ABSTAT: Ontology-driven Linked Data Summaries with Pattern Minimalization[*]

Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino

University of Milano-Bicocca
`firstname.lastname@disco.unimib.it`

Abstract. An increasing number of research and industrial initiatives have focused on publishing Linked Open Data, but little attention has been provided to help consumers to better understand existing data sets. In this paper we discuss how an ontology-driven data abstraction model supports the extraction and the representation of summaries of linked data sets. The proposed summarization model is the backbone of the ABSTAT framework, that aims at helping users understanding big and complex linked data sets. The proposed model produces a summary that is correct and complete with respect to the assertions of the data set and whose size scales well with respect to the ontology and data size. Our framework is evaluated by showing that it is capable of unveiling information that is not explicitly represented in underspecified ontologies and that is valuable to users, e.g., helping them in the formulation of SPARQL queries.

Keywords: data summarization, knowledge patterns, linked data

1 Introduction

As of April 2014 up to 1014 data sets have been published in the Linked Open Data cloud, a number that is constantly increasing[1]. However, a user may find it difficult to understand to what extent a data set covers a domain of interest and structures its content [11,22,19,15,7]. Given a Linked data set, users should be able to answer to questions such as: What types of resources are described in each data set? What properties are used to describe the resources? What types of resources are linked and by means of what properties? How many resources have a certain type and how frequent is the use of a given property? Remarkably, difficulties in answering those questions have several consequences for data consumption, resulting in low adoption of many valuable but unknown data sets [17].

Linked data sets make use of ontologies to describe the semantics of their data. However, answering the above questions by only looking at ontologies is

[*] The final authenticated version is available online at 10.1007/978-3-319-47602-5_51

[1] `http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/`

not easy. Ontologies might be large. For example, at the time of writing DBpedia uses 685 (local) concepts and 2795 properties. Ontologies used to model a large amount of diverse data in a flexible way are often underspecified. The domain is underspecified for 259 properties of the DBpedia Ontology, while the range is underspecified for 187 properties. In relatively expressive ontologies like the Music Ontology, some connections between types may be specified by means of OWL axioms, e.g., qualified range restrictions, which may be difficult to understand for many data practitioners.

Finally, the ontology does not tell how frequently certain modelling patterns occurs in the data set. Answers to the above questions can be collected with explorative queries, but at the price of a significant server overload for data publishers and high response time for data consumers.

ABSTAT is an ontology-driven linked data summarization model proposed to mitigate the data set understanding problem. In our view, a summary is aimed at providing a compact but complete representation of a data set. With complete representation we refer to the fact that every relation between concepts that is not in the summary can be inferred. One distinguishing feature of ABSTAT is to adopt a minimalization mechanism based on *minimal type patterns*. A minimal type pattern is a triple (C, P, D) that represents the occurrences of assertions $<a,P,b>$ in RDF data, such that C is a minimal type of the subject a and D is a minimal type of the object b. Minimalization is based on a *subtype graph* introduced to represent the data ontology. By considering patterns that are based on minimal types we are able to exclude several redundant patterns from the summary and to specify several formal properties of the summaries. As a consequence, summaries based on our model are rich enough to represent adequately the whole data set, and small enough to avoid redundant information. The ABSTAT[2] framework supports users to query (via SPARQL), to search and to navigate the summaries through web interfaces. Other related work on data or ontology summarization have focused on complementary aspects of the summarization, such as the identification of *salient subsets* of knowledge bases using different criteria [22,7,19,15], e.g., connectivity. Other approaches do not represent connections between instance types as our model does [8,1,9].

In this paper we make the following contributions: (i) we describe in detail the summarization model, focusing on the minimalization approach; (ii) we describe the summary extraction workflow; (iii) we provide an experimental evaluation of our approach from two different perspectives, evaluating the compactness and the informativeness of the summaries.

The paper is organized as follows. The summarization model is presented in Section 2. The implementation of the model in ABSTAT is given in Section 3. Experimental results are presented in Section 4. Related work is discussed in Section 5 while conclusions end the paper in Section 6.

[2] `http://abstat.disco.unimib.it`

2 Summarization Model

Ontologies (or vocabularies) are often used to specify the semantics of data modelled in RDF. Ontologies, which are usually represented in languages such as RDFS and OWL2, specify the meanings of the elements of the ontology, e.g., concepts, datatypes, properties, individuals, by means of logical axioms [18]. Although we do not focus on a specific ontological language, 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* defininining the vocabulary of an application domain, and a set of *assertions* describing RDF resources in terms of this vocabulary.

We define a data set 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.

Observe that different data set adopt different policies with respect to the inclusion of entailed assertions in the published assertions: for example, DBpedia explicitly includes the transitive closure of type inference in the published assertion set, while other data sets do not follow the same policy, e.g., LinkedBrainz. Our summarization model has to handle data sets that may have been published following different inference publication policies. However, we will briefly discuss the impact of different inference publication policies on the summarisation model in the next sections.

