Profiling and Mining RDF Data with ProLOD++

Ziawasch Abedjan, Toni Gruetze, Anja Jentzsch, Felix Naumann
Hasso Plattner Institute (HPI), Potsdam, Germany
firstname.lastname@hpi.uni-potsdam.de

***Abstract*—Before reaping the benefits of open data to add value to an organizations internal data, such new, external datasets must be analyzed and understood already at the basic level of data types, constraints, value patterns etc. Such *data profiling*, already difficult for large relational data sources, is even more challenging for RDF datasets, the preferred data model for linked open data.**

We present ProLOD++, a novel tool for various profiling and mining tasks to understand and ultimately improve open RDF data. ProLOD++ comprises various traditional data profiling tasks, adapted to the RDF data model. In addition, it features many specific profiling results for open data, such as schema discovery for user-generated attributes, association rule discovery to uncover synonymous predicates, and uniqueness discovery along ontology hierarchies. ProLOD++ is highly efficient, allowing interactive profiling for users interested in exploring the properties and structure of yet unknown datasets.

I. Profiling Linked Open Data

At the time of writing, Linked Open Data (LOD) as compiled in http://linkeddata.org comprised already more than 300 data sources including prominent examples, such as DBpedia, YAGO, and Freebase. A LOD dataset is usually represented in the Resource Description Framework (RDF) embodying an entity-relationship-graph or a set of triplified facts consisting of subjects, predicates, and objects. Most of the datasets are openly available and connected amongst each other via *sameAs* links between representations of same real-world entities. Hundreds more open RDF datasets are listed for instance at http://datahub.io.

However, consuming LOD is not easy, because the sources are heterogeneous, often inconsistent, and lack often even basic metadata. One of the main reasons for this problem is that many of the data sources, such as DBpedia [7] or YAGO [13], have been extracted from unstructured data. Furthermore, a knowledge base usually evolves over time when more facts and entities are added and rigid schema and ontology definitions, hand-crafted at some point of time, lose validity over all entities of the dataset. Hence it is vital to thoroughly examine and understand each dataset, its structure, and its properties before usage.

Manually inspecting datasets can achieve this goal only to a limited extent: algorithms and tools are needed that profile the dataset to retrieve relevant and interesting meta-data analyzing the entire dataset [14]. Indeed, there are many commercial tools, such as IBM's Information Analyzer, Microsoft's SQL Server Integration Services (SSIS), or Informatica's Data Explorer, and some research prototypes, such as [12], for profiling relational datasets. However all of these tool were designed to profile relational data. LOD which is represented in RDF data has a very different nature and calls for specific profiling and mining techniques. Current tools to work on RDF data are limited to graph visualization and editing: LODlive[1] is a browser-based tool to browse and search in RDF datasets. RDF Pro[2] is a suite for visual editing RDF data. LODStats [6] is a stream-based approach for gathering comprehensive statistics about RDF datasets. Finally, RelFinder [10] is a web-based tool to interactively discover relationships between entities on the Web of Data.

To this end we present ProLOD++, the first web-based tool to profile arbitrary LOD datasets. For this purpose we significantly extended our prototype ProLOD (presented at the NTII'10 workshop [8]), which generated basic statistics (value patterns and distributions, predicate frequencies) and some dependencies (positive, negative, and inverse associations of predicates) on a given RDF dataset. For ProLOD++ we add visualized functionality based on recently developed techniques and algorithms:

- ProLOD++ is able to identify predicate combinations that contain only unique value combinations to distinctly identify entities [2]. We adapted our unique discovery system to handle RDF data and show degrees of uniqueness along ontology hierarchies.
- We integrated *mining configurations*, a framework that allows to mine association rules to identify dependencies between subjects, predicates, and objects from different perspectives [3].
- Our system incorporates a module to suggest ontology definition changes to remedy mismatches of data and ontology [1].
- Combining two mining configurations, ProLOD++ discovers synonymously used predicates in the data [5]
- ProLOD++ provides a user interface for manually adding facts, supported by data-driven suggestions for a given entity.
- Finally, ProLOD++ is able to generate presumably missing facts based on a given dataset.

In the following, we first describe the features of ProLOD++ in more detail. In Section III, we describe how we plan to perform the demonstration of our system, and we conclude in Section IV.

