Benchmarking the Performance of Linked Data Translation Systems

Carlos R. Rivero*
University of Sevilla, Spain
carlosrivero@us.es

Andreas Schultz
Freie Universität Berlin, Germany
a.schultz@fu-berlin.de

Christian Bizer
Freie Universität Berlin, Germany
christian.bizer@fu-berlin.de

David Ruiz
University of Sevilla, Spain
druiz@us.es

ABSTRACT

Linked Data sources on the Web use a wide range of different vocabularies to represent data describing the same type of entity. For some types of entities, like people or bibliographic record, common vocabularies have emerged that are used by multiple data sources. But even for representing data of these common types, different user communities use different competing common vocabularies. Linked Data applications that want to understand as much data from the Web as possible, thus need to overcome vocabulary heterogeneity and translate the original data into a single target vocabulary. To support application developers with this integration task, several Linked Data translation systems have been developed. These systems provide languages to express declarative mappings that are used to translate heterogeneous Web data into a single target vocabulary. In this paper, we present a benchmark for comparing the expressivity as well as the runtime performance of data translation systems. Based on a set of examples from the LOD Cloud, we developed a catalog of fifteen data translation patterns and survey how often these patterns occur in the example set. Based on these statistics, we designed the LODIB (Linked Open Data Integration Benchmark) that aims to reflect the real-world heterogeneities that exist on the Web of Data. We apply the benchmark to test the performance of two data translation systems, Mosto and LDIF, and compare the performance of the systems with the SPARQL 1.1 CONSTRUCT query performance of the Jena TDB RDF store.

Categories and Subject Descriptors

D.2.12 [**Interoperability**]: Data mapping;
H.2.5 [**Heterogeneous Databases**]: Data translation

*Work partially done whilst visiting Freie Universität Berlin.

LDOW2012, April 16, 2012, Lyon, France.

1. INTRODUCTION

The Web of Linked Data is growing rapidly and covers a wide range of different domains, such as media, life sciences, publications, governments, or geographic data [4, 13]. Linked Data sources use vocabularies to publish their data, which consist of more or less complex data models that are represented using RDFS or OWL [13]. Some data sources try to reuse as much from existing vocabularies as possible in order to ease the integration of data from multiple sources [4]. Other data sources use completely proprietary vocabularies to represent their content or use a mixture of common and proprietary terms [7].

Due to these facts, there exists heterogeneity amongst vocabularies in the context of Linked Data. According to [5], on the one hand, 104 out of the 295 data sources in the LOD Cloud only use proprietary vocabularies. On the other hand, the rest of the sources (191) use common vocabularies to represent some of their content, but also often extend and mix common vocabularies with proprietary terms to represent other parts of their content. Some examples of the use of common vocabularies are the following: regarding publications, 31.19% data sources use the Dublin Core vocabulary, 4.75% use the Bibliographic Ontology, or 2.03% use the Functional Requirements for Bibliographic Records; in the context of people information, 27.46% data sources use the Friend of a Friend vocabulary, 3.39% use the vCard ontology, or 3.39% use the Semantically-Interlinked Online Communities ontology; finally, regarding geographic data sets, 8.47% data sources use the Geo Positioning vocabulary, or 2.03% use the GeoNames ontology.

To solve these heterogeneity problems, mappings are used to perform data translation, i.e., exchanging data from the source data set to the target data set [19, 21]. Data translation, a.k.a. data exchange, is a major research topic in the database community, and it has been studied for relational, nested relational, and XML data models [3, 10, 11]. Current approaches to perform data translation rely on two types of mappings that are specified at different levels, namely: correspondences (modelling level) and executable mappings (implementation level). Correspondences are represented as declarative mappings that are then combined into executable mappings, which consist of queries that are

executed over a source and translate the data into a target [7, 18, 19].

In the context of executable mappings, there exists a number of approaches to define and also automatically generate them. Qin et al. [18] devised a semi-automatic approach to generate executable mappings that relies on data-mining; Euzenat et al. [9] and Polleres et al. [17] presented preliminary ideas on the use of executable mappings in SPARQL to perform data translation; Parreiras et al. [16] presented a Model-Driven Engineering approach that automatically transforms handcrafted mappings in MBOTL (a mapping language by means of which users can express executable mappings) into executable mappings in SPARQL or Java; Bizer and Schultz [7] proposed a SPARQL-like mapping language called R2R, which is designed to publish expressive, named executable mappings on the Web, and to flexible combine partial executable mappings to perform data translation. Finally, Rivero et al. [19] devised an approach called Mosto to automatically generate executable mappings in SPARQL based on constraints of the source and target data models, and also correspondences between these data models. In addition, translating amongst vocabularies by means of mappings is one of the main research challenges in the context of Linked Data, and it is expected that research efforts on mapping approaches will be increased in the next years [4]. As a conclusion, a benchmark to test data translation systems in this context seems highly relevant.

To the best of our knowledge, there exist two benchmarks to test data translation systems: STBenchmark and DTSBench. STBenchmark [1] provides eleven patterns that occur frequently when integrating nested relational models, which makes it difficult for at least some of the patterns to extrapolate to our context due to a number of inherent differences between nested relational models and the graph-based RDF data model that is used in the context of Linked Data [14]. DTSBench [21] allows to test data translation systems in the context of Linked Data using synthetic data translation tasks only, without taking real-world data from Linked Data sources into account.