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`. Observe that 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`).

Abstract Knowledge Patterns (AKPs) 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 [18]. For sake of clarity, we will use the term *pattern* to refer to an AKP in the rest of the paper. A pattern is a triple (C, P, D) such that C and D are types and P is a property. Intuitively, an AKP states that there are instances of type C that are linked to instances of a type D by a property P. In ABSTAT we represent a set of AKP occurring in the data set, which profiles the usage of the terminology. However, instead of representing every AKP 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. In the following we better define these concepts and the ABSTAT principles.

Pattern Occurrence. A pattern (C, P, D) *occurs* in a set of assertions $\mathcal{A}$ iff there exist some instances x and y such that $\{C(x), D(y), P(x, y)\} \subseteq \mathcal{A}$. Patterns will be also denoted by the symbol π.

For data sets that publish the transitive closure of type inference (e.g., DBpedia), the set of all patterns occurring in an assertion set may be very large and include several redundant patterns. To reduce the number of patterns we use the observation that many patterns can be derived from other patterns if we use a *Subtype Graph* that represents types and their subtypes.

Subtype Graph. A *subtype graph* is a graph $G = (\mathsf{N}^C, \preceq)$, where N^C is a set of type names (either concept or datatype names) and $\preceq$ is a relation over N^C.

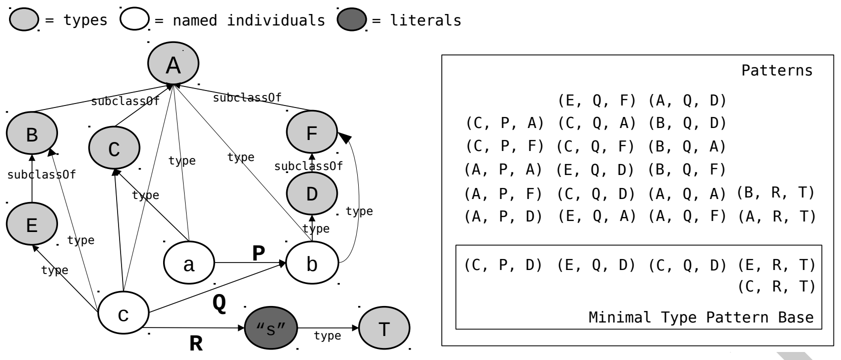
We always include two type names in N^C, namely `owl:Thing` and `rdfs:Literal`, such that every concept is subtype of `owl:Thing` and every datatype is subtype of `rdfs:Literal`. One type can be subtype of none, one or more than one type.

Minimal Type Pattern. A pattern (C, P, D) is a *minimal type pattern* for a relational assertion $P(a, b) \in \mathcal{A}$ and a terminology graph G iff (C, P, D) occurs in $\mathcal{A}$ and there does not exist a type C' such that $C'(a) \in \mathcal{A}$ and $C' \prec^G C$ or a type D' such that $D'(b) \in \mathcal{A}$ and $D' \prec^G D$.

Minimal Type Pattern Base. A *minimal type pattern base* for a set of assertions $\mathcal{A}$ under a subtype graph G is a set of patterns $\widehat{\Pi}^{\mathcal{A},G}$ such that $\pi \in \widehat{\Pi}^{\mathcal{A},G}$ iff π is a minimal type pattern for some relation assertion in $\mathcal{A}$.

Observe that different minimal type patterns (C, P, D) can be defined for an assertion $P(a, b)$ if a and/or b have more than one minimal type. However, the minimal type pattern base excludes many patterns that can be inferred following the subtype relations and that are not minimal type for any assertion. In the graph represented in Figure 1 considering the assertion set $\mathcal{A} = \{P(a, b), C(a), A(a), F(b), D(b), A(b)\}$, there are six patterns occurring in $\mathcal{A}$, i.e., (C, P, D), (C, P, F), (C, P, A), (A, P, D), (A, P, F), (A, P, A). The minimal type pattern base for the data set includes the patterns (E, Q, D), (E, R, T), (C, Q, D), (C, R, T) and (C, P, D) since E and C are minimal types of the instance c, while excluding patterns like (B, Q, D) or even (A, Q, A) since not B nor A are minimal types of any instance.

Data Summary. A *summary* of a data set $\Delta = (\mathcal{A}, \mathcal{T})$ is a triple $\Sigma^{\mathcal{A},\mathcal{T}} = (G, \Pi, S)$ such that: G is *Subtype Graph*, $\widehat{\Pi}^{\mathcal{A},G}$ is a *Minimal Type Pattern Base* for $\mathcal{A}$ under G, and S is a set of *statistics* about the elements of G and Π.



