Accessing Linked Open Data via A Common Ontology[*]

Kiril Simov
Linguistic Modeling Department, IICT-BAS
KIvS@bultreebank.org

Atanas Kiryakov
Ontotext AD
Atanas.Kiryakov@ontotext.com

Abstract

In the paper we present the construction of the FactForge service. FactForge represents a reason-able view over several Linked Open Data (LOD) datasets including DBPedia, Freebase and Geonames. It enables users to easily identify resources in the LOD cloud by providing a general unified method for querying a group of datasets. FactForge is designed also as a use case for large-scale reasoning and data integration. We describe the datasets, ontologies, inference rules, and manipulations done over the data. The datasets are unified via a common ontology – PROTON, whose concepts are mapped to the concepts of the involved LOD datasets. Each of the mapping rules relates a PROTON class or a PROTON property to the corresponding class or property of the other ontologies. This mechanism of constructing a reason-able view over selected LOD datasets ensures that the redundant instance representations are cleaned as much as possible. The instances are grouped in equivalent classes of instances.

1 Introduction

Linked Open Data (LOD) (LOD 2014) facilitates the emergence of a web of linked data by publishing and interlinking open data on the web in RDF (Brickley and Guha 2004). The current datasets in LOD cover a wide spectrum of subject domains – biomedical, science, geographic, generic knowledge, entertainment, government (LOD Cloud 2011). As they constantly grow, we face the problem of conveniently accessing, manipulating and further developing them. It is believed that this large set of interconnected data will enable new classes of applications, making use of more sophisticated querying, knowledge discovery and reasoning.
However, LOD is characterized by heterogeneity and inconsistency of the datasets, which makes their automated use via algorithms difficult. A lot of research effort nowadays has been focused on detecting methods to cope with and preserve the diversity of LOD, which can scale and manage their increasing growth rates. These methods bring experimental results, which show that the state of the art is still far from the performance necessary for real life applications. Highly heterogeneous contexts such as LOD and the Web need mechanisms to ensure consistency based on a set of data agreed upon or commonly acceptable, shared by various datasets, and make them interconnected. In order to provide such a mechanism we use a reference layer, consisting of one or more ontologies with different degrees of generality built on top of LOD and interlinked with their schemata and instances. This is a viable and optimal solution for handling LOD heterogeneity. In the Semantic Web, the idea of having an integrated global ontology which extracts information from the local ontologies and provides a unified view through which users can query the local ontologies is unrealistic, since it is practically impossible to maintain this global ontology in a highly dynamic environment. The idea of building reference structures at the schema level has been advocated previously (Jain et al. 2010). They state that it would be valuable to have a schema describing the subject domain of the datasets in LOD. Besides the reference layer, we think that the actual datasets in LOD needs to be tuned to fit the reference layer. Such a tuning includes: unification of modelling principles for the various datasets and cleaning the instance data that do not fit the conceptualization. In the paper we present the preparation of datasets for one LOD service including these two components: a reference layer and cleaning of the involved datasets, based on the detected conceptual mismatches between the common ontology and conceptualization of each involved dataset.
LOD are valuable source of information of NLP like extraction of vobularies, names, features. In this paper we do not discuss any concrete NLP task or application, but for each of them we need a reliable LOD dataset - the topic of the paper.

[*] The research reported here is done within Ontotext AD

The structure of the paper is as follows: Section 2 gives the background of our idea. Section 3 focuses on the conctruction of FactForge. Section 4 concludes the paper.

2 Background

This section outlines the three components our approach is based on: (a) conceptual schema of the world (ontologies); (b) instance data; and (c) mechanisms for inferring new information from these two sources of information. First, we provide a general overview of ontologies with emphasis on upper level ontologies. Then, we characterize LOD and describe an approach to using the LOD data with reasoning.

