pr(t_i^L \mid p)$ greater than a certain threshold λ are considered. If there is no such child, or if we are in a case similar to that of the `foaf:name` example described previously, a new sub-property t_p of p is created with t as domain; otherwise the method is recursively called on each t_i. Finally, cases analogous to the `foaf:name` example are

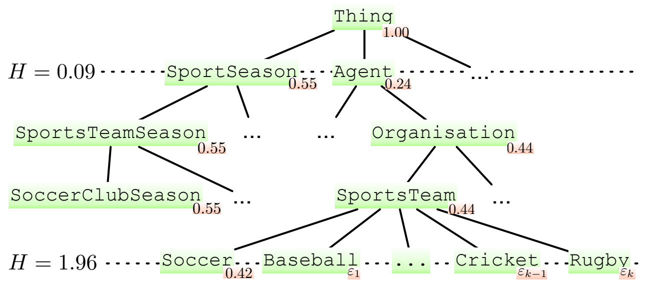

Figure 1: Execution of LeXt on `dpo:manager`.

detected by using the entropy of the probabilities $\Pr(p \mid t_i^D)$ with $t_i \in \text{Ch}(t)$ that captures the intuition presented while introducing the above mentioned statistics. Since, in general, $\sum_{t_i \in \text{Ch}t} \Pr(p \mid t_i^L) \neq 1$, we normalize each probability by dividing it by $Z = \sum_{t_i \in \text{Ch}(t)} t_i$ and we compute the entropy H using Equation 2.

$$H\big(p \mid \text{Ch}(t)\big) = -\sum_{t_i \in \text{Ch}(t)} \frac{\Pr(p \mid t_i)}{Z} \cdot \log_2\left(\frac{\Pr(p \mid t_i)}{Z}\right) \quad (2)$$

Algorithm 1 formally describes the full process. In the pseudo-code, a context of the input property is encoded with a triple $(p', \text{dom}(p'), \textit{coverage})$ where p' is a property identifying the context, $\text{dom}(p')$ is its domain, and $\textit{coverage} \geq \lambda$ is the rate of the occurrences of p covered by the context, denoted by $\text{Cov}(p')$. If the coverage is one, p is used in just one context (see Line 5). In Line 8, a new property p' is created and its domain is set to *curr_root*, while in Line 9, p' is declared to be a sub-property of p: this makes the data retro-compatible under the assumption that the clients can resolve sub-properties. Ideally, after the execution of the algorithm, all the triples referring to the identified meanings should be updated. The algorithm can also be used to obtain hints on how to improve the knowledge base.

The execution steps of the algorithm on `dpo:manager` (m, for short) with $\lambda = 0.4$ and $\eta = 1$ are depicted in Figure 1. The entity types are organized according to the DBpedia type hierarchy and each type t is subscripted by $\Pr(t \mid m)$. As can be observed, during the first step the children of owl:Thing are analyzed: the entropy constraint is satisfied and two nodes satisfy the $\Pr(t \mid m)$ constraint. The exploration of the `dpo:sportsSeason` branch ends when `dpo:SoccerClubSeason` is reached. The triple (`SoccerClubSeason manager`, `dpo:SoccerClubSeason`, 0.55) is returned. The new property is a sub-property of `dpo:manager` that covers 55% of the occurrences. Finally, the algorithm goes down the other branch until the entropy constraint is violated and returns the context (`SportsTeam manager`, `dpo:SportsTeam`, 0.45).

4.3 Discussion

The threshold λ sets a condition on the minimum degree of evidence we need to state that we have identified a new meaning for p, expressed in term of the $\Pr(t \mid p)$ probability. This threshold is of key importance in practice. On the one hand, low thresholds require little evidence and thus foster the creation of new properties, possibly over-populating the schema. On the other hand, high thresholds almost never accept a new meaning of a property, thus inferring coarser domains. In particular, with $\lambda = 1$ the exact domain of p is inferred (which in several cases can result to be $\top$). In