Patterns

```
          (E, Q, F) (A, Q, D)
(C, P, A) (C, Q, A) (B, Q, D)
(C, P, F) (C, Q, F) (B, Q, A)
(A, P, A) (E, Q, D) (B, Q, F)
(A, P, F) (C, Q, D) (A, Q, A) (B, R, T)
(A, P, D) (E, Q, A) (A, Q, F) (A, R, T)
```

```
(C, P, D) (E, Q, D) (C, Q, D) (E, R, T)
                              (C, R, T)
```

Minimal Type Pattern Base

Fig. 1. A small graph representing a data set and the corresponding patterns.

Statistics describe the occurrences of types, properties and patterns. They show how many instances have C as minimal type, how many relational assertions use a property P and how many instances that have C as minimal type are linked to instances that have D as minimal type by a property P.

3 Summary Extraction

Our summarization process, depicted in Figure 2, takes in input an assertion set $\mathcal{A}$ and a terminology $\mathcal{T}$ and produces a summary $\Sigma^{\mathcal{A},\mathcal{T}}$. First, the typing assertion set $\mathcal{A}^C$ is isolated from the relational assertion set $\mathcal{A}^P$, while the subtype graph G is extracted from $\mathcal{T}$. Then, $\mathcal{A}^C$ is processed and the set of minimal types for each named individual is computed. Finally, $\mathcal{A}^P$ is processed in order to compute the minimal type patterns that will form the minimal pattern base $\widehat{\Pi}^{\mathcal{A},G}$. During each phase we keep track of the occurrence of types, properties and patterns, which will be included as statistics in the summary.

Subtype graph extraction. The subtype graph G^C is extracted by traversing all the subtype relations in $\mathcal{T}$. The subtype graph will be further enriched with types from external ontologies asserted in $\mathcal{A}^C$ while we compute minimal types of named individuals (i.e., *external* types). The subtype graph does not include equivalence relations.

For what concerning equivalence relations, even if it is possible to convert them in subtype relations, still preserving the partial ordering of the asserted type, we do not follow such approach because this can lead to counterintuitive or undesired inferences. For example, the `Village`, `PopulatedPlace` and `Place` types from the well known DBPedia ontology are equivalent to the type `Wikidata:Q532` (i.e., village). By including those equivalence relations in G^C, a correct but undesired inference could be that every `Place` is also a `Village` or that `Village`, `PopulatedPlace` and `Place` are equivalent. This real world example highlights an issue related to equivalence relations. Equivalence relations are often used to map named classes from different ontologies (e.g., by leveraging

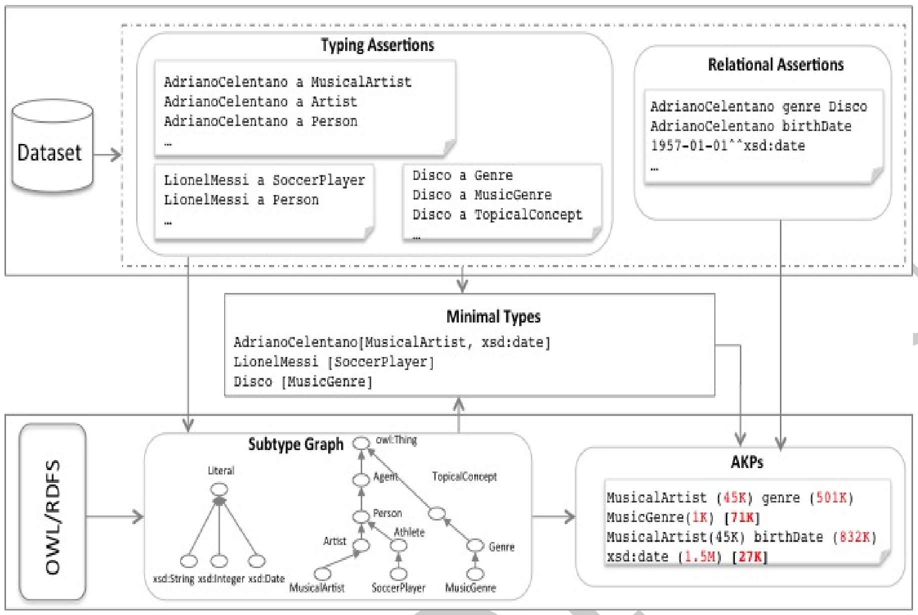


Fig. 2. The summarization workflow.

ontology alignment techniques). The existence of mappings leading to counterintuitive entailments can be explained by precise data management strategies. In the the above mentioned equivalence relation, they may have been added to the ontology in order to map `Village` to `Wikidata:Q532` (i.e., villages with villages). For this reason we do not consider equivalence relations in the extraction of G. As a result, the extracted subtype graph G is complete not with respect to the whole terminology $\mathcal{T}$, but with respect to $\mathcal{T}$ without the equivalence relations.