Ontologies. Ontologies are defined as "a formal, explicit specification of a shared conceptualization" (Studer et al. 1998). They are sets of definitions in a formal language for terms describing the world. Ontologies organize knowledge domains in concepts and relations between them. They allow for inheritance of properties and characteristics, and for reasoning according to different logics. These are some of the powerful mechanisms of ontologies that offer increased knowledge coverage, consistency, and lack of redundancy or contradiction. Depending on the generality of the knowledge domains they cover, several types of ontologies are distinguished. These are upper-level ontologies, domain ontologies and application ontologies. Upper-level ontologies, or foundational ontologies, describe very general concepts that can be used across multiple domains; examples include DOLCE[1], SUMO[2], and PROTON[3]. Domain ontologies cover the conceptualization of given subject domains. They describe concepts and relationships representative for the subject domain like biology, vehicle sales, product types, etc. The most common ontology design principles include: defining the scope of the ontology, creating a balanced class hierarchy, providing methods to evaluate the concepts and properties, as well as consistency checking. The OntoClean method (Guarino, N., & Welty 2002) is a very popular ontology design method. It recommends distinguishing between type and role when defining the concepts. It uses metaproperties to check the consistency of the ontology with predefined constraints helping to discover taxonomic errors. Data driven ontologies, such as the ontology of DBpedia[4], select the concepts based on the availability of data instantiating them.

Linked Open Data. The notion of "linked data" is defined by Tim Berners-Lee (Berners-Lee 2006), as RDF graphs, published on the WWW so that one can explore them across servers by following the links in the graph in a manner similar to the way the HTML web is navigated. "Linked data" are constituted by publishing and interlinking open data sources, following the principles of:

- Using URIs as names for things;
- Using HTTP URIs, so that people can look up these names;
- Providing useful information when someone looks up a URI;
- Including links to other URIs, so that people can discover more things.

To this end, data publishers should make sure that:

- The "physical" addresses of the pieces of published data are the same as the "logical" addresses, used as RDF identifiers (URIs);
- Upon receiving an HTTP request, the server should return a set of triples describing the resource.

LOD provide sets of referenceable, semantically interlinked resources with defined meaning. The central dataset of the LOD is DBpedia. Because of the many mappings between other LOD datasets and DBpedia, the latter serves as a sort of a hub in the LOD graph ensuring a certain level of connectivity. LOD is rapidly growing. The largest number of datasets in LOD belongs to the bio-medical domain. Another big subject area in the LOD cloud is scientific literature collection; entertainment data; government data like; Language dataetc. Finally, some datasets contain general-purpose encyclopedic knowledge such as DBpedia and Freebase, and geographic knowledge such as Geonames, etc.

The use of LOD and the development of applications based on it are difficult because the different LOD datasets are rather loosely connected chunks of information, facts, and instances. They have varying levels of completeness and external linkages. They are mainly connected at the instance level, thus losing the benefits from the enrichment of the data with implicit factual knowledge, when ontologies and schema-level mappings are involved. Even the linkage between instances of

[1] http://www.loa-cnr.it/DOLCE.html

[2] http://www.ontologyportal.org/

[3] http://www.ontotext.com/proton-ontology

[4] http://dbpedia.org/About

different datasets in the LOD cloud, via the predicate `owl:sameAs` shows drawbacks due to the fact that the instances are not described in the same way in the different datasets. They are, strictly speaking, not the same. For instance, New York's population in DBpedia is given as of July 2009, and counts 8,391,881, whereas in Freebase it is 8,363,710 as of 2008. Nevertheless, the two instances of New York from DBpedia and from Freebase are linked together with `owl:sameAs`, which implies that the two resources are fully identical. Yet, the "facts" for each instance differ. Another example points to the country of Kosovo. In DBpedia, it is described as a country, whereas in Freebase, it is denoted as a region. Still these two instances are reliably linked with `owl:sameAs`. Such divergences make the use of LOD data challenging in knowledge demanding applications or for reasoning tasks. On the other hand, introducing schema-level alignment of LOD datasets would provide significant advantages in ensuring the consistency of linkages. Such linkages would enable applications that can answer queries requiring multiple and disparate information sources. The quality of the data in the LOD cloud and their linkage are not the only challenges for the applications. The RDF datasets are supplied with vocabularies, which imply inference and generation of implicit facts. This considerably increases the overall number of facts available for exploration and poses the question of managing LOD. Using linked data for data management is considered to have great potential for the transformation of the web of data into a giant global graph (Heath and Bizer 2011). Still, there are several challenges that have to be overcome to make this possible, namely:

- LOD are hard to comprehend – the fact that multiple datasets are interlinked and accessible in the same data format is not enough to deal with hundreds of data schemata, ontologies, vocabularies and data modeling patterns;
- Diversity comes at a price – often there are tens of different ways of expressing one and the same piece of information even in a single dataset, such as DBpedia;
- LOD is unreliable – many of the servers behind LOD today are slow and have down times higher than the one acceptable for most of the data management setups;
- Dealing with data distributed on the web is slow – a federated SPARQL query that uses, say, three servers within several joins can be very slow;
- No consistency is guaranteed – low commitment to the formal semantics and intended use of the ontologies and schemata.

