Statistically-driven generation of multidimensional analytical schemas from Linked Data

Victoria Nebot*, Rafael Berlanga

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec, 12071, Castellón, Spain
Phone: (+34) 964 72 83 67. Fax: (+34) 964 72 84 35

Abstract

The ever-increasing Linked Data (LD) initiative has given place to open, large amounts of semi-structured and rich data published on the Web. However, effective analytical tools that aid the user in his/her analysis and go beyond browsing and querying are still lacking. To address this issue, we propose the automatic generation of multidimensional analytical stars (MDAS). The success of the multidimensional (MD) model for data analysis has been in great part due to its simplicity. Therefore, in this paper we aim at automatically discovering MD conceptual patterns that summarize LD. These patterns resemble the MD star schema typical of relational data warehousing. The underlying foundations of our method is a statistical framework that takes into account both concept and instance data. We present an implementation that makes use of the statistical framework to generate the MDAS. We have performed several experiments that assess and validate the statistical approach with two well-known and large LD sets.

Keywords: Linked Data, RDF, Multidimensional models, Statistical models

1. Introduction

The vision of the Semantic Web (SW) is to create a common framework that allows data to be shared and reused at different levels (i.e., between applications, enterprises and communities). The most tangible realization of the SW is the Linked

*Corresponding author
Email addresses: romerom@uji.es (Victoria Nebot), berlanga@uji.es (Rafael Berlanga)

Data (LD) cloud, which contains around 85 billion triples over 10 thousand different datasets (as of September 2015)[1] and it is expressed using the Resource Description Framework (RDF) [23] modeling language. Lately, communities from different areas as well as governments and public organizations have published large volumes of interlinked data in the LD cloud following the publication guidelines and providing the basis for creating and populating the Web of Data [14].

Given the explosive growth both in data size and also schema complexity and heterogeneity, LD sources are becoming increasingly difficult to understand and use, which limits the exploration and the exploitation of potential information they contain. This has brought to the fore the need for new tools able to explore, query, analyze and visualize these semi-structured, semantically-enriched and heterogeneous data sets [10]. While several different tools such as graph-based query builders, semantic browsers and exploration tools [5, 7, 15, 4] have emerged to aid the user in querying, browsing and exploring LD, these approaches have limited ability to summarize, aggregate and display data in the form that a scientific or business user expects, such as aggregation tables and graphs. Moreover, they fall short when it comes to provide the user an overview of the potential data that may be of interest from an analytical viewpoint.

In this paper our hipothesis is that LD constitutes a valuable source of knowledge worth exploiting from a multidimensional (MD) perspective. Specially the Business Intelligence (BI) field can benefit enormously from adding and aligning the new knowledge uncovered from LD sources with the existing corporate data warehouses to make better and more informed decisions. BI uses the MD model to view and analyze data in terms of dimensions and measures, which seems the most natural way to arrange data. BI has traditionally been applied to internal, corporate and structured data, which is extracted, transformed and loaded (ETL) into a pre-defined and static MD model. The relational implementation of the MD data model is typically a star schema. The dynamic and semi-structured nature of LD poses several challenges to both potential analysts and current BI tools. On one hand, manual exploration of the datasets using the available browsers and tools to find MD patterns is cumbersome due to the heterogeneity and incompleteness of LD and the lack of support for obtaining informed summaries. Moreover, as the datasets are dynamic, their structure may change or evolve, making the one-time MD design approach unfeasible.

This paper approaches the exploration and discovery of potential analytical

[1] http://stats.lod2.eu/

data from LD sources in a radical and innovative way. We propose a statistical framework to automatically discover candidate MD patterns hidden in LD sources. We call these patterns multidimensional analytical stars (MDAS) . A MDAS is a MD star-shaped pattern at the class level that encapsulates an interesting MD analysis [26]. The main innovations and contributions of this research to the LD and BI community are stated below:

- We define the concept of MDAS as a mapping of the MD model to a statistical layer on top of LD sources. This statistical layer is able to deal both with heterogeneity and incompleteness common in LD sets and feeds itself from the RDF schema graph elements and the instance data. We characterize and identify the facts and potential dimensions and measures that compose a MDAS in terms of classes and properties, whose underlying pattern has been inferred in the statistical layer.