Minimal types computation. For each instance x, we compute the set M_x of minimal types with respect to the subtype graph G^C. Given x we select all typing assertions $C(x) \in \mathcal{A}^C$ and form the set $\mathcal{A}^C_x$ of typing assertions about x. Algorithm 1 presents the pseudocode for computing M_x. We first initialize M_x with the type `owl:Thing` (line 1), then we iteratively process all the type assertions. At each iteration we select a type C and remove from M_x all the supertypes of C according to G^C (lines 6-10). Then, if M_x does not contain any subtype of C according to G^C we can add C to M_x (lines 11-14). Notice that one preliminary step of the algorithm is to include C in G^C if it was not included during the subtype graph extraction phase (lines 3-5). Consequently, if a type C is not defined in the input terminology, is automatically considered as a minimal type for all the instances x. This approach allows us to handle instances of types from ontologies not included in the input terminology. At the moment, we do

Algorithm 1: Computation of the minimal types set for an instance x.

```
Input: $\mathcal{A}_x^C$ type assertions about the entity $x$, $G^C$ subtype graph
Output: the set $M_x$ the minimal types of $x$
$M_x = \{$owl:Thing$\}$;
for $C(x) \in \mathcal{A}_x^C$ do
    if $C \notin G^C$ then
        $G^C = G^C \cup C$
    end
    $Sup = \text{findSuperTypes}(C, M, G^C)$;
    if $Sup \neq \emptyset$ then
        $M_x = M_x \setminus Sup$;
        $M_x = M_x \cup \{C\}$
    end
    $Sub = \text{findSubTypes}(C, M, G^C)$;
    if $Sub = \emptyset$ then
        $M_x = M_x \cup \{C\}$
    end
end
return $M_x$;
```

not retrieve such ontologies, but the monotonicity of the minimal type pattern base with respect to the terminology graph G ensures that once they are added to the $\mathcal{T}$, the size of the summary will not increase.

Minimal type pattern base computation. For each relational assertion $P(x, y) \in \mathcal{A}^P$, we get the minimal types sets M_x and M_y. For all $C, D \in M_x, M_y$ we add a pattern (C, P, D) to the minimal types pattern base. If y is a literal value we consider its explicit type if present, `rdfs:Literal` otherwise. In this phase the subproperty graph is enriched with properties that are not defined by the terminology, but still occur in at least one pattern (i.e., *external* properties).

Summary storing and presentation. Once extracted, a summary $\Sigma^{\mathcal{A},\mathcal{T}}$ is stored, indexed and made accessible through two user interfaces, i.e., ABSTATBrowse and ABSTATSearch, and a SPARQL endpoint. SPARQL based access and ABSTATBrowse [3] are described in our previous demo paper [12] and thus is left out of the scope of this paper. ABSTATSearch[4], is a novel interface that implements a full-text search functionality over a set of summaries. Types, properties and patterns are represented by means of their local names (e.g., `Person`, `birthPlace` or `Person birthPlace Country`) and conveniently tokenized, stemmed and indexed to support full text queries. Since the patterns are represented as triples, the study of principled and specialized matching and ranking techniques is an interesting extension that we leave for future work.

4 Experimental Evaluation

We evaluate our summaries from different, orthogonal perspectives. We measure the *compactness* of ABSTAT summaries and compare the number of their pat-

[3] `http://abstat.disco.unimib.it/browse,http://abstat.disco.unimib.it/sparql`

[4] `http://abstat.disco.unimib.it/search`

Table 1. Data sets and summaries statistics.

	Relational	**Typing**	**Assertions**	**Types (Ext.)**	**Properties (Ext.)**	**Patterns**
db2014-core	∼ 40.5M	∼ 29.7M	∼ 70.1M	869 (85)	1439 (15)	**171340**
db3.9-infobx	∼ 96.3M	∼ 19.7M	∼ 116.4M	821 (58)	62572 (14)	**732418**
lb	∼ 180.1M	∼ 39.6M	∼ 221.7M	21 (9)	33 (0)	**161**

terns to the number of patterns extracted by Loupe [11], an approach similar to ours that does not use minimalization. The *informativeness* of our summaries are evaluated with two experiments. In the first experiment we show that our summaries provide useful insights about the semantics of properties, based on their usage within a data set. In the second experiment, we conduct a preliminary user study to evaluate if the exploration of the summaries can help users in query formulation tasks. In our evaluation we use the summaries extracted from three linked data sets: DBpedia Core 2014 (**db2014-core**)[5], DBpedia 3.9 (**db3.9-infobox**)[6] and Linked Brainz (**lb**). **db2014-core** and **db3.9-infobox** data sets are based on the DBpedia ontology while the **lb** data set is based on the Music Ontology. DBpedia and LinkedBrainz have complementary features and contain real and large data. For this reason they have been used, for example, in the evaluation of QA systems [10].