Algorithm 1 LeXt

```
Require: $0 \leq \lambda \leq 1$ strictness threshold, $\eta \geq 0$ entropy threshold.
Require: $curr\_root \in T$ the current root, $p \in P$.
Require: $acc$ a list containing all the meanings found so far.
Ensure: $acc$ updated with all the meanings of $p$.
 1: $p\_given\_t \leftarrow \{ \Pr(p \mid t_c^D)) \mid t_c \in \text{Ch}(curr\_root) \}$
 2: $H = \text{Entropy}(p\_given\_t)$
 3: $candidates \leftarrow \{ t_c \mid t_c \in \text{Ch}(curr\_root) \wedge \Pr(t_c^D \mid p) \geq \lambda \}$
 4: if $H \geq \eta \vee candidates = \emptyset$ then
 5:     if $\Pr(curr\_root \mid p) = 1$ then
 6:         $acc \leftarrow (p, curr\_root, 1) : acc$
 7:     else
 8:         $p' \leftarrow \text{new\_property}(p, curr\_root)$
 9:         $KB \leftarrow KB \cup \{ (p', \texttt{rdfs:subPropertyOf}, p) \}$
10:         $acc \leftarrow (p', curr\_root, \Pr(curr\_root' \mid p)) : acc$
11:     end if
12: else
13:     for $c \in candidates$ do
14:         $\text{LeXt}(\lambda, \eta, c, acc)$
15:     end for
16: end if
```

Section 5 we show how the algorithm behaves with varying levels of strictness.

The presented algorithm has a number of limitations. In particular, it does not explicitly cover the cases for which one type has more than one parent, thus multi-inheriting from several other types. In that case, an entity type can be processed several times (at most once per parent). We leave to future work studying if simply making sure that each node is processed once is enough to cover that case.

4.4 ReXt and LeRiXt

It is straightforward to define a variant of LeXt that considers property ranges instead of property domains by using $\Pr(t^R \mid p)$ and $\Pr(p \mid t^R)$. We call this method ReXt. In our implementation *we only consider object properties*, that is, properties that connect an entity to another entity (rather than, for example, to a literal since these values are not entities and thus are not in the type hierarchy).

Generalizing LeXt to identify multi-context properties based on both domains and ranges is a more complicated task. The solution we propose is called LeRiXt and consists in using two copies of the type hierarchy, one for the domains, and one for the ranges. At each step there is a "current domain" t_d and a "current range" t_r whose children are analyzed (thus the algorithm takes one more parameter than LeXt). Instead of using the condition $\Pr(t^D \mid p) \geq \lambda$ to select the candidate types to explore, we use $\Pr(t_i^D \wedge t_j^R \mid p) \geq \lambda$ for each $t_i \in \text{Ch}(t_d), t_j \in \text{Ch}(t_r)$, and we recursively call LeRiXt for each pair of types satisfying the constraint (see Line 14 of Algorithm 1).

5. EXPERIMENTS

We empirically evaluate the three methods described in Section 4, namely, LeXt, ReXt, and LeRiXt, first by studying how they behave when varying the threshold λ, and then by measuring the precision of the modifications they suggest. The LOD dataset we selected for our evaluation is DBpedia 2014 since its entity types are organized in a well-specified tree, contrary to Freebase, whose type system is a forest. As we anticipated in Section 4.4, we consider only object properties when the range is used to identify multi-context properties by using ReXt and LeRiXt. The

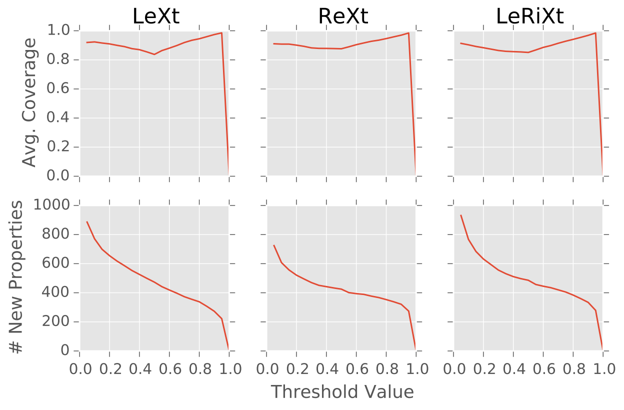

Figure 2: Average coverage and number of new properties with varying values of the threshold λ.

numbers of properties we take into consideration when running LeXt and the other two algorithms are 1'368 and 643, respectively. Finally, during our experimentation we fix the η threshold to 1. This value was chosen based on the analysis of the entropy stopping criterion on a small subset of properties.