In this paper, we present a benchmark to test data translation systems in the context of Linked Data. Our benchmark provides a catalogue of fifteen data translation patterns, each of which is a common data translation problem in the context of Linked Data. To motivate that these patterns are common in practice, we have analyzed 84 random examples of data translation in the Linked Open Data Cloud. After this analysis, we have studied the distribution of the patterns in these examples, and have designed LODIB, the Linked Open Data Integration Benchmark, to reflect this real-world heterogeneity that exists on the Web of Data.

The benchmark provides a data generator that produces three different synthetic data sets, which reflect the pattern distribution. These source data sets need to be translated into a single target vocabulary by the system under test. This generator allows us to scale source data and it also automatically generates the expected target data, i.e., after performing data translation over the source data. The data sets reflect the same e-commerce scenario that we already used for the BSBM benchmark [6].

Table 1: Prefixes of the sample patterns

Prefix	URI
rdfs:	http://www.w3.org/2000/01/rdf-schema#
xsd:	http://www.w3.org/2001/XMLSchema#
fb:	http://rdf.freebase.com/ns/
dbp:	http://dbpedia.org/ontology/
lgdo:	http://linkedgeodataorg/ontology/
gw:	http://govwild.org/ontology/
po:	http://purl.org/ontology/po/
lgdp:	http://linkedgeodata.org/property/
movie:	http://data.linkedmdb.org/resource/movie/
db:	http://www4.wiwiss.fu-berlin.de ↪ /drugbank/resource/drugbank/
skos:	http://www.w3.org/2004/02/skos/core#
foaf:	http://xmlns.com/foaf/spec/
grs:	http://www.georss.org/georss/

LODIB is designed to measure the following: 1) Expressivity: the number of mapping patterns that can be expressed in a specific data translation system; 2) Time performance: the time needed to perform the data translation, i.e., loading the source file, executing the mappings, and serializing the result into a target file. In this context, LODIB provide a validation tool that examines if the source data is represented correctly in the target data set: we perform the data translation task in a particular scenario using LODIB, and the target data that we obtain are the expected target data when performing data translation using a particular system.

This paper is organized as follows: Section 2 presents the mapping patterns of our benchmark; in Section 3, we describe the 84 data translation examples from the LOD Cloud that we have analyzed, and the counting of the occurrences of mapping patterns in the examples; Section 4 deals with the design of our benchmark; Section 5 describes the evaluation of our benchmark with two data translation systems (Mosto and LDIF), and compares their performance with the SPARQL 1.1 performance of the Jena TDB RDF store; Section 6 describes the related work on benchmarking in the Linked Data context; and, finally, Section 7 recaps on our main conclusions regarding LODIB.

2. MAPPING PATTERNS

A mapping pattern represents a common data translation problem that should be supported by any data translation system in the context of Linked Data. Our benchmark provides a catalogue of fifteen mapping patterns that we have repeatedly discovered as we analyzed the heterogeneity between different data sources in the Linked Open Data Cloud. In the rest of this section, we present these patterns in detail. Note that for vocabulary terms in concrete examples we use the prefixes shown in Table 1.

Rename Class (RC). Every source instance of a class C is reclassified into the same instance of the renamed class C' in the target. An example of this pattern is the renaming of class *fb:location.citytown* in Freebase into class *dbp:City* in DBpedia.

Rename Property (RP). Every source instance of a property P is transformed into the same instance

of the renamed property *P'* in the target. An example is the renaming of property *dbp:elevation* in DBpedia into property *lgdo:ele* in LinkedGeoData, in which both properties represent the elevation of a geographic location.

Rename Class based on Property (RCP). This pattern is similar to the Rename Class pattern but it is based on the existence of a property. Every source instance of a class *C* is reclassified into the same instance of the renamed class *C'* in the target, if and only if, the source instance is related with another instance of a property *P*. An example is the renaming of class *dbp:Person* in DBpedia into class *fb:people.deceased_person* in Freebase, if and only if, an instance of *dbp:Person* is related with an instance of property *dbp:deathDate*, i.e., if a deceased person in Freebase exists, there must exist a person with a date of death in DBpedia.

Rename Class based on Value (RCV). This pattern is similar to the previous pattern, but the property instance must have a specific value *v* to rename the source instance. An example is the renaming of class *gw:Person* in GovWILD into class *fb:government.politician* in Freebase, if and only if, each instance of *gw:Person* is related with an instance of property *gw:profession* and its value is the literal "politician". This means that only people whose profession is politician in GovWILD are translated into politicians in Freebase.

Reverse Property (RvP). This pattern is similar to the Rename Property pattern, but the property instance in the target is reversed, i.e., the subject is interchanged with the object. An example is the reverse of property *fb:airports_operated* in Freebase into property *dbp:operator* in DBpedia, in which the former relates an operator with an airport, and the latter relates an airport with an operator.