4.1 Compactness

Table 1 provides a quantitative overview of data sets and their summaries. To evaluate compactness of a summary we measure the *reduction rate*, defined as the ratio between the number of patterns in a summary and the number of assertions from which the summary has been extracted.

Our model achieves a *reduction rate* of ∼0.002 for **db2014-core**, ∼0.006 for **db3.9-infobox**, and $\sim 6.72 \times 10^{-7}$ for **lb**. Comparing the reduction rate obtained by our model with the one obtained by Loupe (∼0.01 for DBpedia and $\sim 7.1 \times 10^{-7}$ for Linked Brainz) we observe that the summaries computed by our model are more compact, as we only include minimal type patterns. Loupe instead, does not apply any minimalization technique thus its summaries are less compact. The effect of minimalization is more observable on DBpedia data sets, since the DBpedia terminology specifies a richer subtype graph and has more typing assertions. We observe also that 85 external types were added to the **db2014-core** subtype graph and 58 to **db3.9-infobox** subtype graph during the minimal types computation phase as they were not part of the original terminology, and thus are considered by default as minimal types.

[5] The DBpedia 2014 version with mapping based property only
[6] The DBpedia Core 3.9 version plus automatically extracted properties

Table 2. Total number of properties with unspecified domain and range in each data set.

	Domain (%)	Range (%)	Domain-Range (%)
db2014-core	259 (∼18%)	187 (∼13%)	48 (∼3.3%)
db3.9-infobox	61368 (∼98%)	61309 (∼98%)	61161 (∼97%)
lb	13 (∼39%)	15 (∼45%)	13 (∼39%)

4.2 Informativeness

Insights about the semantics of the properties. Our summaries convey valuable information on the semantics of properties for which the terminology does not provide any domain and/or range restrictions. Table 2 provides an overview of the total number of unspecified properties from the data sets. For example, around 18% of properties from **db2014-core** data set have no domain restrictions while 13% have no range restrictions. Observe that this data set is the most curated subset of DBpedia as it includes only triples generated by user validated mappings to Wikipedia templates. In contrast for **db3.9-infobox** data set which includes also triples generated by information extraction algorithms, most of the properties (i.e., the ones from the `dbpepdia.org/property` namespace) are not specified within the terminology.

In general, underspecification may be the result of precise modelling choices, e.g., the property `dc:date` from the **lb** data set. This property is intentionally not specified in order to favor its reuse, being the Dublin Core Elements (i.e., `dc`) a general purpose vocabulary. Another example is the `dbo:timeInSpace` property from the **db2014-core** data set, whose domain is not specified in the corresponding terminology. However, this property is used in a specific way as demonstrated by patterns (`dbo:Astronaut`, `dbo:timeInSpace`, `xsd:double`) and (`dbo:SpaceShuttle` `dbo:timeInSpace`, `xsd:double`). Gaining such understanding of the semantics of the `dbo:timeInSpace` property by looking only at the terminology axioms is not possible.

We can push our analysis further to a more fine grained level. Figure 3 provides an overview of the number of different minimal types that constitute the domain and range of unspecified properties extracted from the summary of the **db2014-core** data set. The left part of the plot shows those properties whose semantics is less "clear", in the sense that their domain and range cover a higher number of different minimal types e.g., the `dbo:type` property. Surprisingly, the `dbo:religion` property is among them: its semantics is not as clear as one might think, as its range covers 54 disparate minimal types, such as `dbo:Organization`, `dbo:Sport` or `dbo:EthnicGroup`. Conversely, the property `dbo:variantOf`, whose semantics is intuitively harder to guess, is used within the data set with a very specific meaning, as its domain and range covers only 2 minimal types: `dbo:Automobile` and `dbo:Colour`.

Small-scale user study. Formulating SPARQL queries is a task that requires prior knowledge about the data set. ABSTAT could support users that lack

