How Redundant Is It? - An Empirical Analysis on Linked Datasets

Honghan Wu[1], Boris Villazon-Terrazas[2], Jeff Z. Pan[1], and Jose Manuel Gomez-Perez[2]

[1] Department of Computing Science, University of Aberdeen, UK
[2] iSOCO, Intelligent Software Components S.A., Spain

Abstract. Data redundancy resides in most, if not all, information systems. Linked Data is no exception. Existing approaches try to avoid data redundancies by proposing compression techniques or succinct data structures. However, data redundancies in Linked Data are useful sometimes, e.g., ontology based data access can make use of A-Box redundancies to avoid unnecessary query rewritings. Either you want to avoid it or make use of it, a good understanding about data redundancies will facilitate your task, e.g., identify the exact redundant parts which could be utilised or choose most effective techniques to compress a particular dataset. Unfortunately, little effort has been put on making the data redundancy explicit to data users. In this paper, we introduce a systematic categorisation for Linked Data redundancy, and propose a graph pattern based approach for efficient analysis. Analysis results on representative datasets lead to a main conclusion, that is redundant-aware techniques are demanded.

1 Introduction

Data redundancy has different meanings in different contexts. In database community, data redundancy means that the same piece of data is stored in *multiple places* in a database system, while in information theory it means the *wasted space* used to transmit certain data. In this paper, we focus on data redundancy in Linked Data, which means the *wasted space* used to represent certain meaning in either stand-alone data space or the Web of Data environment.

Depending on the scenarios, data redundancy in Linked Data might have different effects to data consumption tasks. In many cases, Linked Data redundancy might be unwelcome. For example, for storage or exchange purpose, the redundancy will cause unnecessary resource consumptions, e.g., more disk spaces or longer time to download a dump file from the Web. However, in other occasions, the redundancy of the data can be utilised to facilitate the task on hand. For example, in Ontology Based Data Access (OBDA) tasks, A-Box redundancy can be utilised to avoid unnecessary query rewritings so that the efficiency of the system can be improved.

Redundancies in Linked Data have effects on a wide range of applications including data publishing, query answering (in SPARQL endpoints), OBDA, and ontology reasoning. Existing work [8, 3, 5] either focuses on RDF data compression or provides succinct data structure for data access. Little attention has been put on making the data redundancy explicit and available to Linked Data stakeholders. For data consumers, a good knowledge about data redundancies in interested datasets will help users either make use of them efficiently or choose the best technique to avoid them. For data publishers, such knowledge will guide them in making the right decisions like reusing the right vocabularies, linking to right datasets or sometimes not linking at all. More generally, in the linked open data environment, the data redundancies residing in multiple sources are the results of distributed and autonomous social efforts, which are valuable sources for studying the properties and characteristics of the Data Web, e.g. analysing the trustworthiness of statements.

To sum up, a better understanding about data redundancies in linked data will help us know more and deeper about the Data Web, which will in turn facilitate more effective and efficient exploitations on the linked data. In this paper, we focus on revealing the redundancies in Linked Data by both a qualitative analysis of systematic redundancy categorisation and an quantitative analysis on various datasets covering different domains.

The rest of the paper is organised as follows. In section 2, we briefly introduce related work. Section 3 gives an discussion about the categorisation of redundancies in Linked Data. Section 4 proposes our redundancy analysis methodology which covers two different dimensions. Section 5 gives the detailed analysis results on real world datasets. Finally, we conclude the work in section 6.

2 Related Work

RDF compression techniques have been proposed to eliminate data redundancies. Such as RDF serialisations techniques, i.e., HDT serialisation [5], lean graphs [7] and K2-triples [1] can be used to reduce file size. Another approach is based on logical compression, such as the rule-based RDF compression [8], which can be used to substantially reduce the number of triples in an RDF document. Inspired by the HDT approach, Curé et al. proposed WaterFowl [3] as a succinct data structure for RDF data. The OWL *sameAs* network was studied by Ding et al. [4]. The implication of *sameAs* links was raised but not studied. Halpin et al. [6] also proposed a way to analyse identity in linked data based on *sameAs* links. As we mentioned, at the time of writting and to the best of our knowledge none of existing work has made the data redundancies explicit and available to stakeholders of Linked Data, which is the main focus of this paper.