Using reason-able views (Kiryakov et al. 2009), described below, is one solution to the problem of LOD management. Reason-able views are the experimental setting for the approach presented in this paper.

Reason-Able Views (RAV). Reasoning within LOD with standard methods of sound and complete inference with respect to First Order Predicate Calculus is practically infeasible. The closed-world assumption for sound and complete reasoning is practically inapplicable in a web context and has never been even considered for the web of data. Due to the nature of the data in LOD in its current state, inference with them in many cases is useless, as it derives many false statements. Having datasets dispersed in different locations makes reasoning with them impractical. Reason-able views are an approach to reasoning over and managing linked data. Reason-able view is an assembly of independent datasets, which can be used as a single body of knowledge with respect to reasoning and query evaluation. The key principles of constructing reason-able views can be summarized as follows:

- Group selected datasets and ontologies in a compound dataset;
- Clean up, post-process and enrich the datasets if necessary. Do this conservatively, in a clearly documented and automated manner, so that (a) the operation can easily be performed each time a new version of one of the datasets is published; and (b) the users can easily understand the intervention made;
- Load the compound dataset into a single semantic repository and perform inference with respect to tractable OWL dialects;
- Define a set of sample queries against the compound dataset. These determine the "level of service" or the "scope of consistency" contract offered by the reason-able view.

Each RAV aims at lowering the cost and the risks of using specific LOD datasets. The design objectives behind each reason-able view are to:

- Make reasoning and query evaluation feasible;
- Lower the cost of entry through interactive user interfaces and retrieval methods such as URI auto-completion and RDF search;

- Guarantee a basic level of consistency – the sample queries guarantee the consistency of the data;
- Guarantee availability – all data is the same repository;
- Easier exploration and querying of unseen data – sample queries provide re-usable extraction patterns.

RAVs are built according to certain design principles, e.g.:

- All datasets in the view represent linked data;
- Single set of reasonability criteria is imposed on all datasets;
- Each dataset is connected to at least one of the others.

RAVs are implemented in two public services, namely, FactForge and LinkedLifeData.

3 Construction of FactForge

FactForge[5] represents a reason-able view over several important Linked Open Data datasets. It enables users to easily identify resources in the LOD cloud by providing a general unified method for querying a whole group of datasets. FactForge is designed also as a use-case for large-scale reasoning and data integration. In brief, the datasets are unified via a common ontology – PROTON, whose concepts are mapped to the concepts of the involved LOD datasets. We do this by a set of rules. Each of them maps a PROTON class or a PROTON property to the corresponding class or property of the other ontologies. This mechanism of constructing a reason-able view over selected LOD datasets ensures that the redundant instance representations (classes and properties) are cleaned as much as possible. The instances are grouped in equivalent classes of instances. Finally, the instances in these datasets are linked via `owl:sameAs` statements. FactForge development can be divided into six main steps:

1. Selecting the LOD datasets
2. Checking each dataset for consistency
3. Mapping the PROTON concepts to the respective LOD datasets concepts
4. Cleaning the datasets from any discrepancies between the concepts in the different datasets and PROTON
5. Loading all datasets in a joint repository
6. Loading owl:sameAs statements and checking for consistency

[5] http://www.ontotext.com/factforge

Here, we also present solutions for resolving discrepancies when mapping concepts from the central datasets in FactForge and PROTON, as well as the way of cleaning the datasets. In some of the cases, we have to add new instances, which are introduced via inference rules. Ultimately, FactForge provides a deeper understanding of: the Linked Open Data available on the web, some peculiarities of the datasets conceptualization and the problems of integrating the different LOD datasets.