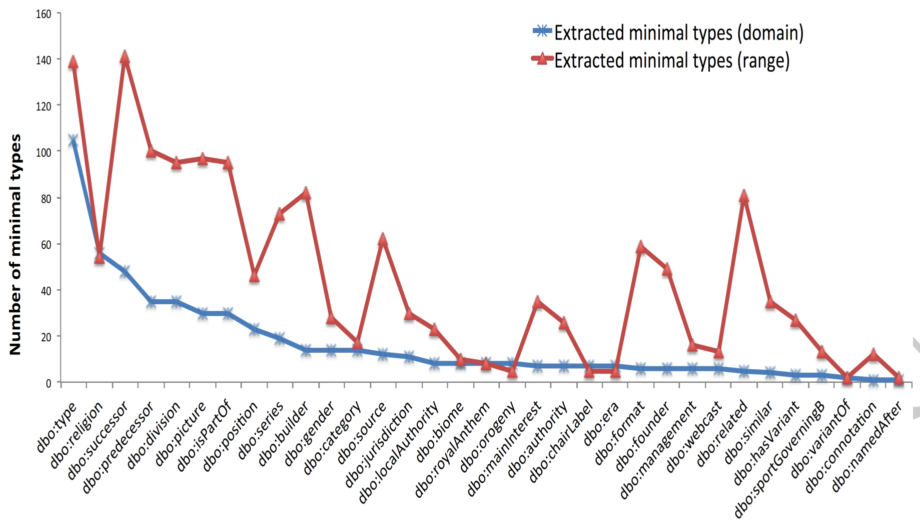


Fig. 3. Distribution of the number of minimal types from the domain and range extracted for not specified properties of the **db2014-core** data set.

such knowledge by providing valuable information about the content of the data set. We designed a user study based on the assignment of cognitive tasks related to query formulation. We selected a set of queries from the *Questions and Answering in Linked Open Data* benchmark[7] [20] to the **db3.9-infobox** data set. The selected queries were taken from logs of the PowerAqua QA system and are believed to be representative of realistic information needs [10], although we cannot guarantee that they cover every possible information need. We provided the participants the query in natural language and a "template" of the corresponding SPARQL query, with spaces intentionally left blank for properties and/or concepts. For example, given the natural language specification *Give me all people that were born in Vienna and died in Berlin*, we asked participants to fill in the blank spaces:

`SELECT DISTINCT ?uri WHERE { ?uri ... <Vienna> . ?uri ... <Berlin> . }`

We selected five queries of increasing length, defined in terms of the number of triple patterns within the `WHERE` clause; one query of length one, two of length two and two of length three. Intuitively, the higher the query length, the more difficult it is to be completed.

We could use a limited number of queries because the tasks are time-consuming and fatigue-bias should be reduced [14]. Overall 20 participants with no prior knowledge about the ABSTAT framework were selected and split into 2 groups: **abstat** and **control**. We profiled all the participants in terms of knowledge about SPARQL, data modelling, DBpedia dataset and ontology, so as to

[7] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

Table 3. Results of the user study.

Group	**Avg. Completion Time (s)**	**Accuracy**
	query 1 - *How many employees does Google have?* - length 1	
abstat	**358.9**	**0.9**
control	380.6	0.8
	query 2 - *Give me all people that were born in Vienna and died in Berlin* - length 2	
abstat	356.3	**1**
control	**346.9**	0.8
	query 3 - *Which professional surfers were born in Australia?* - length 2	
abstat	476.6	0.6
control	**234.24**	**0.7**
	query 4 - *In which films directed by Gary Marshall was Julia Roberts starring?* - length 3	
abstat	**333.4**	0.9
control	445.6	0.9
	query 5 - *Give me all books by William Goldman with more than 300 pages* - length 3	
abstat	**233.4**	**1**
control	569.8	0.7

create two homogeneous groups. We trained for about 20 minutes on how to use ABSTAT only the participants from the first group.

Both groups execute SPARQL queries against the **db3.9-infobox** data set through the same interface and were asked to submit the results they considered correct for each query. We measured the time spent to complete each query and the correcteness of the answers. The correcteness of the answers is calculated as the ratio between the number of correct answers to the given query agains the total number of answers. Table 3 provides the results of the performance of the users on the query completion task[8]. The time needed to perform the 5 queries from all partecipiants in average is 38.6m, while the minimum and the maximum time is 18.4m and 59.2m respectively. The independent *t-test*, showed that the time needed to correctly answer Q5, the most difficult query, was statistically significant for two groups. There was a significant effect between two groups, t(16) = 10.32, $p < .005$, with mean time for answering correctly to Q5 being significantly higher (+336s) for the control group than for abstat group. Using 5 queries is coherent with other related work which suggest that the user study would have 20-60 participants, who are given 10-30 minutes of training, followed by all participants doing the same 2-20 tasks, during a 1-3 hour session [14].

Observe that the two used strategies to answer the queries by participants from the **control** group were: to directly access the public web page describing the DBpedia named individuals mentioned in the query and very few of them submitted explorative SPARQL queries to the endpoint. Most of the users searched on Google for some entity in the query, then consulted DBpedia web pages to find the correct answer. DBpedia is arguably the best search-

[8] The raw data can be found at `http://abstat.disco.unimib.it/downloads/user-study`

able dataset, which is why this explorative approach was successful for relatively simple queries. However, this explorative approach does not work with other non-indexed datasets (e.g., LinkedBrainz) and for complex queries. Instead, participants of the **abstat** group took advantage of the summary, obtaining huge benefits in terms of average completion time, accuracy, or both. Moreover, they achieved increasing accuracy over queries at increasing difficulty, still performing the tasks faster. We interpret the latter trend as a classical cognitive pattern, as the participants became more familiar with ABSTATBrowse and ABSTAT-Search web interfaces.

