ABSTAT: Linked Data Summaries with ABstraction and STATistics

Matteo Palmonari[1(✉)], Anisa Rula[1], Riccardo Porrini[1], Andrea Maurino[1], Blerina Spahiu[1], and Vincenzo Ferme[2]

[1] University of Milano-Bicocca, Milan, Italy
{palmonari,rula,porrini,maurino,spahiu}@disco.unimib.it
[2] University of Lugano (USI), Lugano, Switzerland
vincenzo.ferme@usi.ch

Abstract. While much work has focused on continuously publishing Linked Open Data, little work considers how to help consumers to better understand existing datasets. ABSTAT framework aims at providing a better understanding of big and complex datasets by extracting summaries of linked data sets based on an ontology-driven data abstraction model. Our ABSTAT framework takes as input a data set and an ontology and returns an ontology-driven data summary as output. The summary is exported into RDF and then made accessible through a SPARQL endpoint and a web interface to support the navigation.

1 Introduction

As of April 2014 up to 1014 data sets have been published in the Linked Open Data (LOD) cloud, a number that is constantly increasing[1]. However, for a user interested in using available LOD, it is not easy to understand to what extent a data set covers a domain of interest, how the knowledge is structured in the data set, for example, to compare two data sets based on their content. To this end, a data consumer should be able to answer to questions such as: what types of resources are described in the 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 these questions have several consequences for data consumption.

Linked data sets make use of ontologies to describe the semantics of their data. However, answering the above questions by only looking at ontolologies is not easy since ontologies may be large. For example, DBpedia uses a vocabulary consisting of 705 (local) concepts and 2,794 properties. Ontologies may be underspecified to model a large amount of diverse data in a flexible way. For example, either the domain or the range is unspecified for 585 properties of the DBpedia Ontology. In addition, in relatively expressive ontologies like the Music Ontology, some connections between types may be specified by means of OWL

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

F. Gandon et al. (Eds.): ESWC 2015, LNCS 9341, pp. 128–132, 2015.
DOI: 10.1007/978-3-319-25639-9_25

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 occur in the data set. Answers to the above questions can be collected with explorative queries, but at the price of a significant server overload and high response times.

In this paper we describe ABSTAT, a linked data summarization framework based on an ontology-based ABstraction model and on the computation of STATistics. ABSTAT extracts summaries that are represented in RDF and can be queried or navigated through a web interface, which demonstrates the contribution of our framework to answer questions like the ones listed at the beginning of this section. Although a summary contains several elements, as explained in the next section, the key feature of our approach is the extraction of Abstract Knowledge Patterns (AKPs) from the data with the help of the data ontology (we consider OWL and RDFS ontologies).

In our approach, a *type* is an ontology concept or a datatype. Observe that, differently from a literal, an ontology instance may have several types specified in the data, and most of them can be related by a subconcept relation. AKPs are triples having the form $<subjectType, pred, objectType>$, which represent the occurrence of triples $<sub, pred, obj>$ in the data, such that *subjectType* is a minimal type of *sub* and *objectType* is a minimal type of *obj*. By considering minimal types of resources, computed with the help of the data ontology, we exclude several redundant AKPs from the summary.

Our summarization framework introduces several novel contributions to mitigate the data set understanding problem. The AKPs that are extracted and represented with ABSTAT provide a richer characterisation of a data set, if compared to popular approaches based on metadata (e.g., using the DCAT vocabulary). Other approaches like Linked Open Vocabularies[2] and LODStats provide several statistics about the usage of vocabularies, types and properties [1] but they do not represent the connections between types. SchemeEx extracts interesting information for large data sets, by considering the co-occurrence of types and properties [3], but does not extract connections between types and does not represent the extracted information as a summary in RDF. Knowledge patterns have been extracted from RDF data in a previous work [4], but in the context of domain specific experiments and not with the purpose of defining a general linked data summarisation framework. ABSTAT's model can be applied to any data set that use a reference ontology and focuses on the representation of the summary. Another approach focuses on ranking patterns in the ontology (e.g., a subclass relation, or a property range restriction) based on their salience so as to present to the user a subset of significant axioms of the ontology [5], while we aim to provide an abstract but complete view of the data set. Another approach adopts bisimulation and contraction to find common patterns in data [2]. Their summaries are very different from the ones obtained with our abstraction model.