Resourcesify (Rsc). Every source instance of a property *P* is split into a target instance of property *P'* and an instance of property *Q*. Both instances are connected using a fresh resource, which establishes the original connection of the instance of property *P*. Note that the new target resource must be unique and consistent with the definition of the target vocabulary. An example is the creation of a new URI or blank node when translating property *dbp:runtime* in DBpedia into *po:duration* in BBC by creating a new instance of property *po:version*.

Deresourcesify (DRsc). Every source instance of a property *P* is renamed into a target instance of property *P'*, if and only if, *P* is related to another source instance of a property *Q*, that is, both instances use the same resource. In this case, the source needs more instances than the target to represent the same information. An example of this pattern is that an airport in DBpedia is related with its city served by property *dbp:city*, and the name of this city is given as value of *rdfs:label*. This is transformed into property *lgdp:city_served* in LinkedGeoData, which relates an airport with its city served (as literal).

1:1 Value to Value (1:1). The value of every source instance of a property *P* must be transformed by means of a function into the value of a target instance of property *P'*. An example is *dbp:runtime* in DBpedia is transformed into *movie:runtime* in LinkedMDB, in which the source is expressed in seconds and the target in minutes.

Value to URI (VtU). Every source instance of a property *P* is translated into a target instance of property *P'* and the source object value is transformed into an URI in the target. An example of this pattern is property *grs:point* in DBpedia, which is translated into property *fb:location.location.geolocation* in Freebase, and the value of every instance of *grs:point* is transformed into an URI.

URI to Value (UtV). This pattern is similar to the previous one but the source instance relates to a URI that is transformed into a literal value in the target. An example of the URI to Value pattern is property *dbp:wikiPageExternalLink* in DBpedia that is translated into property *fb:common.topic.official_website* in Freebase, and the URI of the source instance is translated to a literal value in the target.

Change Datatype (CD). Every source instance of a datatype property *P* whose type is *TYPE* is renamed into the same target instance of property *P'* whose type is *TYPE'*. An example of this pattern is property *fb:people.person.date_of_birth* in Freebase whose type is *xsd:dateTime*, which is translated into target property *dbp:birthDate* in DBpedia whose type is *xsd:date*.

Add Language Tag (ALT). In this pattern, every source instance of a property *P* is translated into a target instance of property *P'* and a new language tag *TAG* is added to the target literal. An example of this pattern is that *db:genericName* in Drug Bank is renamed into property *rdfs:label* in DBpedia and a new language tag "@en" is added.

Remove Language Tag (RLT). Every source instance of a property *P* is translated into a target instance of property *P'* and the source instance has a language tag *TAG* that is removed. An example is *skos:altLabel* in DataGov Statistics, which has a language tag "@en", is translated into *skos:altLabel* in Ordnance Survey and the language tag is removed.

N:1 Value to Value (N:1). A number of source instances of properties P_1, P_2, ..., P_n are translated into a single target instance of property *P'*, and the value of the target instance is computed by means of a function over the values of the source instances. An example of this pattern is that we concatenate the values of properties *foaf:givenName* and *foaf:surname* in DBpedia into property *fb:type.object.name* in Freebase.

Aggregate (Agg). In this pattern, we count the number of source instances of property *P*, which is translated into a target instance of property *Q*. An example is property *fb:metropolitan_transit.transit_system.transit_lines* in Freebase whose values are aggregated into a single value of *dbp:numberOfLines* for each city in DBpedia.

Table 2: Mapping patterns in the LOD Cloud

Code	Source triples	Target triples
RC	?x a C	?x a C'
RP	?x P ?y	?x P' ?y
RCP	?x a C FILTER EXISTS { {?x P ?y} UNION {?y P ?x} }	?x a C'
RCV	?x a C ?x P v	?x a C'
RvP	?x P ?y	?y P' ?x
Rsc	?x P ?y	?x Q ?z ?z P' ?y
DRsc	?x Q ?z ?z P ?y	?x P' ?y
1:1	?x P ?y	?x P' f(?y)
VtU	?x P ?y	?x P' toURI(?y)
UtV	?x P ?y	?x P' toLiteral(?y)
CD	?x P ?y^^TYPE	?x P' ?y^^TYPE'
ALT	?x P ?y	?x P' ?y@TAG
RLT	?x P ?y@TAG	?x P' ?y
N:1	?x P_1 ?v_1 ... ?x P_n ?v_n	?x P' f(?v_1, ..., ?v_n)
Agg	?x P ?y	?x Q count(?y)

Finally, we present a summary of these mapping patterns in Table 2. The first column of this table stands for the code of each pattern; the second and third columns establish the triples to be retrieved in the source and the triples to be constructed in the target using a SPARQL-like notation. Note that properties are represented as P and Q, classes as C, constant values as v, tag languages as *TAG*, and data types as *TYPE*.

3. LODIB GROUNDING

In order to base the LODIB Benchmark on realistic real-world distributions of these mapping patterns, we analyzed 84 data translation examples from the LOD Cloud and counted the occurrences of mapping patterns in these examples. First, we selected different Linked Data sources by exploring the LOD data set catalog maintained on CKAN[1]. The criteria we followed was to choose sources that comprise a great number of *owl:sameAs* links with other Linked Data sources, i.e., more than 25,000. Furthermore, we tried to select sources from the major domains represented in the LOD Cloud. Therefore, the selected Linked Data sources are the following: ACM (RKB Explorer), DBLP (RKB Explorer), Dailymed, Drug Bank, DataGov Statistics, Ordnance Survey, DBpedia, GeoNames, Linked GeoData, LinkedMDB, New York Times, Music Brainz, Sider, GovWILD, ProductDB, and OpenLibrary. Note that, for each domain of the LOD Cloud, there are at least two Linked Data sources that contribute to our statistics except from the domain of user-generated content.