The noticeable exception is query 3. In particular, participants from the **abstat** group completed the query in about twice the time of participants from **control** group. This is due to the fact that the individual `Surfing` (which is used as object of the property `dbo:occupation`) is classified with no type other than `owl:Thing`. As a consequence, participants from the **abstat** group went trough a more time consuming trial and error process in order to guess the right type and property. Participants from the **abstat** group finally came to the right answer, but after a longer time. This issue might be solved by applying state-of-the-art approaches for type inference on source RDF data [13] and suggest possible improvements of ABSTAT for example including values for concepts that are defined by closed and relatively small instance sets.

5 Related Work

Different approaches have been proposed for schema and data summarization. Most of them identify pieces of knowledge that are more relevant to the user, while the others do not represent the relations among instances but are limited in presenting the co-occurence of the most frequent types and properties. We compare our work to approaches explicitly proposed to summarize Linked Data and ontologies, and to extract statistics about the data set.

A first body of work has focused on summarization models aimed at identifying subsets of data sets or ontologies that are considered to be more relevant. Authors in [22] rank the axioms of an ontology based on their salience to present to the user a view about the ontology. RDF Digest [19] identifies the most salient subset of a knowledge base including the distribution of instances in order to efficiently create summaries. Differently from these approaches, ours aims at providing a complete summary with respect to the data set.

A second body of work has focused on approaches to describe linked data sets by reporting statistics about the usage of the vocabulary in the data. The most similar approach to ABSTAT is Loupe [11], a framework to summarize and inspect Linked Data sets. Loupe extracts types, properties and namespaces, along with a rich set of statistics. Similarly to ABSTAT, Loupe offers a triple inspection functionality, which provides information about *triple patterns* that appear in the data set and their frequency. Triple patterns have the form <*subjectType*, *property*, *objectType*> and are equivalent to our patterns. However, Loupe does not apply any mimimalization technique: as shown in Section 4.1, summaries computed by our model are significatively more compact.

In [4], authors propose a graph-based approach called ELIS for visualising and exploring induced schema for Linked Open Data. Similarly as in ABSTAT, ELIS also extract patterns, called `schema-level patterns` as a combination between a set of subject types and a set of object types. These subject sets and object sets can be seen as nodes connected by a property. Differently from ABSTAT, ELIS allows users to visualise the induced schema. In ABSTAT, we do not aim at visualising the induced schema, but providing a compact summary which users can browse and search for a pattern, a concept or a property. The summary provided by ABSTAT only includes minimal types, while ELIS does not apply any minimalization technique.

In [2], authors consider vocabulary usage in the summarization process of an RDF graph and use information similar to patterns. A similar approach is also used in MashQL [5], a system proposed to query graph-based data (e.g., RDF) without prior knowledge about the structure of a data set. Our model excludes several redundant patterns from the summary through minimalization, thus producing more compact summaries. Knowledge pattern extraction from RDF data is also discussed in [16], but in the context of domain specific experiments and not with the purpose of defining a general linked data summarization framework. Our summarization model can be applied to any data set that uses a reference ontology and focuses on the representation of the summary.

Other approaches proposed to describe data sets do not extract connections between types but provide several statistics. SchemeEx extracts interesting theoretic measures for large data sets, by considering the co-occurrence of types and properties [8]. A data analysis approach on RDF data based on an warehouse-style analytic is proposed in [3]. This approach focuses on the efficiency of processing analytical queries which poses additional challenges due to their special characteristics such as complexity, evaluated on typically very large data sets, and long runtime. However, this approach differently from ours requires the design of a data warehouse specially for a graph-structured RDF data. Linked Open Vocabularies[9], RDFStats [9] and LODStats [1] provide several statistics about the usage of vocabularies, types and properties but they do not represent the connections between types.

The approach in [21] induces a schema from data and their axioms represent stronger patterns compared to the patterns extracted by our approach. ABSTAT aims to represent every possible connections existing among types while EL axioms aims to mine stronger constraints.

The authors in [6] have a goal even more different than ours. They provide lossless compression of RDF data using inference obtaining thus a reduction rate of 0.5 in best cases. Our approach loses information about instances because aims at representing schema-level patterns, but achieves a reduction rate of 0.002.