3 Linked Data Redundancy Categorisation

Given the fact that most Linked Datasets are represented in RDF data model, in the first part of this section, we study RDF data redundancy and point out the main focuses of this paper. In addition to RDF representation, the other important characteristic of Linked Data is that it is *linked*. What are the aspects of being *linked* relevant to the data redundancy, and how can they be effecting it? In the second subsection we try to answer these questions.

3.1 Redundancy in RDF Data

From the data model level, an RDF dataset is essentially a set of RDF graphs that contain a set of triples. To share or consume an RDF dataset, e.g., for storage, transmission

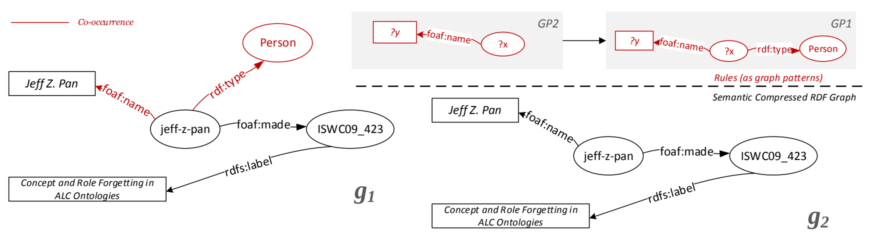


Fig. 1. Original v.s. Semantically Compressed

or query-answering, it needs to be represented in the second level, i.e., the serialisation level, where it has to be serialised as a sequence of bits. In this level, an RDF dataset usually takes the form of textual or binary files by using predefined syntaxes, e.g., RDF/XML, N-Triples or even sophisticated compression format (e.g. HDT [5]). The redundancies of RDF data reside in both levels of RDF data representation, i.e., data model level and serialisation level.

In the data model level, the size of data can be calculated by the number of triples. Hence, in this level, the data redundancy exists if less triples can be used to represent the same semantic meanings of the original data. In the serialisation level, the data is represented as a sequence of bits. Given a fix set of triples, one serialisation is said to be more redundant than the other if it uses more bits than its counterpart.

In [9], we proposed a fine-grained categorisation of RDF data redundancies. Table 1 illustrates such categorisation of RDF redundancies and also puts it in the dimension of RDF representation levels. In this paper we mainly focus on the *semantic redundancy* and the second type of syntactic redundancy, i.e., the *inter-structural redundancy*. Both are highlighted with a grey background in Table 1. Semantic redundancy is selected because it can be generated or removed by the T-Box axioms of a dataset. Hence, it is of interest to most of the Linked Data consumption tasks like inference computation and ontology based data access. Syntactic redundancy is also important to data consumption because a concise serialisation is beneficial not only to data transmission but also to query answering tasks [5]. In this category, a recent study [9] points out that most existing compression techniques, e.g. [5], cannot deal with the inter-structural one [9]. Hence, it is particularly interesting to analyse inter-structural redundancies in Linked Open Data. The rest of this section will discuss the redundancy types to be analysed in this paper.

Table 1. RDF Data Redundancy Categorisation

Types	Semantic Redundancy	Syntactic Redundancy		Symbolic Redundancy
		Intra-structural	Inter-structural	
Data Model Level	✓	-	-	-
Serialisation Level	-	✓	✓	✓

Semantic Redundancy An RDF dataset is said to be semantically redundant if some triples can be removed without leading to any changes in its meaning. In most cases the removal of these triples requires additional rules to be added in the dataset so that it is possible to re-generate the removed triples when needed. The usual form of such rules is the T-Box, i.e., the concept level statements in an ontology. For example, in Fig. 1, we have an RDF graph of g_1. In the FOAF ontology [3], there is a rule of <$foaf{:}name, rdfs{:}domain, foaf{:}Person$>. Based on the $rdfs2$ rule in the RDF specification, the type assertion in g_1, i.e. <jeff-z-pan, $rdf{:}type, foaf{:}Person$>, can be removed, given the presence of <jeff-z-pan, $foaf{:}name, JeffZ.Pan$>.

In a more general perspective, the semantic redundancy can be identified by co-occurrences of triple patterns (cf. the red part of g_1 in Fig. 1). Hence, it is not necessary to restrain the ability of semantic redundancy identification by the limitation of T-Box rules in representing triple pattern co-occurrences. In this regard, Joshi et al. [8] applied association rule mining approach to identify the co-occurrences in terms of association rules. In this paper, we apply a graph pattern based rule system to represent the semantic redundancies. For example, in Fig. 1, g_2 has less triples than g_1 but the two are semantically equivalent. The redundancy in g_1 is represented by the graph pattern rule in the upper part of g_2. Obviously, graph pattern based rules can represent more complex triple co-occurrences than T-Box rules.