II. Profiling and Mining Features

The features of ProLOD++ can be loosely categorized into profiling, mining, or cleansing tasks, as illustrated in Table I. We first describe fundamental functionality to organize the data and generate statistics and basic link and pattern analysis.

[1]http://en.lodlive.it/
[2]http://www.linkeddatatools.com/rdf-pro-semantic-web

Profiling	**Mining**	**Cleansing**
Key analysis	Unsupervised clustering and labeling	Auto-completion
Predicate and value distribution	Association rules on S, P, and O's	Ontology alignment
String pattern analysis	Inverse predicate discovery	Fact generation/amendement
Link analysis	Synonym predicate discovery	
Data type analysis		

TABLE I: Functionalities of ProLOD++

Then we describe more advanced components to discover key-candidates, mining-based approaches to generate meta-data, and to discover inverse or synonymous predicates. Finally, we present mining-based approaches to improve a given dataset.

A. *Clustering and Labeling*

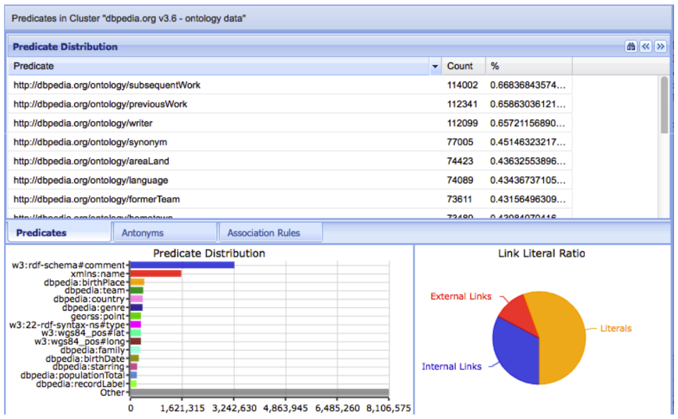
In comparison to other data models (i.e., the relational model), RDF lacks to provide explicit schema information that precisely defines the types of entities and their attributes. Therefore, many datasets provide ontologies that categorize entities and define data types and semantics of properties. However, ontology information for an arbitrary dataset is not always available or may be incomplete.

To understand and organize an RDF dataset it is desirable to generate meta information that can yield a separation of entities into semantically related groups or clusters. Attributes occurring in different semantic groups might have diverse semantics. For instance, the predicate `length` might appear in different domains. It might occur as a measure of distance between start and end of an object, such as a train, a measure of time for the duration of a song title, etc. To solve this shortcoming, ProLOD++ provides an unsupervised hierarchical clustering algorithm [8], which is based on the well-known *K-Means* algorithm. The generated clusters are visualized as a tree with the complete dataset as its root and can be modified at will. In this way, it is possible to perform further analysis only on subsets of the dataset that correspond to clusters on different granularity levels.

To provide the user with hints about the content to be expected within the automatically retrieved clusters, ProLOD++ provides labels for each cluster. The labeling is based on text property values of the clustered entities. The algorithm scores all terms within these texts based on their *tf-idf* values and selects the top-k (thus most relevant) terms. Of course ProLOD++ allows also to cluster entities based on predefined ontologies derived by properties, such as `rdf:typeOf`.

B. *Statistics and Pattern Analysis*

During the initial import of a dataset, ProLOD++ gathers statistics about frequencies and distributions of distinct subjects, predicates, and objects. These distributions are computed online when selecting clusters of the dataset. Figure 1 illustrates a screenshot of a ProLOD++ tab, which lists the predicates of a chosen cluster with their number of occurrences and a chart illustrating the most frequent predicates at the bottom. Furthermore on the lower right corner a pie chart illustrates the range of the predicates and the ratio of internal and external links: red for the ratio of predicates linking to objects from external datasets or namespaces, blue denotes the ratio of predicates linking to entities from the same namespace, and yellow shows the large amount of predicates that have a literals as their object value.