[9] `http://lov.okfn.org/`

6 Conclusion and Future Work

Getting an understanding of the shape and nature of the data from large Linked Data sets is a complex and a challenging task. In this paper, we proposed a minimalization-based summarization model to support data set understanding. Based on the experimentation we show that our summarization framework is able to provide both *compact* and *informative* summaries for a given data set. We showed that using ABSTAT framework the summaries are more compact than the ones generated from other models and they also help the user to gain insights about the semantics of underspecified properties in the ontology. The results of our preliminary experiment showed that ABSTAT help users formulating SPARQL queries both in terms of time and accuracy.

We plan to run the experiment in large scale, thus including more users with different background characteristics in order to analyse in details which is the target group of users for which ABSTAT is more useful. Several are the future research directions. We plan to complement our coverage-oriented approach with relevance-oriented summarization methods based on connectivity analysis. Another interesting direction was highlighted by our user study, that is the inference of specific types for untyped instances found in the data set. We are also planning to consider the inheritance of properties to produce even more compact summaries. Finally, we envision a complete analysis of the most important data set available in the LOD cloud.

References

1. S. Auer, J. Demter, M. Martin, and J. Lehmann. LODStats - An Extensible Framework for High-Performance Dataset Analytics. In *EKAW (2)*, 2012.
2. S. Campinas, T. E. Perry, D. Ceccarelli, R. Delbru, and G. Tummarello. Introducing RDF Graph Summary with Application to Assisted SPARQL Formulation. In *DEXA*, 2012.
3. D. Colazzo, F. Goasdoué, I. Manolescu, and A. Roatiş. RDF Analytics: Lenses over Semantic Graphs. In *WWW*, 2014.
4. T. Gottron, M. Knauf, A. Scherp, and J. Schaible. ELLIS: interactive exploration of linked data on the level of induced schema patterns. In *Proceedings of the 2nd International Workshop on Summarizing and Presenting Entities and Ontologies (SumPre 2016) co-located with the 13th Extended Semantic Web Conference (ESWC 2016), Anissaras, Greece, May 30, 2016.*, 2016.
5. M. Jarrar and M. Dikaiakos. A Query Formulation Language for the Data Web. *IEEE Trans. Knowl. Data Eng*, 24(5), 2012.
6. A. K. Joshi, P. Hitzler, and G. Dong. Logical linked data compression. In *The Semantic Web: Semantics and Big Data*, pages 170–184. Springer, 2013.
7. S. Khatchadourian and M. P. Consens. ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud. In *ESWC (2)*, 2010.
8. M. Konrath, T. Gottron, S. Staab, and A. Scherp. SchemEX - Efficient construction of a data catalogue by stream-based indexing of linked data. *J. Web Sem.*, 16, 2012.
9. A. Langegger and W. Wöß. RDFStats - An Extensible RDF Statistics Generator and Library. In *DEXA*, 2009.

10. V. Lopez, C. Unger, P. Cimiano, and E. Motta. Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13, 2013.
11. N. Mihindukulasooriya, M. Poveda Villalon, R. Garcia-Castro, and A. Gomez-Perez. Loupe - An Online Tool for Inspecting Datasets in the Linked Data Cloud. In *ISWC Posters & Demonstrations*, 2015.
12. M. Palmonari, A. Rula, R. Porrini, A. Maurino, B. Spahiu, and V. Ferme. ABSTAT: Linked Data Summaries with ABstraction and STATistics. In *ESWC Posters & Demonstrations*, 2015.
13. H. Paulheim and C. Bizer. Type Inference on Noisy RDF Data. In *ISWC*, 2013.
14. A. Perer and B. Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceedings of the SIGCHI conference o Human Factors in computing systems*, pages 265–274. ACM, 2008.
15. S. Peroni, E. Motta, and M. d'Aquin. Identifying Key Concepts in an Ontology, through the Integration of Cognitive Principles with Statistical and Topological Measures. In *ASWC*, 2008.
16. V. Presutti, L. Aroyo, A. Adamou, B. A. C. Schopman, A. Gangemi, and G. Schreiber. Extracting Core Knowledge from Linked Data. In *COLD2011*, 2011.
17. M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the Linked Data Best Practices in Different Topical Domains. In *ISWC*, 2014.
18. S. Staab and R. Studer. *Handbook on ontologies*. Springer Science & Business Media, 2010.
19. G. Troullinou, H. Kondylakis, E. Daskalaki, and D. Plexousakis. RDF Digest: Efficient Summarization of RDF/S KBs. In *ESWC*, 2015.
20. C. Unger, C. Forascu, V. Lopez, A. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question Answering over Linked Data (QALD-4). In *CLEF*, 2014.
21. J. Völker and M. Niepert. Statistical schema induction. In *The Semantic Web: Research and Applications*, pages 124–138. Springer, 2011.
22. X. Zhang, G. Cheng, and Y. Qu. Ontology summarization based on rdf sentence graph. In *WWW*, 2007.