[2] http://lov.okfn.org/.

2 The ABSTAT Framework

Our ABSTAT framework takes as input a data set and an ontology (used by the data set) and returns an ontology-driven data summary. The summary is exported in RDF to support query and navigation.

A Linked Data Summary of a data set Δ that uses an ontology V consists of five components: (1) a Concept Graph, (2) a set of Abstract Knowledge Patterns (AKPs), (3) a set of Co-occurrence Patterns (CP), (4) a set of *Types* (concepts and datatypes) and *Properties*, and (5) a set *Stat* of functions that describe the occurrence of the other components in the data. AKPs are defined as in Sect. 1. CPs represent the co-occurrence of predicates and types and have the structure <*SubjectType*,*Property*> and <*Property*,*ObjectType*>. We explain the Concept Graph, the methods to extract the components, and their meaning here below.

Concept Graph. The Concept Graph represents the subconcept relation between concept types. It is extracted from V by traversing its subclass relation and considers also the ontologies directly imported by V (e.g., the Music Ontology, used in LinkedBrainz, imports other 7 ontologies). Every ontology concept that has no explicit superconcept is considered subconcept of `owl:Thing`.

AKPs and CPs. The dataset is partitioned in two sets of triples: one partition contains all the *typing statements*, i.e., all the triples where the predicate is `rdf:type`; the other partition contains *relation statements*, i.e., every other triple. We process all the typing statements to collect, for every instance x that occurs in some typing statement, its set of *minimal* types. This is done by traversing the subtype relation in the Concept Graph. An instance x may have one or more minimal types such that none of the minimal type is subtype of one another; although only one *minimum* type is often found. Then we scan every relation statement of the dataset (only once). For each triple <*sub*,*pred*,*obj*> we extract all the minimal types of *sub* and *obj*, denoted respectively by μ^{sub} and μ^{obj}, as follows. If x is a resource (in the subject or object position) we set μ^x as the set of minimal types found after scanning the typing statements; if x is untyped and no typing information was found, we define $\mu^x = \{$`owl:Thing`$\}$. If x is a typed literal, the minimal (and minimum) type is defined by its datatype. If x is an untyped literal, its minimal type is `rdfs:Literal`. An AKP $< t^{sub}, pred, t^{obj} >$ occurs in a triple <*sub*,*pred*,*obj*> iff $t^{sub} \in \mu^{sub}$ and $t^{obj} \in \mu^{obj}$. If an AKP occurring in a triple is present in the summary already, then we update the occurrence of the AKP. Else we add the AKP to the summary. When we scan a triple, we apply the above described method to extract CPs or update their occurrence.

Types and Properties. The set of Types and Properties of a dataset is defined by considering all the types, ontology concepts and datatypes, which occur in some AKP or in the ontology. Their occurrence stats are defined using the minimal type semantics used for AKPs.

To represent summaries in RDF we designed the *ld-summaries* ontology in OWL. We model AKPs and CPs through reified statements. Properties and types

are represented with the `lds:Property`, `skos:Concept` and `lds:Datatype` concepts, respectively. The concept graph is represented by using the `skos:broader` property that connects the types. We represent URIs of properties and types by using the summary base (i.e., their local id in the summary) and link them to their global id from the source ontology with the `rdfs:seeAlso` property. This representation enables the linking between properties and types from both an intra-summary and inter-summary perspective. The data property `lds:occurrence` associates the occurrence statistic to AKPs, CPs, properties and types.

The extraction of the concept graph is implemented in JAVA and uses Jena to manipulate the ontologies. The rest of the summarisation framework is implemented in AWK[3], a scripting language specific for text processing and efficient when a large amount of triples have to be processed.

3 Demonstration

We now describe a practical use case of the ABSTAT framework. We showcase the benefits of using the ABSTAT framework for dataset understanding by means of a Web application (http://abstat.disco.unimib.it:8880), which supports the interactive visualization of the summaries. We store the summaries in a Virtuoso triple store instance so as to support querying and navigation.

Mike is a young entrepreneur who wants to create *MyMusicNow*, a new mobile app with music data. He wants to exploit one data set from the LOD cloud to semantically link news coming from twitter accounts of music artists. To realize his app Mike finds two datasets: DBpedia[4] and LinkedBrainz[5]. Mike does not know the semantic structure of the two datasets and their coverage in terms of instances. Hence, Mike uses ABSTAT to analyze the data sets and find answers to the following queries: (1) How are music artists/groups/songs described? (2) How many instances are covered? (3) Is there any incongruence in the data?