After selecting these sources, we randomly selected 42 examples, each of which comprises a pair of instances that are connected by an *owl:sameAs* link. For each of these examples, we analyzed both directions: one instance is the source and the other instance is the target, and backwards. Therefore, the total number of examples we analyzed was 84. Then, we manually counted the number of mapping patterns that are needed to translate between the representations of the instances (neighboring instances were also considered to detect more complex structural mismatches). These statistics are publicly-available at [22].

In the next step, we computed the averages of our mapping patterns grouped by the pair of source and target data set. To compute them, in some cases, we analyzed the translation of one single instance since the data set of the Linked Data source comprises only a couple of classes, such as Drug Bank or Ordnance Survey. In other cases, we analyzed more than one instance since the data set comprises a large number of classes, such as DBpedia or Freebase.

Table 3 presents the statistics of the mappings patterns that we have found in the LOD Cloud. The two first columns stand for the source and target Linked Data data sets, the following columns contain the averages of each mapping pattern according to the source and the target, i.e., we count the occurrences of mapping patterns in a number of examples and compute the average. Note that, for certain data sets, we analyzed several examples of the same type; therefore, the final numbers of these columns are real numbers (no integers). Finally, the last column contains the total number of instances that we analyzed for each pair of Linked Data data sets.

On the one hand, Rename Class and Rename Property mapping patterns appear in the vast majority of the analyzed examples, since these patterns are very common in practice. On the other hand, there are some patterns that are not so common, e.g., Value to URI and URI to Value patterns appear only once in all analyzed examples (between DBpedia and Drug Bank). Table 4 presents the average occurrences of the LODIB mapping patterns over all analyzed examples.

4. LODIB DESIGN

Based on the previously described statistics, we have designed the LODIB Benchmark. The benchmark consists of three different source data sets that need to be translated by the system under test into a single target vocabulary. The topic of the data sets is the same e-commerce data set that we already used for the BSBM Benchmark [6]. The data sets describe products, reviews, people and some more lightweight classes, such as product price using different source vocabularies. For translation from the representation of an instance in the source data sets to the target vocabulary, data translation systems need to apply several of the presented mapping patterns. The descriptions of these data sets are publicly-available at the LODIB homepage [22].

These data sets take the previously computed averages of Table 4 into account by multiplying them by a constant (11), and divided each one by another constant (3, the total number of data translation tasks, i.e., from each source data set to the target data set). As a result, each of the three data translation tasks comprises a number of mapping patterns, and we present the numbers in Table 5, in which the total number of mapping patterns for each task is 18.

[1] `http://thedatahub.org/group/lodcloud`

Table 3: Mapping patterns in Linked Data sources

Source	**Target**	**RC**	**RP**	**RCP**	**RCV**	**RvP**	**Rsc**	**DRsc**	**1:1**	**VtU**	**UtV**	**CD**	**ALT**	**RLT**	**N:1**	**Agg**	**Total**
ACM	DBLP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Dailymed	Drug Bank	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
DataGov Stats	Ordnance Survey	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1
DBLP	ACM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
DBpedia	Freebase	2.14	8.57	0.64	0.00	2.21	2.29	0.00	0.57	0.00	0.00	1.14	0.00	0.00	0.14	0.07	14
DBpedia	Geonames	1.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3
DBpedia	Linked GeoData	1.00	3.50	0.00	0.00	0.00	0.00	1.50	0.13	0.00	0.00	2.25	0.00	0.00	0.00	0.00	8
DBpedia	LinkedMDB	1.00	5.50	0.33	0.00	0.33	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6
DBpedia	Drug Bank	1.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00	3.00	0.00	0.00	1
DBpedia	New York Times	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1
DBpedia	Music Brainz	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Drug Bank	DBpedia	1.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00	3.00	0.00	1.00	0.00	1
Drug Bank	Freebase	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Drug Bank	Sider	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Drug Bank	Dailymed	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Freebase	DBpedia	2.14	8.57	0.29	0.07	2.21	0.00	2.29	0.79	0.00	0.00	1.14	0.00	0.00	0.00	0.14	14
Freebase	GovWILD	1.00	4.50	0.00	0.00	0.00	0.00	2.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2
Freebase	Drug Bank	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
GeoNames	DBpedia	1.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
GovWILD	Freebase	1.00	4.50	0.00	0.50	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	2
Linked GeoData	DBpedia	1.00	3.50	0.88	0.75	0.00	1.50	0.00	0.13	0.00	0.00	2.25	0.00	0.00	0.00	0.00	8
LinkedMDB	DBpedia	1.00	5.50	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6
Music Brainz	DBpedia	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
New York Times	DBpedia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
OpenLibrary	ProductDB	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Ordnance Survey	DataGov Stats	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1
ProductDB	OpenLibrary	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Sider	Drug Bank	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1