Inter-structure Syntactic Redundancy As shown in Table 1, the other two types of redundancies, i.e., syntactic and symbolic ones, both resides in the serialisation level. Their volume can be evaluated using the number of bits used in the serialisation. To separate the two, we can use a simple formula $|F| = n \times r$, where F is the serialisation file, n is the number of resource occurrences and r is the average bits needed to represent a resource. The syntactic redundancies reside in the component of n, while the symbolic ones are in r.

Most existing serialisation approaches apply syntaxes to reduce n. The idea is to group triples by subjects or objects so that the multiple occurrences of the same resource only need to be serialised once. For example, the RDF/XML serialisation standard provides abbreviation and striping syntaxes. However, such syntaxes only work on concrete graph structures. Similar graph structures (i.e. graph patterns) which repeatedly occur in the data are not taken into account. In the following example, the graph pattern GP has two instances of $Inst_1$ and $Inst_2$. The structure of GP appears twice in both instances. This means that each of the two predicate resources of GP, i.e. $foaf{:}name$ and $foaf{:}made$, has two occurrences in its instances, which is avoidable when GP structure (stored wherever) is referred instead of duplicated in both instances.

$$\begin{aligned} GP &: <?U, foaf{:}name, ?Y>, <?U, foaf{:}made, ?Z> \\ Inst_1 &: <\textit{jeff-z-pan}, foaf{:}name, \text{Jeff Z. Pan}>, <?U, foaf{:}made, ISWC09\_423> \\ Inst_2 &: <jose, foaf{:}name, \text{Jose Manuel Gomez Perez}>, <?U, foaf{:}made, ISWC13\_1xx> \end{aligned}$$

We use the term of *intra-structural redundancy* to denote the unnecessary resource occurrences in concrete graph structure, while the term of *inter-structural redundancy* is used for the unnecessary resource occurrences of graph patterns in its instances. Most of existing serialisation approaches do not provide facilities to identify graph patterns. Hence, they leave the inter-structural redundancy untouched.

[3] `http://xmlns.com/foaf/spec/`

3.2 Redundancy in Linked Data

One of the most important characteristics of Linked Data is its *linking* capability, which is to create connections from one information source to the other. The connection can be created in the concept level, where individuals of one dataset are described using concepts from another dataset or vocabulary. We denote this type of connections as *T-Box Reuse*. The second type of connections is the linkage in individual level, where individuals in one dataset are specified to be the same as their counterparts in the other, e.g., by using $owl{:}sameAs$ assertions. This type of connections is called as *A-Box Linkage* in this paper. Both types of connections have implications in changing the semantics of the original dataset. These implications might change the redundancies of the dataset in question. In rest part of this subsection, we briefly discuss the possible changes on data redundancies caused by *T-Box Reuse*, and leave the *A-Box Linkage* for future work.

T-Box Reuse Essentially, T-Box Reuse will bring new rules to the original dataset. These rules are a subset of the materialised axioms of the reused T-Box, which are applicable to individuals. These new rules can infer a set of new triples [4] to the reusing RDF dataset.

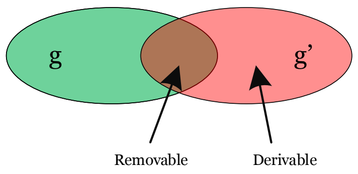


Fig. 2. Data Redundancies affected by T-Box Reuse

Fig. 2 illustrates a typical case of such inference results. In Fig. 2, the original RDF dataset, labelled as g, is denoted by the green oval, and the new set of inferred triples is denoted by the pink oval with a label of g'. Depending on whether they are in the overlap with existing triples, new triples can be divided into two parts. The two parts will lead to two different consequences to the redundancies of the original data. The overlap part, labelled $removable$ in Fig. 2, contains those triples in g, which can be inferred by new rules as well. This means that they are turned to be semantically redundant by the reused T-Box. As a consequence, the dataset turns to have more redundancies. On the contrary, the other part ($g' \setminus removable$), labelled as $derivable$ in Fig. 2, will decrease the data redundancy according to the definition of data compression ratio [5]. Specifically, in this scenario, data compression ratio can be calculated by $\frac{|g|+|derivable|}{|g|}$, in which the bigger $|derivable|$ component becomes, the larger the compression ratio will be.