The impact of λ on the output of the algorithms is studied in terms of average property coverage and number of generated sub-poperties. Recall that in Section 4.2 we defined the coverage of a sub-property. Here we measure the *property coverage*, defined as the overall rate of occurrences of a certain property p that is covered by its sub-properties, that is, the sum of $\text{Cov}(p')$ for all p' generated sub-property of p.

In the upper part of Figure 2 the average over the property coverage is shown for various λ. We notice that, as expected, lower values of λ lead to a high coverage since many new properties covering small parts of the data are created. As the value of the threshold increases, fewer and fewer properties are created, reaching the minimum at $\lambda = 1$. Interestingly, we observe that the average coverage curve is M-shaped with a local minimum at $\lambda = 0.5$. That is the consequence of the fact that with $\lambda \geq 0.5$ the new properties are required to cover at least half of the occurrences of the original property, leaving no space for other contexts, thus, at most one new context can be identified for each property. Finally, at $\lambda = 1$ the average coverage drops to 0 since no sub-property can cover all the instances of the original property.

In order to evaluate the output produced by the methods, 3 authors and 2 external experts evaluated the output of the algorithms computed on a sample of fifty randomly selected DBpedia properties using $\lambda = 0.1$ and $\eta = 1$. To decide whether the context separation proposed by the algorithm is correct or not, we built a web application showing to the judges the clickable URI of the original property together with the types of the entities it appears with. The judges had then to express their opinion on every generated sub-property.

The judgments were aggregated by majority vote and then precision was computed by dividing the number of positive judgments by the number of all judgments. LeXt, ReXt, and LeRiXt achieved a precision of 96.50%, 91.40%, and 87.00%, respectively.

We note that this result was obtained with just one configuration of the parameters—we leave a deeper evaluation of the algorithm as future work.

In practice, we envision our algorithms to be used as a decision-support tool for LOD curators rather than a fully automatic system to fix LOD datasets.

6. CONCLUSIONS

In this paper, we tackled the problem of extracting and then amending domain and range information from LOD. The main idea behind our work stems from the observation that many properties are misused at the instance level or used in several, distinct contexts. The three algorithms we proposed, namely, LeXt, ReXt, and LeRiXt, exploit statistics about the types of the entities appearing as subject and object in the triples involving the property analyzed in order to identify the various cases in which a multi-context property is used. Once a particular context is identified, a new sub-property is derived such that occurrences of the original property can be substituted using the newly generated sub-property. Our methods can also be used to provide insight into the knowledge base analyzed and how it should be revised in subsequent iterations. We evaluated our methods by studying their behavior with different parameter settings and by asking Semantic Web experts to evaluate the generated sub-properties.

The algorithms we propose require the entities contained in the dataset to be typed with types organized in a tree-structured type hierarchy. As future work, we plan to run a deeper evaluation of our techniques, and to design a method that overcomes the limitation presented above by considering the case in which the entity types are organized in a Direct Acyclic Graph, thus supporting multiple inheritance.

Acknowledgments

This work was supported by the Haslerstiftung in the context of the Smart World 11005 (Mem0r1es) project and by the Swiss National Science Foundation under grant number PP00P2_128459.

7. REFERENCES

[1] L. Bühmann and J. Lehmann. Universal OWL axiom enrichment for large knowledge bases. *LNCS*, 7603 LNAI:57–71, 2012.

[2] C. d'Amato, N. Fanizzi, and F. Esposito. Inductive learning for the semantic web: What does it buy? *Semantic Web*, 1(1):53–59, 2010.

[3] G. A. Grimnes, P. Edwards, and A. Preece. Learning meta-descriptions of the foaf network. In *The Semantic Web–ISWC 2004*, pages 152–165. Springer, 2004.

[4] M. Knuth and H. Sack. Data Cleansing Consolidation with PatchR. In *ESWC*, volume 8798 of *LNCS*, pages 231–235. Springer, 2014.

[5] H. Paulheim and C. Bizer. Improving the Quality of Linked Data Using Statistical Distributions. *I. J. Semantic Web Inf. Syst.*, 10(2):63–86, Jan. 2014.

[6] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211, 2002.

[7] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *ISWC*, pages 245–260, 2014.

[8] G. Töpper, M. Knuth, and H. Sack. DBpedia ontology enrichment for inconsistency detection. *I-SEMANTICS*, page 33, 2012.

[9] J. Völker and M. Niepert. Statistical schema induction. *LNCS*, 6643 LNCS:124–138, 2011.