Table 4: Average occurrences of the mapping patterns

RC	RP	RCP	RCV	RvP	Rsc	DRsc	1:1	VtU	UtV	CD	ALT	RLT	N:1	Agg
0.87	2.01	0.08	0.05	0.18	0.24	0.24	0.30	0.04	0.04	0.24	0.14	0.14	0.09	0.01

Table 5: Number of mapping patterns in the data translation tasks

Task	RC	RP	RCP	RCV	RvP	Rsc	DRsc	1:1	VtU	UtV	CD	ALT	RLT	N:1	Agg
Task 1	3	7	1	0	1	1	1	1	0	1	1	0	1	0	0
Task 2	3	7	0	1	0	1	1	1	1	0	1	1	0	1	0
Task 3	3	7	0	0	1	1	1	1	0	0	1	1	1	0	1

We have implemented a data generator to populate and scale the three source data sets that we have specified in the previous section, which is publicly-available at the LODIB homepage [22]. In the data generator, we defined a number of data generation rules, and the generated data are scaled based on the number of product instances that each data set contains. In our implementation, we use an extension of the language used in [8], which allows to define particular value generation rules for basic types, such as *xsd:string* or *xsd:date*. In addition, missing properties often occurs in the context of the Web of Data, therefore, we also provide 44 statistical distributions in our implementation to randomly select distribute properties, including Uniform, Normal, Exponential, Zipf, Pareto and empirical distributions, to mention a few.

In this section, we provide examples on how a data translation system needs to translate the data from the source to the target vocabulary regarding the mapping patterns in the three data translation tasks. Specifically, Figure 1 presents a number of source triples that are translated into a number of target triples. Note that we use prefixes *src1:*, *src2:*, *src3:* and *tgt:* for referring to the source data sets and the single target vocabularies of the data sets; and *src1-data:*, *src2:-data*, *src3:-data* and *tgt:-data* for referring to the source and target data. These examples are the following:

Rename Class (RC). Class *src1:Product* needs to be renamed into class *tgt:Product*, e.g., *src1-data:Canon-Ixus-20010* product instance.

Rename Property (RP). Property *src1:name* needs to be renamed into property *rdfs:label*, e.g., the name of *src1-data:Canon-Ixus-20010* product instance.

Rename Class based on Property (RCP). In this case, class *src1:Person* needs to be renamed into class *tgt:Reviewer*, if and only if, property *src1:author* exists, e.g., *src1-data:Smith-W* person instance.

Rename Class based on Value (RCV). In this example, class *src2:Product* needs to be renamed into class *tgt:OutdatedProduct*, if and only if, property *src2:outdated* exists and has value "Yes", e.g., *src2-data:HTC-Wildfire-S* product instance.

Reverse Property (RvP). In this example, property *src1:author* is reversed into property *tgt:author*, e.g., *src1-data:Review-CI-001* review instance and *src1-data:Smith-W* person instance are related and reversed in the target.

Resourcesify (Rsc). Property *src1:birthDate* needs to be renamed into property *tgt:birthDate* and a new target instance of property *tgt:birth* is needed, e.g., the date of birth of *src1-data:Smith-W* person instance.

Deresourcesify (DRsc). Property *src2:revText* needs to be renamed into property *tgt:text*, if and only if, the instance of property *src2:revText* is related to another source instance of property *src2:hasText*, e.g., the text of *src2-data:Review-HTC-W-S* review instance.

1:1 Value to Value (1:1). Property *src2:price* needs to be renamed into property *tgt:productPrice*, and the value must be transformed by means of function *usDollarsToEuros*, since the source price is represented in US dollars and the target in Euros, e.g., the price of *src2-data:HTC-Wildfire-S* product instance.

Value to URI (VtU). In this example, we need to rename property *src1:personHomepage* into property *tgt:personHomepage*, and the values of the source instances are transformed into URIs in the target, e.g., the homepage of *src1-data:Smith-W* person instance.

URI to Value (UtV). In this example, we need to rename property *src2:productHomepage* into property *tgt:productHomepage*, and the URIs of the source instances are transformed into values in the target, e.g., the homepage of *src2-data:HTC-Wildfire-S* product instance.

Change Datatype (CD). Property *dc:date* in the first source needs to be translated into *dc:date*, and its type is transformed from *xsd:string* into *xsd:date*, e.g., the date of *src1-data:Review-CI-001* review instance.

Add Language Tag (ALT). property *src2:mini-cv* needs to be renamed into property *tgt:bio* and a new tag language "@en" is added in the target, e.g., the CV of *src2-data:Doe-J* person instance.

Remove Language Tag (RLT). property *src1:revText* needs to be renamed into property *tgt:text* and the tag language of the source is removed, e.g., the text of *src1-data:Review-CI-001* review instance.