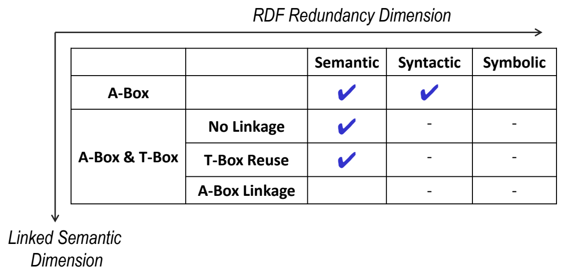
[4] Some triples inferred by new rules might be inferred by the dataset's own T-Box as well. To make discussion easier, we use the term *new triples* to denote those triples which can NOT be inferred by the dataset's own T-Box but the new rules.

[5] `http://en.wikipedia.org/wiki/Data_compression_ratio`

To sum up, $removable$ and $derivable$ affect the data redundancies in opposite ways. Analyses on them will reveal the effects on data redundancy of concept-level connections between Linked Data.

4 Two Dimension Analysis

According to above discussions, the redundancies in Linked Data can be analysed from two dimensions (cf. Fig. 3). The first dimension is that of the RDF data redundancy, where the focus is to reveal different categories of redundancies from data model level to serialisation level. The second dimension is from the *linked* semantic point of view, where the focus is to analyse the data redundancy based on different types of semantics. The *A-Box* semantics is to analyse the data redundancy only from data level without any T-Box axioms. The other three types are considering both data and T-Box information. The *No Linkage* semantics is to do the analysis based on the dataset's main T-Box. The main T-Box is the one most used for describing individuals in the data, which is either defined by the data provider or reused from other sources. In our analysis, we manually identify such main T-Box for each dataset. It is not necessary that every dataset has a main T-Box. In such cases, this analysis is not applicable. The *T-Box Reuse* semantics is the type of analysis focusing on redundancy changes caused by concept level connections, i.e., reusing T-Box, while the *A-Box Linkage* is to reveal the changes of data redundancies caused by the individual level connections.


RDF Redundancy Dimension
Linked Semantic Dimension

		Semantic	Syntactic	Symbolic
A-Box		✓	✓	
A-Box & T-Box	No Linkage	✓	-	-
	T-Box Reuse	✓	-	-
	A-Box Linkage		-	-

Fig. 3. Two Dimension Analysis on Linked Data Redundancy

In the matrix of Fig. 3, there are total 6 valid types of analyses. In this paper, we focus on four of them, which are marked with ticks in the figure. For *A-Box* semantics, we propose a graph pattern based approach to reveal the semantic and syntactic (inter-structure only) redundancies. Based on the graph pattern approach, we propose a virtual materialisation approach to analyse data redundancies in *No Linkage* and *T-Box Reuse* semantics. The other two types of analyses are left for future work.

4.1 Graph Pattern Based Analysis Method

As discussed in section 3.1, we focus on semantic redundancy and the inter-structural syntactic redundancy, both of which will be benefited from the ability to identify frequent graph patterns. For semantic redundancy, identified graph patterns can be used to identify possible rules for removing redundant triples. Combined with the knowledge of instance numbers of these graph patterns, these rules can be used to calculate the volume of semantic redundancy as number of removable triples. Similarly, for inter-structural redundancy, the structure of graph patterns and their instances numbers can give us a way to calculate the volume of syntactic redundancy in the data. In this subsection, we describe an entity description pattern based approach for redundancy analysis.

In an RDF graph, we call its non-literal nodes as entities. For an entity e in an RDF graph G, we can get a data block for it by extracting triples in G each of which has e as its subject or object. We call such kind of data blocks as Entity Description Blocks (EDBs for short).

For an EDB, it can be summarised by a notion of entity description pattern which is defined in Definition 1. *EDP*, the short name for entity description pattern, is the building block of our analysis approach.