- We develop a statistical framework to approach the problem of discovering MDAS in a foundational and automatic way. The automation of the process relieves the analyst from the burden of having to explore the dataset to become acquainted with it. Moreover, the discovery of the MDAS is driven both by the implicit semantics of the data and the statistical arrangement of the triple instances.

- We overcome the long-term known issues of data heterogeneity and incompleteness in the LD world. Both issues arise from the very same nature of LD and, in particular, the RDF modeling language, which does not impose a hard schema on the instance data. This modeling approach provides more flexibility than traditional modeling approaches based on hard schemas and constraints but introduces the above-mentioned new challenges. The statistical nature of the developed approach is able to deal with both heterogeneity and incompleteness of the datasets and discovers different configurations of MDAS that capture such heterogeneity.

- The statistical nature of the approach allows us to use well-known sampling techniques to build the statistical model instead of using the complete dataset, which may not always be available (e.g., SPARQL endpoints) or is too large to be processed (e.g., Big Data).

- We provide an implementation that makes use of the statistical model to generate MDAS. The algorithms provide the analyst with all the pieces to

compose and configure MDAS while ensuring the population with instance data. On one hand, we automatically generate the *bases* of the stars (i.e., the nucleus), where a ranking of properties according to their relevance for the star is presented so that the user can select properties aided by the statistical indicators. Moreover, we also generate dimension types to enrich the base of the star with potential dimensions and measures. Dimension types are organized into groups to alleviate heterogeneity among dimensions that are expressed syntactically different but are semantically similar.

- We present several experiments with two different and well-known LD sets and assess the quality of our statistical model to generate MDAS.

The structure of the paper is as follows. In Section 2 we motivate our approach with an example. Section 3 presents preliminary concepts. Section 4 presents the main foundations that underlie our approach. That is, we present a model for MDAS over LD sources. Section 5 defines the statistical model developed to approach the problem of generating MDAS and the implementation. Section 6 presents the experiments and results. In Section 7 we review the literature related to the problem of analyzing LD and Section 8 gives some conclusions and future work.

2. Motivating example

We will now illustrate the need for an automatic and statistical approach to infer MD patterns (i.e., MDAS) from LD sources. We use the Enipedia[2] dataset for the examples. Enipedia is an initiative aimed at providing a collaborative environment through the use of wikis and the SW for energy and industry issues. They provide energy data from different open data sources structured and linked in RDF. Figure 1 shows an "ideal" RDF knowledge base that models information about `Powerplants`, which have a `Country`, the type of `Fuel` and the `carbonemmissions-23kg` associated (see the upper schema part). The lower part of the figure has descriptions about resources, e.g., powerplants ($\&r1, \&r3$), countries ($\&r4, \&r5$) and fuel type ($\&r2, \&r7$). The resources are connected to other resources or literals by an arc if there is an `rdf:property` relating them. For example, the fact that *&r1* is connected to *&r4* by a `state` property means that the resource represented by *&r1* is located in the resource represented by *&r4*. In addition, resources are connected to

[2]http://enipedia.tudelft.nl/wiki/Main_Page

resource classes using the `rdf:type` property, which indicates that the resource is a member of the class it is connected to. For example, resource $\&r1$ is a `Powerplant`. In the schema, classes and properties may also be related by `rdfs:subclassOf` and `rdfs:subpropertyOf` properties, respectively, indicating a hierarchy on classes and properties. Also, the domains and ranges of a property may be defined using the `rdfs:domain` and `rdfs:range` properties.