N:1 Value to Value (N:1). properties *foaf:firstName* and *foaf:surname* in the second source need to be translated into property *tgt:name*, and their values are concatenated to compose the target value, e.g., the first name and surname of *src2-data:Doe-J* person instance.

```
src1-data:Canon-Ixus-20010
    a                   src1:Product ;
    src1:name           "Canon Ixus"^^xsd:string .
src2-data:HTC-Wildfire-S
    a                   src2:Product ;
    src2:outdated       "Yes"^^xsd:string ;
    src2:price          "199.99"^^xsd:double ;
    src2:productHomepage <htc.com/> .
src3-data:VPCS
    a                   src3:Product ;
    src3:hasReview      src3-data:Review-VPCS-01 ;
    src3:hasReview      src3-data:Review-VPCS-02 ;
    src3:hasReview      src3-data:Review-VPCS-03 .
src1-data:Review-CI-001
    a                   src1:Review ;
    src1:author         src1-data:Smith-W ;
    dc:date             "01/10/2011"^^xsd:string ;
    src1:revText        "This camera is awesome!"@en .
src2-data:Review-HTC-W-S
    a                   src2:Review ;
    src2:hasText        src2-data:Review-HTC-W-S-Text .
src2-data:Review-HTC-W-S-Text
    a                   src2:ReviewText ;
    src2:revText        "Great phone"^^xsd:string .
src1-data:Smith-W
    a                   src1:Person ;
    src1:birthDate      "06/07/1979"^^xsd:date ;
    src1:personHomepage "wsmith.org"^^xsd:string .
src2-data:Doe-J
    a                   src2:Person ;
    src2:mini-cv        "Born in the US."^^xsd:string ;
    foaf:firstName      "John"^^xsd:string ;
    foaf:surname        "Doe"^^xsd:string .
```

(a) Source triples

```
src1-data:Canon-Ixus-20010
    a                   tgt:Product ;
    rdfs:label          "Canon Ixus"^^xsd:string .
src2-data:HTC-Wildfire-S
    a                   tgt:OutdatedProduct ;
    tgt:productPrice    "152.59"^^xsd:double ;
    tgt:productHomepage "htc.com/"^^xsd:string .

src3-data:VPCS
    a                   tgt:Product ;
    tgt:totalReviews    "3"^^xsd:integer .


src1-data:Review-CI-001
    a                   tgt:Review ;
    dc:date             "01/10/2011"^^xsd:date ;
    tgt:text            "This camera is awesome!" .

src2-data:Review-HTC-W-S
    a                   tgt:Review ;
    tgt:text            "Great phone"^^xsd:string .

src1-data:Smith-W
    a                   tgt:Reviewer ;
    tgt:author          src1-data:Review-CI-001 ;
    tgt:birthDate       tgt-data:Smith-W-BirthDate .
tgt-data:Smith-W-BirthDate
    a                   tgt:Birth ;
    tgt:birthDate       "06/07/1979"^^xsd:date ;
    tgt:personHomepage  <wsmith.org> .
src2-data:Doe-J
    a                   tgt:Person ;
    tgt:bio             "Born in the US."@en ;
    tgt:name            "John Doe"^^xsd:string .
```

(b) Target triples

Figure 1: Sample data translation tasks.

Aggregate (Agg). we count the number of instances of source property *src3:hasReview*, and this number needs to be translated as the value of property *tgt:totalReviews*, e.g., the reviews of *src3-data:VPCS* product instance.

5. EXPERIMENTS

The LODIB benchmark can be used to measure two performance dimensions of a data translation system. For one thing we state the expressivity of the data translation system, that is, the number of mapping patterns that can be expressed in each system. Secondly we measure the performance by taking the time to translate all source data sets to the target representation. For our benchmark experiment, we generated data sets in N-Triples format containing 25, 50, 75 and 100 million triples. For each data translation system and data set the time is measured starting with reading the input data set file and ending when the output data set has been completely serialized to one or more N-Triples files.

We have applied the benchmark to test the performance of two data translation systems:

Mosto It is a tool to automatically generate executable mappings amongst semantic-web ontologies [20]. It is based on an algorithm that relies on constraints such as *rdfs:domain* of the source and target ontologies to be integrated, and a number of 1-to-1 correspondences between TBox ontology entities [19]. Mosto tool also allows to run these automatically generated executable mappings using several semantic-web technologies, such as Jena TDB, Jena SDB, or Oracle 11g. For our tests we advised Mosto to generate (Jena specific) SPARQL Construct queries. The data sets were translated using these generated queries and Jena TDB (version 0.8.10).

LDIF It is an ETL like component for integrating data from Linked Open Data sources [24]. LDIF's integration pipeline includes one module for vocabulary mapping, which executes mappings expressed in the R2R [7] mapping language. All the R2R mappings were written by hand. LDIF supports different run time profiles that apply to different work loads. For the smaller data sets we used the in-memory profile, in which all the data is stored in memory. For the 100M data set we executed the Hadoop version, which was run in *single-node* mode (pseudo-distributed) on the benchmarking machine as the in-memory version was not able to process this use case.

To allow other researchers to reproduce our results, the con-

figuration and all used mappings for Mosto and LDIF are publicly-available at the LODIB homepage [22]. To set the results of these two systems into context of the more popular tools in the Linked Data space, we compared the performance of both systems with the SPARQL 1.1 performance of the Jena TDB RDF store (version 0.8.10). All the mappings for Jena TDB were expressed as SPARQL 1.1 Construct queries, which were manually written by ourselves. For loading the source data sets we used the more efficient *tdbloader2*, which also generates data set statistics that are used by the TDB optimizer.