3.1 Reference Layer Mapping Rules

This section describes the methodology for creating a correspondence between two dataset conceptualizations of the real world. When constructing such a correspondence, several manipulations of the datasets facts are conducted: (1) introducing new individuals; (2) deleting some individuals; (3) modifying some individuals; (4) inserting/deleting/updating relations between individuals; (5) inserting/deleting/updating characteristics of the individuals. The idea behind LOD is that such transformations are minimal. Ideally, there should be no transformations at all. We respect this recommendation, as much as possible, when constructing FactForge, except in cases where the resulting reason-able view contradicts with the conceptualization of the PROTON ontology.
Thus, in the development of FactForge, our first aim is to support a full querying of the resulting repository via PROTON. We use only `rdfs:subClassOf` or `rdfs:subPropertyOf` statements in order to ensure a complete mapping coverage of the PROTON ontology to the other schemas in FactForge. Generally, the mapping statements can be arbitrary couples but in most cases they are simply `rdfs:subClassOf` or `rdfs:subPropertyOf` statements between classes or properties explicitly defined in the PROTON ontology, and the ontology or the schema of a given dataset. For example[6]:

[6] Here are the namespace declarations used in the document:
```
@prefix ptop:
<http://www.ontotext.com/proton/protontop#> .
@prefix pext:
<http://www.ontotext.com/proton/protonext#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-
schema#> .
@prefix dbp: <http://dbpedia.org/ontology/> .
@prefix dbp-prop:<http://dbpedia.org/property/> .
@prefix fb: <http://rdf.freebase.com/ns/> .
```

```
dbp:SportsTeam
  rdfs:subClassOf  pext:Team .
foaf:homepage
  rdfs:subPropertyOf  pext:hasWebPage .
```

However, due to the different conceptualizations, in some cases a more complex mapping is needed. For example, in the Geonames dataset geographical objects are classified by codes and not by an ontology hierarchy. In such cases the mapping is done by more complex statements such as:

```
[ rdf:type owl:Restriction ;
  owl:onProperty geo-ont:featureCode ;
  owl:hasValue geo-ont:A.PCL ]
      rdfs:subClassOf pext:Country .
```

Some of these compound statements require adding new individuals. In such cases, we use the OWLIM inference rules to create the necessary additions. Here is an example:

```
//dbp-ont:PrimeMinister rdfs:subPropertyOf
// [ptop:hasPosition [pupp:hasTitle]].
  Id:PM
      p  <rdf:type> <dbp-ont:PrimeMinister>
     -------------------------------------
      p <ptop:hasPosition> j
      j <pext:hasTitle> <pext:PrimeMinister
```

Here, the inference rule is necessary because the conceptualizations in the DBPedia ontology and in the PROTON ontology are different. In DBPedia, Prime Minister belongs to a class of politicians, which is a class of person, while in PROTON, Prime Minister is a title of a job position. Thus, in DBPedia, a given Prime Minister is an individual whereas in PROTON he is an individual who has a position PrimeMinister. Since the instance data about the position itself (j in the rule above) is missing in the DBPedia dataset, it has to be created so that the mapping between the two ontologies is consistent.

3.2 Cleaning Two LOD datasets

In this section we present two of the most popular LOD datasets - DBPedia and Freebase with respect to discrepancies between their conceptualization and ontology in the reference layer.

DBPedia Ontology and Dataset. The DBPedia dataset is created by extracting structured information from Wikipedia and presenting it in an RDF form (http://dbpedia.org/About). The conceptualization of the DBPedia dataset is based on the categories that are designed and implemented in Wikipedia, i.e. the data in the info-box section of the articles. This conceptualization is presented as an ontology. For our purposes, we have used version 3.8. It contains 359 classes, 800 object properties and 975 data types. The instances in the DBPedia dataset are classified according to the conceptual information in its ontology and some other well-known ontologies like: http://schema.org and http://xmlns.com/foaf/spec/. In addition, some of the classes and properties of these other ontologies are used in the definition of the DBPedia ontology. In the majority of cases, the conceptualizations of the DBPedia and PROTON ontologies are compatible and the mapping between them is straightforward as discussed earlier. However, there are still some differences as illustrated in the following two examples:

Architect as a Person. In the DBPedia ontology, many roles in society, mainly performed by persons, are formalized as subclasses of the class dbp-ont:Person.

```
dbp-ont:Architect
      rdf:type owl:Class;
      rdfs:subClassOf dbp-ont:Person .
```