In the previous ideal scenario, where all resources comply with a well-defined schema, it is easy to see that, by exploring the schema, we can find interesting associations between classes that resemble MD patterns. For example, from the previous schema, the class `Powerplant` is a good candidate for being the fact, the classes `Country` and `Fuel` can act as dimensions and the `carbonemmissions-23kg` property can act as the measure.

Figure 1: Example of an ideal RDF knowledge base.

Unfortunately, the real LD world could not be further from the previous ideal scenario. The flexibility of the RDF model and the decentralized approach for creating and publishing LD has resulted in very heterogeneous datasets, where not only different datasets modeling the same domain objects are annotated with different schemas, but also, within datasets, the schemas are usually very poor (if non-existent) and the annotation of resources is incomplete and incoherent. Therefore, the exploration and use of these datasets for analytical purposes becomes increasingly difficult. Figure 2 shows an example of the heterogeneity in the Enipedia dataset, where four resources of type `Powerplant` are presented. We observe that different properties are used to refer to the location of the powerplant (i.e., `country`, `state`, `county`), with no defined relation among them neither domain and range definition at the schema level. It is also frequent that a property is used with different domains and ranges. The class representing the location also varies from `Country` to `OECDMember` with no relation between them. The same occurs with classes `Fuel` and `Energy_concept`, which represent apparently the same class. Also, not all powerplant resources have attached the same data properties (i.e., $\&r4$ and $\&r7$ have `carbonemmissions-23kg`, whereas $\&r10$ has also the property `carbonemmissionsnextdecade-23kg` and $\&r1$ has none).

Figure 2: Example of heterogeneity in LD sources.

In the current LD scenario, approaches that rely only on exploiting the schema

to discover interesting associations and patterns in the data become insufficient, as in most of the cases, the only schema that we find of top of LD sets are the semantic types of the resources, that is, the classes. Even in the cases where data have a larger semantic layer on top, this schema is usually heterogeneous both in its shape and usage. Therefore, in this paper we approach the problem by proposing a solution that combines both the semantics provided by the existing schema and statistical information derived from the instance data.

3. Preliminaries

In this section we introduce the main foundations and related topics that underlie the developed approach.

3.1. Multidimensional modeling

The term BI refers to all the decision support technologies aimed at making better informed and faster decisions in the enterprise environment. During the past two decades, businesses have been increasingly leveraging their data with sophisticated analysis techniques to get comprehensive knowledge and gain insight of their data. Data warehousing and OLAP are now mature technologies and have been traditionally applied in the field of BI. Both technologies are based on the MD model abstraction. Multidimensionality is based on the fact/dimension dichotomy [20]. The information is conceptually modeled in terms of facts, the central entities of the desired analysis (e.g., a sale), and dimensions, which provide contextual information for the facts (e.g., the products sold). The dimensions are also known to be the different points of view (i.e., MD point) from where a fact can be analyzed. A cube consists of fact instances (or simply facts), where each fact is identified by a MD point (a point for each dimension) and quantified by measure values. Usually, the dimensions are hierarchically organized into levels. For instance, products can be grouped into product categories. Typically, the facts have associated numerical measures (e.g., the quantity sold or the total price), and queries aggregate fact measure values up to a certain level (e.g., total profit by product category and month). This provides the user an easy-to-understand and dynamic visualization of data.

The application of MD modeling to corporate, structured relational data has been for quite some time an active area of research, at the logical level [21] and later at the conceptual level [13]. Early MD methods view the modeling of facts and dimensions as a complex and manual process performed by the data warehouse designer. Other approaches try to assist the designer and sometimes

even automate this complex process either by automatically analyzing the data sources (i.e., supply-driven approaches) or by formalizing the user requirements (i.e., requirement-driven approaches). Hybrid approaches combine both strategies.

Regarding the implementation of the MD model using the relational technology (ROLAP), the star schema is the most common logical schema used, where data are stored in relational tables. The star schema consists of a fact table plus one dimension table for each dimension. Each tuple in the fact table has a foreign key column to each of the dimension tables and numeric columns that represent the measures.