Table 6 gives an overview of the expressivity of the data translation systems. All mapping patterns are expressable in SPARQL 1.1, so all the mappings are actually executed on Jena TDB. The current implementation of the Mosto tool generates Jena-specific SPARQL Construct queries, which could, in general, cover all the mapping patterns. However, the goal of Mosto tool is to automatically generate SPARQL Construct queries by means of constraints and correspondences without user intervention, therefore, the meaning of a checkmark in Table 6 is that it was able to automatically generate executable mappings from the source and target data sets and a number of correspondences amongst them. Note that Mosto tool is not able to deal with RCP and RCV mapping patterns since it does not allow the renaming of classes based on conditional properties and/or values. Furthermore, it does not support Agg mapping pattern since it does not allow to aggregate/count properties. In R2R it is not possible to express aggregates, therefore no aggregation mapping was executed on LDIF. In order to check if the source data has been correctly and fully translated, we developed a validation tool that examines if the source data is represented correctly in the target data set. Using the validation tool, we verified that all three systems produce proper results.

To compare the performance and the scaling behaviour of the systems we have run the benchmark on an Intel i7 950 (4 cores, 3.07GHz, 1 x SATA HDD) machine with 24GB of RAM running Ubuntu 10.04.

Table 7 summarizes the overall runtimes for each mapping system and use case. Since Mosto and R2R were not able to express all mapping patterns, we created three groups: 1) one that did not execute the RCV, RCP and AGG mappings, 2) one without the AGG mapping and 3) one executing the full set of mappings. The results show that Mosto and Jena TDB have – as expected – similar runtime performance because Mosto internally uses Jena TDB. LDIF on the other hand is about twice as fast on the smallest data set and about three times as fast for the largest data set compared to Jena TDB and Mosto. One reason for the differences could be that LDIF highly parallelizes its work load, both in the in-memory as well as the Hadoop version.

6. RELATED WORK

The most closely related benchmarks are STBenchmark [1] and DTSBench [21]. Alexe et al. [1] devised STBenchmark, a benchmark that is used to test data translation systems in the context of nested relational models. This benchmark provides eleven patterns that occur frequently in the information integration context. Unfortunately, this benchmark is not suitable in our context since semantic-web technologies have a number of inherent differences with respect to nested relational models [2, 14, 15, 25].

Rivero et al. [21] devised DTSBench, a benchmark to test data translation systems in the context of semantic-web technologies that provides seven data translation patterns. Furthermore, it provides seven parameters that allow to create a variety of synthetic, domain-independent data translation tasks to test such systems. This benchmark is suitable to test data translation amongst Linked Data sources, however, the patterns that it provides are inspired from the ontology evolution and information integration contexts, not the Linked Data context. Therefore, it allows to generate synthetic tasks based on these patterns, but not real-world Linked Data translation tasks.

There are other benchmarks in the literature that are suitable to test semantic-web technologies. However, they cannot be applied to our context, since none of them focuses on data translation problems, i.e., they do not provide source and target data sets and a number of queries to perform data translation. Furthermore, these benchmarks focus mainly on Select SPARQL queries, which are not suitable to perform data translation, instead of on Construct SPARQL queries.

Guo et al. [12] presented LUBM, a benchmark to compare systems that support semantic-web technologies, which provides a single ontology, a data generator algorithm that allows to create scalable synthetic data, and fourteen SPARQL queries of the Select type. Wu et al. [26] presented the experience of the authors when implementing an inference engine for Oracle. Bizer and Schultz [6] presented BSBM, a benchmark to compare the performance of SPARQL queries using native RDF stores and SPARQL-to-SQL query rewriters. Schmidt et al. [23] presented SP^{2}Bench, a benchmark to test SPARQL query management systems, which comprises both a data generator and a set of benchmark queries in SPARQL.

7. CONCLUSIONS

Linked Data sources try to reuse as much existing vocabularies as possible in order to ease the integration of data from multiple sources. Other data sources use completely proprietary vocabularies to represent their content or use a mixture of common terms and proprietary terms. Due to these facts, there exists heterogeneity amongst vocabularies in the context of Linked Data. Data translation, which relies on executable mappings and consists of exchanging data from a source data set to a target data set, helps solve these heterogeneity problems.

In this paper, we presented LODIB, a benchmark to test data translation systems in the context of Linked Data. Our benchmark provides a catalogue of fifteen data translation patterns, each of which is a common data translation problem. Furthermore, we analyzed 84 random examples of data translation in the LOD Cloud and we studied the distribution of the patterns in these examples. Taking these results into account, we devised three source and one target data set based on the e-commerce domain that reflect the mapping pattern distribution. Each source data set comprises one data translation task.