Definition 1. *(Entity Description Pattern) Given an entity description block B_e, its description pattern is a tuple $P_e = (C_e, A_e, R_e, V_e)$, where*

- $C_e = \{c_i | < e, rdf : type, c_i >\in G\}$ *is called as the class component;*
- $A_e = \{p_i | < e, p_i, l_i >\in G$ *and* l_i *is a literal*$\}$ *is called as the attribute component;*
- $R_e = \{r_i | < e, r_i, o_i >\in G$ *and* o_i *is a URI resource or blank node*$\}$ *is called as the relation component;*
- $V_e = \{v_i | < s_i, v_i, e >\in G\}$ *is called as the inverse relation component.*

Taking the RDF graph g_1 in Fig. 1 for example, the EDP of entity *jeff-z-pan* is $\big(\{foaf{:}Person\}, \{foaf{:}name\}, \{foaf{:}made\}, \emptyset\big)$, and the one of *ISWC09_423* is $\big(\emptyset, \{rdfs{:}label\}, \emptyset, \{foaf{:}made\}\big)$.

By generating EDPs for all entities in an RDF graph G, we can get a EDP representation of G, which is a set of EDPs. As we will show in later section, the number of EDPs in an RDF dataset is usually much less than the number of entities. This is because many entities are sharing same EDP. The more entities are sharing one EDP; the more frequent its structure is duplicated. Such duplications are the source of data redundancies, which we break down to semantic redundancy and syntactic redundancy.

Semantic Redundancy Identified By EDP In the definition of EDP, the class component C_e is a set of constant class names. Hence, it is straightforward to generate a graph pattern based substitution rule for removing these type assertions from the original data. In particular, the substitution rule is defined as $(\emptyset, A_e, R_e, V_e) \rightarrow (C_e, A_e, R_e, V_e)$. Let f_{P_e} be the instance number of P_e, the number of triples can be removed by the rule is $|C_e| \times f_{P_e}$. In *jeff-z-pan*'s example, the rule will be $(\emptyset, \{foaf{:}name\}, \{foaf{:}made\}, \emptyset) \rightarrow (\{foaf{:}Person\}, \{foaf{:}name\}, \{foaf{:}made\}, \emptyset)$. In this particular case, this rule is equivalent to the rule of <$foaf{:}name, rdfs{:}domain, foaf{:}Person$> from $FOAF$ vocabulary.

Inter-structural Syntactic Redundancy Identified By EDP The inter-structural redundancy denotes the unnecessary structure recurrences in EDP's instances, i.e. EDBs. Given an EDP P_e, it is straightforward to calculate its recurrences as $(|A_e| + |R_e| + |V_e|) \times f_{P_e}$, the unit of which is resource occurrence.

Virtual A-Box Materialisation on EDP When considering T-Box of an RDF dataset, the data redundancy might be changed due to the above-mentioned $removable$ and $derivable$ triple sets. To compute the two triple sets, inferences need to be performed on A-Boxes, which might be too expensive for large scale data analyses. For redundancy analysis purpose, we only need to know the size of $removable$ and $derivable$ sets instead of getting the exact triples. Inspired by this observation, we propose a virtual materialisation approach based on EDP to compute the triple sizes for the two triple sets.

As supporting efficient query answering is the basis of many Linked Data consumption tasks, our discussion in this paper is based on OWL2 QL profile [2]. Given an EDP P_e, according OWL2 QL profile, it can be proved that the four components of P_e are sufficient for inference computation on EDP. When applying inference rules defined in [2] on an EDP, we use a new EDP: $P_e^M = (C_e^M, A_e^M, R_e^M, V_e^M)$ to store all the inference results. After the inference, the number of *derivable* triples can be calculated as follows.

$$|derivable| = f_{P_e} \times \sum_{c \in N} f(c), \qquad (1)$$

where $N = (C_e^M \setminus C_e) \cup (A_e^M \setminus A_e) \cup (R_e^M \setminus R_e) \cup (V_e^M \setminus V_e)$ *and* f *is an auxiliary dictionary which stores the instantiated times of each concept in* $P_e \cup P_e^M$.

The number of *removable* triples can be calculated as follows.

$$|removable| = f_{P_e} \times \sum_{c \in E} f(c), \qquad (2)$$

where $E = (C_e^M \cap C_e) \cup (A_e^M \cap A_e) \cup (R_e^M \cap R_e) \cup (V_e^M \cap V_e)$.