Despite the importance and great success of applying MD modeling to analyze relational data, the issue of automatically discovering MD analytical patterns in sources other than relational (i.e., semi-structured, heterogeneous, semantics-enriched data) has been poorly addressed in the literature [31, 2]. In our approach, we aim at discovering these MD analytical patterns from unconventional data (i.e., LD sources) based on a) the semantics of the data schema and b) the probabilistic distribution of the instance data.

3.2. Linked Data and RDF/S

The evolution of the Web from a global information space of linked documents to one where data are linked (Web of Data) has propelled a set of best practices for publishing and connecting structured data on the Web known as LD. Among the basic principles for publishing LD we highlight the following: each entity has a unique URL identifier, the identifiers should be dereferenceable by HTTP and the entity representations should be interlinked together to form a global LD cloud.

RDF has become the most adopted data model to represent LD, as RDF allows for the description of resources and how they relate to each other. The underlying structure of RDF is a collection of triples of the form (*subject predicate object*), which form an RDF graph. An RDF triple states that there is some relationship, indicated by the predicate, holding between the subject and the object of the triple. We consider only valid RDF triples using URIS (U), blank nodes (B) and literals (L). By means of the clause `rdf:type` we can express class membership, even though in RDF there is no technical distinction between the schema and the instance data.

RDFS [8] defines a simple modeling language on top of RDF. An RDF schema defines a finite set of class names C and property names P, uniquely identified by the names in $N = C \cup P$. It provides primitives that allow to express set membership of objects in property and class extensions. That is, RDFS uses classes,

subsumption relationships on both classes and properties, and global domain and range restrictions for properties as modeling primitives. For each property $p \in P$, $domain(p)$ is a class, and $range(p)$ is either a class or a literal.

RDFS also provides inference semantics: structural inference given by the transitivity of subsumption relations and type inference given by the typing system. The closure of an RDF schema contains all triples that are either explicit or can be inferred from explicit triples using the previous types of inference. An RDFS knowledge base $\mathcal{S}$ is an RDF schema graph that is closed w.r.t. both structural and type inference.

We use the traditional graph data model to represent RDF/S data (both the RDF schema and the RDF instance data). In particular, an RDF schema graph $\mathcal{S}$ is a labelled directed graph that expresses a collection of triples $\mathcal{T}_{\mathcal{S}} = (s, p, o) = U \times U \times U$ where the nodes represent classes $c \in C$, the edges represent properties $p \in P$ and there can be hierarchical relations between classes and properties.

An RDF instance graph I is a labelled directed graph that expresses a collection of triples $\mathcal{T}_I = (s, p, o) = U \times U \times (U \cup L)$ where the nodes represent instances of classes $c \in C$ and the edges represent properties $p \in P$ of the RDF schema graph $\mathcal{S}$. That is, `rdf:type` declarations link the RDF instance graph I with the RDF schema graph $\mathcal{S}$.

Notice that, even though we have modelled an RDF instance graph and its corresponding schema graph, usually, LD sources do not contain complete and consistent schemas as defined here. Therefore, the goal of this paper is to statistically capture the heterogeneity of the data and to infer MD patterns at the schema level.

4. MD analytical stars

In this section we develop the MD data model of a MDAS from LD sources, which is based on the model proposed by [30]. In this paper, we do not address the construction of dimension hierarchies from semantic sources, as this has been previously addressed in [26].

Definition 4.1. A MDAS is a two-tuple $S = (F_T, D_T)$, where F_T is a *fact type* and $D_T = \{T_i\}_{i=1,\ldots,n}$ is its corresponding *dimension types*.