Predicates in Cluster "dbpedia.org v3.6 - ontology data"

Predicate Distribution

Predicate	Count	%
http://dbpedia.org/ontology/subsequentWork	114002	0.66836843574...
http://dbpedia.org/ontology/previousWork	112341	0.65863036121...
http://dbpedia.org/ontology/writer	112099	0.65721156890...
http://dbpedia.org/ontology/synonym	77005	0.45146323217...
http://dbpedia.org/ontology/areaLand	74423	0.43632553896...
http://dbpedia.org/ontology/language	74089	0.43436737105...
http://dbpedia.org/ontology/formerTeam	73611	0.43156496309...

Fig. 1: Predicate distribution visualized by ProLOD++

To further elaborate the meaning and usage of predicates of a dataset, the pattern analysis provides the user with statistics about data types and pattern distributions of the object values of particular predicates. For instance, the predicate `release_date` over several movies might contain date values in different formats (year-month-day, month/day/year, ...), the raw release year, or even empty values. The pattern analysis provides a means to perform a drill-down from automatically determined data types, over pattern expressions, down to the actual object values. All metadata is dynamically produced and appropriately visualized.

C. *Uniqueness Analysis*

The Web of Linked Data is built upon the idea that data items on the Web are connected by RDF links. The reality on the Web shows that Linked Data sources set some RDF links pointing at data items in related data sources, but they clearly do not set RDF links to *all* data sources that provide related data[3]. Writing linkage rules for unknown and large datasets is a time-consuming task. This includes finding main classes occurring in the dataset as well as finding relevant sets of properties that define entities unambiguously.

For instance, describing entities in an unambiguous way is a crucial task for embedding information Linked Data entities into applications or websites. News sites that incorporate infoboxes for named entities can use a comprehensive set of property values to describe a real-world entity to the reader. To this end, ProLOD++ is able to identify predicate combinations that contain only unique values as key-candidates to distinctly identify entities.

[3]http://lod-cloud.net/state/

ProLOD++ uses our new unique discovery technique, DUCC [11], to identify unique predicate combinations on clusters. DUCC provides a scalable and efficient method for finding all unique *and* non-unique column combinations in big datasets by using a novel hybrid graph traversal technique, which traverses the lattice in combination of depth-first and random walk.

Besides the concrete unique predicate combinations, the number of non-NULL values per predicate, the uniqueness of all the property values per predicate, and the number of unique values per predicate. Having these values at hand, the user can determine possible keys based on the unique predicate combinations retrieved by DUCC taking into consideration the ratio of non-NULL values for each of the predicates. For example out of the 185,081 athletes in DBpedia, only 36 have a `dbpedia:espnId` value, yet all of these values are unique. This defines `dbpedia:espnId` as a unique predicate combination for athletes in DBpedia.

Furthermore, ProLOD++ can cluster entities based on a dataset's underlying ontology. This allows the user to see the evolution of key features through hierarchical levels, thus determining class-specific predicates in unique predicate combinations. I.e., the degree of uniqueness of `dbpedia:espnId` can be observed to for things, persons, athletes, and finally football players.

D. Rule-based Analysis

ProLOD provided a simple association rule mining engine that allowed to discover positive and negative association rules among predicates [8]. In ProLOD++, we replace this system by a new framework adapting the mining configuration methodology that generalizes the targets of mining to each part of an RDF statement [3]. The algorithm used for mining is FP-Growth [9]. The engine is able to discover positive association rules as well as negative association rules. In the following we present four applications derived from mining configurations to enhance the usability of RDF data.

1) Ontology re-engineering: "Misuse" of ontology definitions can be traced in part to too specific or too generic ontologies. A mismatch of data and ontology impedes the integration of data sources. Based on an existing ontology, we identify two typical cases where the specification differs from usage patterns: *overspecification* and *underspecification*. A certain class is overspecified, if one or more properties are declared for this class by the ontology, but are rarely (if ever) used for real-world data, e.g., `scottishName` for `Settlement`. A class is underspecified, when in real-world data certain properties are used frequently even though they are not specified by the vocabulary. Underspecification may occur when the class definition lacks certain properties that are commonplace in instance data, e.g., `genre` for `Band`. Further examples are shown in Table II.

ProLOD++ features our rule-based approach that automatically identifies over- and underspecifications and makes suggestions in order to change the ontology definition [1]. With ProLOD++ the user can identify classes that are under- or over-specified. Furthermore, tables with suggestions to remove/add predicates from/to a certain class are presented; where, upon the selection of a suggested predicate, a user can identify the original domain class of the property.