5 Redundancy Analysis Results on Linked Datasets

5.1 Datasets

The datasets selected for analysis are identified from Linked Open Data cloud [6]. There is a coloured version which categorises the datasets in different domains. We selected 5 datasets from the Linked Open Data Cloud, which cover 5 out 6 domains listed in the coloured version. The general information about datasets is listed in Table 2.

In addition to trying to have a diverse domain coverage in dataset selection, we tried to select datasets with different sizes (cf. the $\#Triples$ column). We also selected two different datasets from one particular dataset (the Ordnance Survey), which are quite different in topics and size. The main purpose is to have our sample as representative as possible, while keeping the number of datasets manageable.

One strategy we apply in our analyses it to focus on data instances of most popular EDPs instead of working on all data. The idea is to maintain an efficient analysis while capturing the most part of the data. The threshold chosen is 90%, which means that we analysis 90% of the data.

[6] http://lod-cloud.net/

Table 2. General Information of Selected Datasets

Dataset	Domain	#Triples	$\#EDP$	$\#EDP@90\%$
LinkedMDB	Media	6,148,121	10,316	26
LOV	User-generated content	54,630	492	15
DBLP	Publication	94,450,169	438	6
Ordnance Survey (50K Gazetteer)	Geographic & Government	2,368,655	6	1
Ordnance Survey (Code-Point)	Geographic & Government	33,750,456	19	2

In the table, there are other metrics related to EDP. $\#EDP$ is the total number of EDPs identified in the dataset, which is a quantitative measurement about the variety of how entities are described. $\#EDP@90\%$ is the minimal number of top popular EDPs the sum of whose instance numbers is not less than 90% of the total number of entities in the dataset. From Table 2, although the total number of EDPs are large in some cases, e.g., LinkedMDB has more than 10 thousand EDPs, the major part of the data (90%) resides in a very small number of EDPs in all cases. The more popular one EDP is; the more redundancy there will be. Hence, having a small number of very popular EDPs indicates a large volume of data redundancies.

5.2 The results

A-Box only results Table 3 gives the analysis results by only considering A-Box level information. Both syntactic and semantic redundancies are analysed by EDP based approach. In the table, $\#RRes$ is the redundant resource occurrences of inter-structural redundancies; $RRatio_{syn}$ is the syntactic redundancy ratio, i.e. $\frac{\#RRes/3}{\#Triples}$; $\#RTriple$ is the redundant triples (i.e. semantic redundancy); $RRatio_{sem}$ is the semantic redundancy ratio, i.e. $\frac{\#RTriple}{\#Triples}$; and $\#GPRules$ is the number of graph pattern substitution rules needed to remove semantic redundant triples.

As for *syntactic redundancies*, in all datasets, they are considerably large. The redundant ratio is more than 20% in all cases except DBLP where the ratio is still near 6.5%. The biggest ratio was obtained from *50K Gazetteer*, which is more than 32%. Furthermore, it is notable that we only consider inter-structural redundancy and the ratio is calculated from redundancies in 90% data over the whole data. This means that the overall syntactic redundancy ratio should be even more.

Table 3. A-Box Only: Semantic Redundancy and Syntactic Redundancy

Dataset	Syntactic Redundancy		Semantic Redundancy		
	$\#RRes$	$RRatio_{syn}$	$\#RTriple$	$RRatio_{sem}$	$\#GPRules$
LinkedMDB	4,475,952	24.27%	610,463	9.93%	21
LOV	36,718	22.40%	8,845	16.19%	8
DBLP	18,283,964	6.45%	2,901,347	3.07%	3
Ordnance Survey (50K Gazetteer)	2,331,720	32.81%	259,080	10.94%	1
Ordnance Survey (Code-Point)	27,455,294	27.12%	1,595,931	4.73%	3

The right part of Table 3 shows that the EDP approach can identify substantial semantic redundancies as well. More than 3% of triples are redundant in all analysed datasets. The most semantically redundant one is *LOV* dataset, which has more than 16% redundant triples. The $\#GPRules$ column shows an interesting phenomenon, i.e. these semantic redundancies can be removed by very small number of rules. The most efficient rule set comes from *50K Gazetteer*, where one single rule can remove more than 10% triples.

Linked Semantics Table 4 shows the data redundancies under different explicit semantics. Two types of semantics of *No Linkage* and *T-Box Reuse* are analysed. $\#DTerm$ is the number of terms from reused T-Box, which are directly used by current dataset; and $\#RAxioms$ is the number of axioms from (materialised) reused T-Box, which are applicable in the dataset materialisation.