In the RDF model, the fact type maps to a class, that is, $F_T \in C$. Thus, the following SPARQL query retrieves all possible fact types.

```
SELECT DISTINCT ?c
WHERE {
    ?e a ?c .
}
```

Listing 1: SPARQL query for retrieving fact types

Example 4.1. For demonstration purposes, we will show the generation and shape of one possible MDAS from the example data set shown in Figure 1. However, our method generates all possible MDAS that satisfy certain restrictions. Suppose we select `Powerplant` class as fact type.

Definition 4.2. A dimension type T is a sequence $(c_1, r_1, c_2, ..., r_{n-1}, c_n)$ where $c_1 = F_T$, $c_i, ..., c_{n-1} \in C$, $c_n \in C \cup DT$, $r_i \in P$ and there are no cycles[3]. The length of a dimension type is determined by the number of triple patterns that compose it (i.e., $n-1$).

In the RDF model, a dimension type maps to a sequence of classes and properties that start at the fact type and finish in a class or a datatype container.

Notice that the length of a dimension type is not bounded, which makes the possible number of dimension types exponential. As an example, the following SPARQL query shows how to retrieve all possible dimension types of length one, that is, sequences (c_1, r_1, c_2), taking into account that `Powerplant` is the selected fact type. For dimension types of length n, a similar query must be executed by joining triple patterns until reaching the exact length. In our model, everything that characterizes the fact type is considered to be dimensional (i.e., dimension types), even attributes that would be considered as measures in other models.

```
SELECT   ?r1 ?c2
WHERE {
    ?e1 a Powerplant .
    ?e1 ?r1 ?e2 .
    FILTER (datatype(?e2) = ?c2 || ?e2 a ?c2 )
}
```

Listing 2: SPARQL query for retrieving dimension types of length one given `Powerplant` as fact type

[3]From now on, we use the symbol r to refer to properties instead of p.

We support the aggregation semantics as in [30] for each dimension type T by keeping track of what types of aggregate functions can be applied. We distinguish between three types of aggregate functions: Σ, applicable to data that can be added together, ϕ, applicable to data that can be used for average calculations, and ε, applicable to data that is constant, i.e., it can only be counted. Considering only the standard SQL aggregation functions, we have that Σ= SUM, COUNT, AVG, MIN, MAX, ϕ=COUNT, AVG, MIN, MAX and ε=COUNT. The aggregation types are ordered, $\varepsilon \subset \phi \subset \Sigma$, so data with a higher aggregation type, e.g., Σ, also possess the characteristics of the lower aggregation types. We assume a function $Aggtype : T \rightarrow \{\Sigma, \phi, \varepsilon\}$ that gives the aggregation type for each dimension type.

Example 4.2. Following with the example, from all the possible dimension types of any length, we select the following ones: *Country*=(`Powerplant`, `state`, `Country`), *Fuel*=(`Powerplant`, `fuel_type`, `Fuel`, `rdfs:label`, `string`) and *CarbonEmmissions*: (`Powerplant`, `carbonemmissions-23kg`, `float`). According to the previous aggregation semantics, the aggregation types of the dimension types are:
Aggtype(CarbonEmmisions)= Σ, *Aggtype(Country)*= ε, and *Aggtype(Fuel)*= ε.

Definition 4.3. A set of facts $F = \{f\}$ is a set of resources f such that each f is of type F_T.

The following SPARQL query retrieves the facts given the fact type `Powerplant`.

```
SELECT DISTINCT ?f
WHERE {
    ?f a Powerplant .
}
```

Listing 3: SPARQL query for retrieving facts

Example 4.3. The facts of type `Powerplant` are $\{\&r1, \&r3\}$

Definition 4.4. A dimension D of type $T = (c_1, r_1, c_2, ..., r_{n-1}, c_n)$ is a set of resource sequences $(e_1, r_1, e_2, ..., r_{n-1}, e_n)$ where $e_1, ..., e_{n-1} \in U$, $e_n \in U \cup L$, $r_i \in P$ and $\forall e_i$, $(e_i\ a\ c_i) \in \mathcal{T}_S$ and $(e_i, r_i, e_{i+1}) \in \mathcal{T}_I$. We denote with $D_v = \{e_{in}\}$ the set of dimension values, which are the values at the end of each sequence.