The definition in PROTON is:

```
pext:Architect
 rdf:type pext:Profession ;
 rdfs:comment "A profession of planning,
    design and oversight of the
    construction of buildings and some
    other artefacts. (Wikipedia)"@en .
```

and

```
pext:Profession
  rdf:type owl:Class ;
  rdfs:subClassOf pext:SocialFunction .
```

The main difference is that in PROTON the class pext:Architect is defined as a profession and a social function, in order for someone (or something) to have this profession. This means that not only persons can perform it. While in DBPedia the definition follows the logic that all architects described in Wikipedia are, in fact, persons. It is relatively easy to overcome such conceptual differences by an appropriate mapping between the two ontologies:

```
dbp-ont:Architect rdfs:subClassOf
  [ rdf:type owl:Restriction ;
    owl:onProperty pext:hasProfession ;
    owl:hasValue pext:Architect ] .
```

This statement determines that all instances of `dbp-ont:Architect` correspond to the instances of the PROTON ontology with the profession `pext:Architect`.

Sport as an Activity. Another example is the definition of Sport. DBPedia defines it as follow:

```
dbp-ont:Sport
```

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

```
        rdf:type owl:Class;
    rdfs:comment "A sport is commonly
        defined as an organized,
        competitive, and skillful
        physical activity."@en;
  rdfs:subClassOf dbp-ont:Activity .
```

and PROTON defines it as:

```
pext:Sport
 rdf:type owl:Class ;
 rdfs:comment "A specific type of
        sport game"@en ;
 rdfs:subClassOf pext:SocialAbstraction.
```

The difference is that in DBPedia, Sport is a specific activity and its characteristics such as game rules, number of participants, etc. are not defined in the class `dbp-ont:Sport`. In PROTON the characteristics of the sport game are defined in the class `pext:Sport` as a social abstraction. The actual realization of the definition as a sport event is an instance of activity. Unfortunately, any mapping between the two ontologies cannot solve this conceptual difference. The following mappings:

```
dbp-ont:Activity
  rdfs:subClassOf pext:Activity .
```

and

```
dbp-ont:Sport
  rdfs:subClassOf pext:Sport .
```

automatically make all instances of the class `dbp-ont:Sport` in PROTON to be simultaneously instances of the classes `ptop:Happening` and `ptop:Abstract`, which are mutually disjoint.

In FactForge such conceptualization differences between the two ontologies are solved by not loading the DBPedia ontology into the FactForge repository. In this way, we make use of the richness of the DBPedia instances but impose the conceptualization of PROTON ontology over it.

Another reason for not loading the DBPedia ontology is that the definitions in the DBPedia ontology also contain mappings to other ontologies. However, we believe that including ontology statements referring to classes (properties, etc) of other ontologies is not a good practice. First, presenting the necessary conceptualization requires importing the other ontology. And second, this can introduce some contradictions in the ontology that uses these statements. For example, the DBPedia ontology contains some statements from the Schema ontology (http://schema.org). However, because DBPedia is not an extension of the Schema ontology, therefore it is better to store these statements separately. If they are included in the definitions of the DBPedia classes, this can lead to some contradictions as illustrated in the examples below for University and College:

```
dbp-ont:University
 rdf:type owl:Class;
 rdfs:subClassOf
      dbp-ont:EducationalInstitution ;
 owl:equivalentClass
      schema:CollegeOrUniversity .
```

and

```
dbp-ont:College a owl:Class;
 rdf:type owl:Class;
 rdfs:subClassOf
     dbp-ont:EducationalInstitution ;
 owl:equivalentClass
     schema:CollegeOrUniversity .
```

Using `owl:equivalentClass` makes these two classes - `dbp-ont:University` and `dbp-ont:College` - the same. Such equivalent statements are difficult to be noticed in the DBPedia ontology as it is full of them but it is also not very easy to use DBPedia without such statements. The instance data also contains statements that result from inferences from the DBPedia ontology. In order to avoid all conceptualizations that follow from the DBPedia ontology we have to clean the DBPedia instance data from such inferences. Here are some examples:

Subclass - Superclass inference. In the DBPedia instance data, each instance of sport is classified as sport but also as an activity. Therefore, even if we do not load the DBPedia ontology into the FactForge repository, this inference is present in the instance data. Thus, the classification of the DBPedia sport instances will also be wrong in PROTON when mapping PROTON to DBPedia. To clean this instance data statement we have created a deletion statement of the following type:

```
delete {?s a dbp-ont:SuperClass} where
    { ?s a dbp-ont:SubClass .
      ?s a dbp-ont:SuperCLass . }
```

Here is an example:

```
delete {?s a dbp-ont:Activity} where
    { ?s a dbp-ont:Sport .
      ?s a dbp-ont:Activity . }
```

In this way, if there is a statement for a subclass, we delete all the statements for the super classes. After that, we use the inference mechanisms of the repository to make the inferences that follow from the mapping to the PROTON ontology.

rdfs:domain and rdfs:range statements. In the DBPedia instance data, some statements for domain and range have properties connected to instances that do not belong to the appropriate classes. Such unclassified instances in DBPedia could be wrongly classified in PROTON, based on these domain and range statements. In order to clean such cases we use queries of the following type:

```
delete {?s dbp-ont:dbpediaProperty ?y }
where
{?s dbp-ont:dbpediaProperty ?y .
 ?y rdf:type ?c .
filter(
         ?c = dbp-ont:Class01
     ||   ?c = dbp-ont:Class02
     ||   ...
     ## List of all unappropriate classes
     )
}
```

Here is part of an example of the property `dbp-ont:birthPlace`.

```
delete {?s dbp-ont:birthPlace ?y } where
{?s dbp-ont:birthPlace ?y .
 ?y rdf:type ?c .
filter(?c = dbp-ont:AcademicJournal
   ||   ?c = dbp-ont:Activity
   ||   ?c = dbp-ont:AdministrativeRegion
...
        )
}
```

Apart from the deleted statements discussed earlier, we have deleted all instance data described by statements using classes that are not from the DBPedia ontology. In this way, the DBPedia instance data has a clean interpretation in terms of the PROTON conceptualization.

Freebase Dataset. Freebase[7] is a community-curated database of well-known people, places, and things. In Freebase, real-world entities are represented as topics. There are topics for movie stars, countries, cities, etc. The information for each topic is structured in three levels as defined in the Freebase schema. The first layer comprises several domains (76). Each domain is defined by type (second layer) and each type has properties (third layer). The types are connected via the special relation inclusion of type. This relation connects more specific types with more general types: the type fb:base/litcentral/named_person includes the type: fb:people/person. It is not possible to interpret this relation as superclass-to-subclass relation, because it is not strict in the sense that each instance of the subclass inherits the properties of the instance of the super class. For example, the type fb:film/actor also includes the type fb:people/person. But its definition is: "The Film Actor type includes people (and credited animals) who have appeared in any film ...". Therefore, in most cases, the instances of the type fb:film/actor are people but there are also cases where they are not. Thus, the interpretation of the type inclusion relation is not strict with respect to inheritance of the properties from the included type. In the example above, if the film actor is a person, then he or she inherits all properties from the type for persons. But if it is not a person, then it does not inherit any of these properties. Instead, it inherits properties from some other type(s).

[7] http://www.freebase.com/

These peculiarities of the Freebase schema impose some restrictions over the mapping to the PROTON ontology. Mapping so many types and properties requires more extensive work. Therefore, for our purposes, we have mapped only the types with more than 500 instances in the Freebase dataset to the PROTON concepts. Another criterion is that the mapping does not produce any misclassification of some instances. For many types the mapping is straightforward:

```
fb:location.location
  rdf:type owl:Class;
  rdfs:comment "The Location type is
   used for any topic with a fixed
   location..."@en ;
  rdfs:label "Location";
      rdfs:subClassOf ptop:Location .
```

For types representing professions and other social roles, the mappings are similar to the mapping used for the DBPedia ontology:

```
  fb:military-militarycommander
    rdfs:subClassOf
      [rdf:type owl:Restriction ;
       owl:onProperty pext:hasTitle ;
       owl:allValuesFrom
pext:Commander].
```

Some of the types are mediators between a type and a grouping of several other types. This is mainly used to represents event information. For example, the type *Website ownership* describes an event of owning a website by an agent for some period. A website can be owned by different agents in different periods, thus it is important that these ‘owning’ events are represented as different instances in the dataset.