No Linkage analyses are done by considering the datasets' main T-Boxes. As shown in left part of Table 4, *LinkedMDB* and *LOV* do not have a main T-Box. They are using concepts defined in their own name spaces but without specifications about the semantics of these concepts. Such situations are very common in Linked Data Cloud. For *DBLP* dataset, we identified its main T-Box as SWRC ontology [7]. Considering this T-Box, around 1.6 million triples can be further derived but no triples in the dataset are removable. This means that the current data is semantically less redundant when T-Box axioms are taken into account. The case of *50K Gazetteer* is similar, where main ontology of the dataset is officially published [8]. An interesting finding is that the $derivable$ triples of *50K Gazetteer* are even more than its original triples. This indicates that the dataset is published in a quite concise way. In *Code-Point*'s case, the T-Box can help remove around 1.6 million triples. This means that these triples turn to be redundant.

Table 4. Linked Semantics: Data Redundancies Considering (Linked) T-Box Axioms

Dataset	No Linkage		T-Box Reuse			
	$derivable$	$removable$	$derivable$	$removable$	$\#DTerms$	$\#RAxioms$
LinkedMDB	-	-	1,652,385	0	2	6
LOV	-	-	2,197	0	2	11
DBLP	1,669,644	0	42,851,260	1,231,703	2	10
Ordnance Survey (50K Gazetteer)	4,361,100	0	-	-	-	-
Ordnance Survey (Code-Point)	36,706,413	1,595,931	-	-	-	-

The right part of Table 4 gives the analysis result of considering reused T-Box axioms. 3 out of 5 datasets are reusing one popular T-Box, i.e. FOAF ontology. In *LOV* dataset, about 4% new triples can be inferred by the reusing FOAF ontology. In other two cases, *LinkedMDB* and *DBLP*, a surprisingly large number of triples can be derived. Note that in all cases, the datasets are only using two terms from FOAF. Even the total number of axioms applicable for the inference is quite small (cf. $\#RAxioms$ in the table). This indicates that *T-Box Reuse* might lead to substantial $derivable$ triples even when the number of reused terms is very small.

[7] http://ontoware.org/swrc/

[8] http://data.ordnancesurvey.co.uk/datasets/os-linked-data

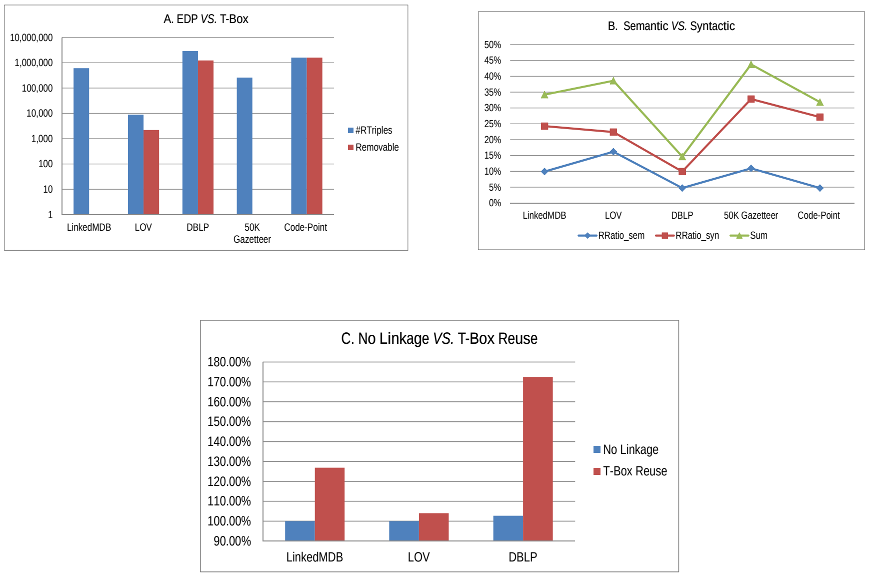


Fig. 4. Comparisons: A. *EDP VS. T-Box*; B. *Semantic VS. Syntactic*; C. *No Linkage VS. T-Box Reuse*.