Figure 1 shows two screenshots of the main page of ABSTAT. After selecting the LinkedBrainz dataset, Mike visualizes the AKPs and their occurrences. He can either navigate the summary or filter out the AKPs by subject and/or object types, and/or by property and look at their occurrences. In the filter area, a self filling feature supports him in identifying the types of interests (e.g., `mo:SoloMusicArtist`, `mo:MusicArtist`, `dbo:MusicalArtist` or `dbo:Band`).

Mike easily finds that LinkedBrainz contains 237,464 `mo:MusicArtist`, while DBpedia contains 105,420 `dbo:MusicalArtist`, i.e., 60 % less than LinkedBrainz. On the other hand, musical artists are described in DBpedia with a richer and more diverse set of properties than in LinkedBrainz, as shown in Fig. 1. Mike also discovers potential errors in the two datasets. There is only one occurrence of the AKP <`mo:MusicArtist`,`mo:member_of`,`mo:MusicArtist`> in LinkedBrainz, which may depend on a mistyped instance. Several counterintuitive AKPs, such

[3] http://awk.info/.
[4] http://dbpedia.org.
[5] http://linkedbrainz.org/.

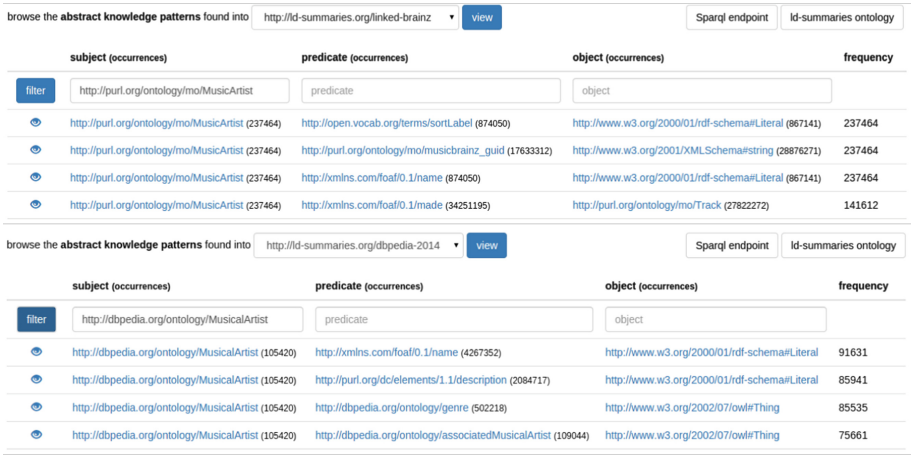

Fig. 1. The ABSTAT web interface.

as <dbo:Band,dbo:genre,dbo:Band> and <dbo:Band,dbo:instrument,dbo:Band>, which may reveal incorrect data, occur quite frequently also in DBpedia.

In order to gain a comparable level of understanding of the two datasets without the help of ABSTAT, Mike should have (1) manually inspected the two ontologies and understood their schemas, and (2) written many SPARQL queries to get the statistics about their number of instances. Getting answers for these queries may require a lot of time and even lead to timeouts in server responses. Finally, Mike could have hardly spotted the incongruences higlighted by ABSTAT, with the risk of developing unreliable services for his app.

Acknowledgements. This research has been supported in part by FP7/2013-2015 COMSODE (under contract number FP7-ICT-611358).

References

1. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats – an extensible framework for high-performance dataset analytics. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 353–362. Springer, Heidelberg (2012)
2. Khatchadourian, S., Consens, M.P.: ExpLOD: summary-based exploration of interlinking and RDF usage in the linked open data cloud. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 272–287. Springer, Heidelberg (2010)
3. Konrath, M., Gottron, T., Staab, S., Scherp, A.: SchemEX - efficient construction of a data catalogue by stream-based indexing of linked data. J. Web Sem. **16**, 52–58 (2012)
4. Presutti, V., Aroyo, L., Adamou, A., Schopman, B.A.C., Gangemi, A., Schreiber, G.: Extracting core knowledge from linked data. In: COLD (2011)
5. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on RDF sentence graph. In: WWW, pp. 707–716 (2007)