Table 6: Expressivity of the mapping systems

	RC	RP	RCP	RCV	RvP	Rsc	DRsc	1:1	VtU	UtV	CD	ALT	RLT	N:1	Agg
Mosto queries	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
SPARQL 1.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
R2R	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table 7: Runtimes of the mapping systems for each use case (in seconds)

	25M	50M	75M	100M
Mosto SPARQL queries / Jena TDB[1]	3,121	7,308	10,622	15,763
R2R / LDIF[1]	1,506	2,803	4,482	*5,718
SPARQL 1.1 / Jena TDB[1]	2,720	6,418	10,481	16,548
R2R / LDIF[2]	1,485	2,950	4,715	*5,784
SPARQL 1.1 / Jena TDB[2]	2,839	6,508	12,386	19,499
SPARQL 1.1 / Jena TDB	2,925	6,858	12,774	20,630

* *Hadoop version of LDIF as single node cluster. Out of memory for in-memory version.*
[1] *without RCP, RCV and AGG mappings*
[2] *without AGG mapping*

Current benchmarks concerning data translation focus on nested relational models, which is not suitable for our context since semantic-web technologies have a number of inherent differences with respect to these models, or in the general context of semantic-web technologies. To the best of our knowledge, LODIB is the first benchmark that is based on real-world distribution of data translation patterns in the LOD Cloud, and that is specifically tailored towards the Linked Data context.

In this paper, we compared three data translation systems, Mosto, SPARQL 1.1/Jena TDB and R2R, by scaling the three data translation tasks. In this context, Mosto is able to deal with 12 out of the 15 mapping patterns described in this paper, SPARQL 1.1/Jena TDB deals with 15 out of 15, and R2R deals with 14 out of 15. Furthermore, the results show that R2R outperforms both Mosto and SPARQL 1.1/Jena TDB data translation systems when performing the three data translation tasks. Our empirical study has shown that, to translate data amongst data sets in the LOD Cloud, there is only needed a small set of simple mapping patterns. In this context, the fifteen mapping patterns identified in this paper were enough to cover the vast majority of data translation problems when integrating these data sets.

As the Web of Data grows, the task of translating data amongst data sets moves into the focus. We hope that LODIB benchmark will be considered useful by the developers of the currently existing Linked Data translation systems as well as the systems to come. More information about LODIB is publicly-available at the homepage [22], such as the exact specification of the benchmark data sets, the data generator, examples of the mapping patterns, or the statistics about these mappings that we found in the LOD Cloud.

Acknowledgments

Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E), and partially financed through funds received from the European Community's Seventh Framework Programme (FP7) under Grant Agreement No. 256975 (LATC) and Grant Agreement No. 257943 (LOD2).

8. REFERENCES

[1] B. Alexe, W. C. Tan, and Y. Velegrakis. STBenchmark: Towards a benchmark for mapping systems. *PVLDB*, 1(1):230–244, 2008.
[2] R. Angles and C. Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1), 2008.
[3] M. Arenas and L. Libkin. Xml data exchange: Consistency and query answering. *J. ACM*, 55(2), 2008.
[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
[5] C. Bizer, A. Jentzsch, and R. Cyganiak. State of the LOD cloud. Available at: `http://www4.wiwiss.fu-berlin.de/lodcloud/state/#terms`, 2011.
[6] C. Bizer and A. Schultz. The Berlin SPARQL benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24, 2009.
[7] C. Bizer and A. Schultz. The R2R framework: Publishing and discovering mappings on the Web. In *1st International Workshop on Consuming Linked Data (COLD)*, 2010.
[8] D. Blum and S. Cohen. Grr: Generating random RDF. In *ESWC (2)*, pages 16–30, 2011.
[9] J. Euzenat, A. Polleres, and F. Scharffe. Processing ontology alignments with SPARQL. In *CISIS*, pages 913–917, 2008.
[10] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
[11] A. Fuxman, M. A. Hernández, C. T. H. Ho, R. J. Miller, P. Papotti, and L. Popa. Nested mappings:

Schema mapping reloaded. In *VLDB*, pages 67–78, 2006.

[12] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[13] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[14] B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between OWL and relational databases. *J. Web Sem.*, 7(2):74–89, 2009.

[15] N. F. Noy and M. C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, 2004.

[16] F. S. Parreiras, S. Staab, S. Schenk, and A. Winter. Model driven specification of ontology translations. In *ER*, pages 484–497, 2008.

[17] A. Polleres, F. Scharffe, and R. Schindlauer. SPARQL++ for mapping between RDF vocabularies. In *ODBASE*, pages 878–896, 2007.

[18] H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. In *ODBASE*, pages 832–849, 2007.

[19] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Generating SPARQL executable mappings to integrate ontologies. In *ER*, pages 118–131, 2011.

[20] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Mosto: Generating SPARQL executable mappings between ontologies. In *ER Workshops*, pages 345–348, 2011.

[21] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. On benchmarking data translation systems for semantic-web ontologies. In *CIKM*, pages 1613–1618, 2011.

[22] C. R. Rivero, A. Schultz, and C. Bizer. Linked Open Data Integration Benchmark (LODIB) specification. Available at: `http://www4.wiwiss.fu-berlin.de/bizer/lodib/`, 2012.

[23] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. SP^{2}Bench: A SPARQL performance benchmark. In *ICDE*, pages 222–233, 2009.

[24] A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. LDIF - Linked Data integration framework. In *2nd International Workshop on Consuming Linked Data (COLD)*, 2011.

[25] M. Uschold and M. Grüninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[26] Z. Wu, G. Eadon, S. Das, E. I. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. Implementing an inference engine for RDFS/OWL constructs and user-defined rules in oracle. In *ICDE*, pages 1239–1248, 2008.