Comparisons The top-left figure in Fig. 4 compares the volumes of semantic redundancies identified by EDP approach ($\#RTriples$) and T-Box axioms ($Removable$). The $Removable$ in the figure is the sum of $Removable$s of both *No Linkage* and *T-Box Reuse*. In *Cod-Point* dataset, the two approaches identified the same number of redundant triples. In all other cases, EDP approach is much better. This indicates that there exist a large volume of semantic redundancies which are only identifiable to more generalised rule systems than the T-Box axiom.

The top-right figure illustrates the comparison between semantic redundancy and syntactic redundancy. In all cases, the semantic one is less. The Sum of the two gives an idea about the overall data redundancies in these datasets. In 4 out of 5 datasets, the redundancy is more than 30%. The largest one is in *50K Gazetteer*, which has 44% redundant data, and *DBLP* is least redundant dataset with about 15% redundancy.

The bottom figure in Fig. 4 shows the differences in the size of A-Box serialisation with and without Reused T-Box semantics. In the figure, the size of serialisation is illustrated using the percentage: $Percentage = \frac{|\text{A-Box Materialisation}|}{|\text{Original A-Box}|}$. In all cases, the materialisations are increased by *T-Box Reuse* significantly. Notably, there are more than 26% increment for *LinkedMDB* and nearly 70% for DBLP.

6 Conclusion

In this paper, we introduced a systematic approach for analysing Linked Data redundancy. The most straightforward conclusion is that data redundancies in Linked Datasets are not only huge but also diverse. This leads to our main conclusion: *redundancy-aware* *techniques are demanded for both data consumption and data publishing*. The conclusion is supported by several observations as follows.

– **For Data Compression** The compression tools or techniques should be aware that different types of redundancies exist. Different redundancies might need different techniques. Moreover, the knowledge of redundancy distribution is crucial to tools which want to make a trade-off between efficiency and compression ratio.
– **For Data Access** As shown in the results (cf. Table 2), data entities are described via different data patterns (EDPs). In most cases, the data distribution among EDPs is skewed, i.e. a small number of EDPs have a large number of data instances. For efficient data access purpose, the knowledge of such distribution and how to make use of it are critical.
– **For OBDA and reasoning** Existing wok has shown that A-Box redundancy can be utilised to avoid unnecessary rewritings. Results in this paper suggests that moving a step forward from using rules equivalent to T-Box axiom to more general rule systems might improve the effectiveness of existing approaches significantly.
– **For Data Publisher** Being encouraged to link their data to other datasets, the publisher should be aware of the consequences of such linking in terms of bringing in $removable$ and $derivable$ triples. We have shown that a very small number of reuses can lead to a very large number of $derivable$. Publishers should be alarmed before creating links, and tools are needed to estimate such consequences and prompt them to the publisher.

References

1. S. Álvarez-García, N. R. Brisaboa, J. D. Fernández, and M. A. Martínez-Prieto. Compressed k2-triples for full-in-memory rdf engines. arXiv preprint arXiv:1105.4004, 2011.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. Journal of Automated reasoning, 39(3):385–429, 2007.
3. O. Curé, G. Blin, D. Revuz, and D. C. Faye. Waterfowl: A compact, self-indexed and inference-enabled immutable rdf store. In The Semantic Web: Trends and Challenges, pages 302–316. Springer, 2014.
4. L. Ding, J. Shinavier, Z. Shangguan, and D. L. McGuinness. Sameas networks and beyond: analyzing deployment status and implications of owl: sameas in linked data. In The Semantic Web–ISWC 2010, pages 145–160. Springer, 2010.
5. J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary rdf representation for publication and exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web, 19:22–41, 2013.
6. H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson. When owl: sameas isnâĂŹt the same: An analysis of identity in linked data. In The Semantic Web–ISWC 2010, pages 305–320. Springer, 2010.
7. L. Iannone, I. Palmisano, and D. Redavid. Optimizing rdf storage removing redundancies: an algorithm. In Innovations in Applied Artificial Intelligence, pages 732–742. Springer, 2005.
8. A. K. Joshi, P. Hitzler, and G. Dong. Logical linked data compression. In The Semantic Web: Semantics and Big Data, pages 170–184. Springer, 2013.
9. J. Z. Pan, J. M. G. Pérez, Y. Ren, H. Wu, and M. Zhu. SSP: Compressing RDF data by summarisation, serialisation and predictive encoding. Technical report, 07 2014. Available as `http://www.kdrive-project.eu/resources/`.