2) Synonym discovery: Another remedy to handle inconsistencies between ontology and data is to identify predicates that substitute each other, such as `starring` and `artist`. These pairs of predicates help to understand the usage of predicates and can be used for query relaxation. ProLOD++ provides an association rule-based approach for discovering such pairs of synonymously used predicates [5]. Typically those predicate pairs have a similar range of object values and do not co-occur for the same subject. Both intuitions can be modelled by mining configurations as proposed in [5]. Table III illustrates the top 5 synonym pairs discovered by applying the synonym discovery on entities of type `Work` or `Organisation`.

	DBpedia `Work`	DBpedia `Organisation`
1.	`artist`, `starring`	`city`, `location`
2.	`artist`, `musicComposer`	`city`, `hometown`
3.	`author`, `writer`	`location`, `hometown`
4.	`creator`, `writer`	`city`, `ground`
5.	`composer`, `musicComposer`	`city`, `locationCity`

TABLE III: Top 5 synonym pairs on DBpedia 3.7 subsets

3) Inverse predicate discovery: Inverse predicates unveil potential redundancies within the dataset. Due to the directed nature of predicates in the RDF data model, it is possible (and sometimes useful) to express the same fact with two triples. Elaborated knowledge bases can express such relationships in the ontology language OWL using `owl:inverseOf`. In ProLOD++, an inverse predicate pair is defined as a pair of predicates $\langle p_1, p_2 \rangle$ that *commonly* occurs in inverted triples, such as $\langle :a, p_1, :b \rangle$ and $\langle :b, p_2, :a \rangle$. Please note, the pair might consist of only one predicate ($p_1 = p_2$). Examples for inverse predicate pairs are $\langle$`before,after`$\rangle$, $\langle$`precededBy,followedBy`$\rangle$, $\langle$`spouse,spouse`$\rangle$, or $\langle$`star,exoplanet`$\rangle$. The inverse predicate discovery algorithm of ProLOD++ identifies and displays all inverse predicates within all triples of a dataset that occur more often than a configurable frequency threshold.

4) Fact generation: ProLOD++ provides two different methods to generate new facts based on a given dataset. First, a user-driven approach that supports the process of manual fact generation, and second, a completely automatic approach to generate missing but correct facts [4].

a) Predicate and object suggestion: A user can add new facts to a given dataset via the ProLOD++ GUI. It supports the user by suggesting predicate as well as object values according to association rules that have been mined from the dataset.

When a user is inserting or editing the facts related to a specific subject, the system is aware of all predicates or objects that have already been inserted for the current subject. To generate a list of suggestions, all rules that incorporate the previously inserted predicates as their antecedents are retrieved. The suggestions then are all those predicates that occurred as consequences of the retrieved rules. The ranking of the suggestions is based on scores that are computed for each suggestion by aggregating all confidence values of the retrieved rules that have the specific predicate suggestion as

Property	Overspecified Class	Suggested Class
`philosophicalSchool`	`Person`	`Philosopher`
`countySeat`	`PopulatedPlace`	`AdministrativeRegion`
`anthem`	`PopulatedPlace`	`Country`
`depth`	`Place`	`Lake`
`numberOfGraduateStudents`	`EducationalInstitution`	`College/University`

TABLE II: Suggestions for overspecified classes

their consequence. Based on the next chosen predicate the suggestion list changes again, because the rules that contain the new predicate as their antecedent are also taken into account.

b) Auto-amendment of new facts: ProLOD++ can automatically *amend* the given dataset with completely new facts, combining association rules between predicates and objects. The intuition behind the amendment approach is that we act within the Open World Assumption and a missing fact for an entity is not necessarily a wrong fact. For example the rule `Wehrmacht` → `World War II` is comprehensible but raises the question why the confidence value is not 100%, since per being registered as members of the Wehrmacht (the Nazi Germany army) should have accordingly taken part in World War II. Therefore, for some very strong rules among objects $o_1 \rightarrow o_2$, we can conclude that subjects that *violate* such a rule by having only o_1 as an object value can be amended with a new fact containing o_2 as a new object. To discover the appropriate predicate again association rule mining is applied.