At present, we have not yet mapped the mediator types to PROTON. For this type of mapping it is necessary to use an appropriate subclass of the class ptop:Happening. For example, the type *Website ownership* can be mapped to a subclass of the class ptop:Situation, where the start and end date of the ownership are stated, the owner and the address of the website are specified, etc. As this requires huge extension of PROTON, it is not featured in the current version.

In the original dataset, there are also several errors in the instance classification. For example, organisation and location are very often represented by the same instance. More specifically, the types `fb:organization.organization` and `fb:location.location` have 42763 instances in common. We believe that such cases result from the linguistic intuition of the users who created the data in question. In many cases, the same word denotes both the meaning of an institution and a location. We do not consider this a good practice for the semantic representation in LOD and we think that it should be avoided. The different classes (types in Freebase) have different properties. Although the Freebase types are not strict in inheriting properties, some types are still not mutually compatible (intuitively). For example, due to this misclassification, the instance of the United States of America (https://www.freebase.com/m/09c7w0) is not only an instance of the types Country, Location but also of Food. We believe that such knowledge has to be represented in a different way.

It is important to note that correcting such cases of instances classification to many disjoint types (classes) is outside the scope of the current version of FactForge. In future, we envisage to introduce new instances for each disjoint class and to keep relations between them where necessary via appropriate properties. Although we could perform such an extension of Freebase, in our view, it is better this to be done in the original dataset. We consider these mismatches as a result from crowdsourcing where some of the providers of knowledge where influenced by the semantics of natural language.

4 Conclusion

In this paper we present some problems in accessing LOD via a common ontology. The main problems of using this approach with respect to involved datasets are demonstrated via examples from two of the most popular LOD datasets: DBPedia and Freebase. The main lessons learned are as follows:

1. The world can be modelled in many different ways, which can be formally incompatible but still understandable by human users. It is true that the main value of a dataset is in its usefulness to the stakeholders. However, this is not enough in terms of the Semantic Web where the goal is to have LOD datasets that can be processed by machines. To achieve this, it is necessary to apply some formal evaluation of the represented knowledge.
2. The incompatibility can appear on different levels: granularity of conceptualization, representation of different kinds of knowledge (for example, the difference between sortals and roles), etc. Generally, the conclusion is that if we want LOD to achieve their goals, they should not only follow some formats but also their conceptualizations should adhere to certain restrictions and ensure compatibility.
3. Constructing new ontologies based on existing ones has to incorporate the complete semantics of the corresponding ontologies instead of just fragments of them. Such an approach will have an effect on the consistency of the new ontologies and their interoperability with the existing ones.

In our view LOD needs more requirements on semantic level in order to be more reliable web of semantically linked open data..

Acknowledgements

This work was partially supported by MULTISENSOR project (FP7-610411) .

References

Linking Open Data. Retrieved from W3C Semantic Web Education and outreach community project: http://linkeddata.org/. (2014).

Brickley D., & Guha R.V. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema/. (2004).

State of the LOD Cloud. Retrieved from http://www4.wiwiss.fu-berlin.de/lodcloud/state. (2011).

Jain, P., Hitzler, P., Sheth, A. P., Verma, K., & Yeh, P. Z. Ontology Alignment for Linked Open Data. In Patel-Schneider, Y. P. P. (Ed.), Proceedings of the 9th International Semantic Web Conference. Shanghai. (2010).

Studer R, Benjamin V. R., & Fensel D. Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering , 25 ((1-2)), pp. 161-199. (1998).

Guarino, N., & Welty, C. Evaluating Ontological Decisions with OntoClean. Communications of the ACM, 45(2) , pp. 61-65. (2002).

Berners-Lee, T. Design Issues: Linked Data. Retrieved from http://www.w3.org/DesignIssues/LinkedData.html. (2006).

Heath, T., & Bizer, C. Linked Data: Evolving the Web into a Global Data Space. (J. H. (eds.), Ed.) Synthesis Lectures on the Semantic Web: Theory and Technology (1:1), 1-136. (2011).

Kiryakov, A., Ognyanoff, D., Velkov, R., Tashev, Z., & Peikov, I. LDSR: Materialized Reason-able View to the Web of Linked Data. In R. H. Patel-Schneider (Ed.), Proceedings of OWLED 2009 . Chantilly, USA. (2009).