We further provide the possibility to evaluate the generated facts by allowing the user to chose a test dataset. For example applying the amendment approach on the complete DBpedia 3.6 dataset generated 26,775 new facts out of which 8,325 were included in the later DBpedia 3.7 version [4].

III. Tool Demonstration

ProLOD++ is purely browser-based to be either distributed for local execution or hosted as a service. An initial version, concentrating on the "profiling" functionality of Table I, can already be viewed at http://hpi-web.de/naumann/projects/prolod.html, keeping in mind that several extensions based on previous publications are still under development. All algorithms are implemented in Java and data is stored in a relational database. The user interface is based on GWT[4].

During the demo, users can (theoretically) bring along their own RDF dataset, upload it to ProLOD++ and begin analysis. Of course, we will have prepared several interesting datasets from various domains, including the cross-domain DBpedia dataset.

After initial analysis, users are able to browse the results across several tabs. Each tab displays the basic set of metadata; each item can be selected for drill-down. For instance, we can show a distribution of predicate usage. Clicking on a particular predicate reveals its value patterns and their frequencies. Clicking on a particular pattern then shows the actual data values and their frequencies. Visualizations are chosen as appropriate.

The user is able to vary the parameters such as, support or confidence, for mining configurations and to adapt the mining based applications such as, the discovery of synonyms or inverse predicates, to their datasets. Due to the very fast response times, users can interactively explore and analyze new datasets, and easily assess their volume, cleanliness, and ultimately their added-value. Finally, the user can try different suggestion configurations that support the insertion of new triples.

[4]http://www.gwtproject.org/

IV. Conclusion

In this paper, we motivated automatic meta-data generation for LOD datasets and propose to demonstrate ProLOD++ a profiling and mining tool for RDF datasets. We described relevant system components and visualization aspects that allows a user to explore and transform its data using a range of functionalities based on recently developed mining and profiling techniques. We believe that our tool supports data consumers to better understand and therefore consume or integrate LOD sources. Furthermore data publishers and engineers can use the mining functionalities to identify design flaws or mismatches that should be avoided in later releases of their datasets. We think that the demo will be interesting for researchers in the fields of RDF data management, ontology engineering, Linked Data, and web mining.

References

[1] Z. Abedjan, J. Lorey, and F. Naumann. Reconciling ontologies and the web of data. In *CIKM*, pages 1532–1536, Maui, HI, 2012.

[2] Z. Abedjan and F. Naumann. Advancing the discovery of unique column combinations. In *CIKM*, Glasgow, UK, 2011.

[3] Z. Abedjan and F. Naumann. Context and target configurations for mining RDF data. In *SMER*, Glasgow, UK, 2011.

[4] Z. Abedjan and F. Naumann. Improving rdf data through association rule mining. *Datenbank-Spektrum*, 13(2):111–120, 2013.

[5] Z. Abedjan and F. Naumann. Synonym analysis for predicate expansion. In *ESWC*, Montpellier, France, 2013.

[6] S. Auer, J. Demter, M. Martin, and J. Lehmann. Lodstats an extensible framework for high-performance dataset analytics. In *EKAW*, volume 7603, pages 353–362, 2012.

[7] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics (JWS)*, 7:154–165, 2009.

[8] C. Böhm, F. Naumann, Z. Abedjan, D. Fenz, T. Grütze, D. Hefenbrock, M. Pohl, and D. Sonnabend. Profiling linked open data with ProLOD. In *NTII*, pages 175–178, 2010.

[9] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[10] P. Heim, S. Lohmann, and T. Stegemann. Interactive relationship discovery via the semantic web. In *ESWC*, volume 6088, pages 303–317, 2010.

[11] A. Heise, J.-A. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable Discovery of Unique Column Combinations. *PVLDB*, 7(4), 2013.

[12] S. Kandel, R. Parikh, A. Paepcke, J. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Advanced Visual Interfaces*, 2012.

[13] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The yago-naga approach to knowledge discovery. *SIGMOD Record*, 37(4):41–47, 2009.

[14] F. Naumann. Data profiling revisited. *SIGMOD Record*, 42(4), 2013.