The following SPARQL query populates the dimension type *Country*=(`Powerplant`, `state`, `Country`).

```
SELECT   ?e1 ?r1 ?e2
WHERE {
    ?e1 a Powerplant .
    ?e2 a Country .
    ?e1 ?r1 ?e2
}
```

Listing 4: SPARQL query for populating the dimension type *Country*

Example 4.4. The dimension *Country* is composed by the set of sequences $Country = \{(\&r1, state, \&r4), (\&r3, state, \&r5)\}$. $Country_v$ are the values $\{\&r4, \&r5\}$

Definition 4.5. Let F be a set of facts, and D a dimension. A fact-dimension relation between F and D is a set $R = \{(f, e)\}$, where $f \in F$ and $e \in D_v$. Thus R links facts to dimension values.

Example 4.5. The fact-dimension relation R links `Powerplant` facts to *Country* dimension values. The result is $R = \{(\&r1, \&r4), (\&r3, \&r5)\}$.

Definition 4.6. A multidimensional object (MO) is a four-tuple $M = (S, F, D, R)$, where $S = (F_T, D_T = \{T_i\})$ is the fact schema, $F = \{f\}$ is a set of facts of type F_T, $D = \{D_i\}_{i=1,\ldots,n}$ is a set of dimensions where each D_i is of type T_i and $R = \{R_i\}_{i=1,\ldots,n}$ is a set of fact-dimension relations, such that $(f, e) \in R_i \implies f \in F \wedge e \in Dv_i$.

Example 4.6. From the running example, we obtain a MO $M = (S, F, D, R)$ where: $S = (Powerplant, \{Country, Fuel, CarbonEmmissions\}$ and $F = \{\&r1, \&r3\}$.
Now, we describe the dimension types, whose aggregation semantics are defined in Example 4.2.
Country=(Powerplant, state, Country),
Fuel=(Powerplant, fuel_type, Fuel, rdfs:label, string),
CarbonEmmissions=(Powerplant, carbonemmissions-23kg, float).
The dimensions:
Country={(&r1, state, &r4), (&r3, state, &r5)},
Fuel={(&r1, fuel_type, &r2, rdfs:label, "Wind"),
(&r3, fuel_type, &r7, rdfs:label, "Gasoil")},
CarbonEmmissions={(&r1, carbonemmissions-23kg, 8.1e+4),
(&r3, carbonemmissions-23kg, 2.1e+5)}.
The R relation:

$R_{Country} = \{(\&r1, \&r4), (\&r3, \&r5)\}$,
$R_{Ruel} = \{(\&r1, \text{“}Wind''), (\&r3, \text{“}Gasoil'')\}$,
$R_{CarbonEmmissions} = \{(\&r1, 8.1e+4), (\&r3, 2.1e+5)\}$.

The standard relationship between fact and dimension is many-to-one: each fact links to one and only one dimension value and, usually, each dimension value points back to many facts. As this is too strict in our setup, we do not enforce this constraint. Instead, we check the capacity of a dimension type to aggregate or summarize fact values into dimension values, and enforce the found dimension types to satisfy a certain degree of aggregation power. That is, we try to avoid one-to-one and one-to-many relationships, where no real aggregation is being produced between the set of facts and the set of dimension values.

Definition 4.7. Given a dimension type $T = (c_1, r_1, c_2, r_2, ...r_{n-1}, c_n)$, the aggregation power of T, $Aggpower : T \rightarrow \mathbb{R}$, is calculated as the ratio between the number of instances of the fact class c_1 and the number of different instances (or literals) of the sink class (or datatype) c_n that satisfy the query given by the dimension type.

Example 4.7. Given the number of instances (or literals) in parenthesis that satisfy the underlying queries of the dimension types, we show their aggregation power:

$$Country = (Powerplant_{(160)}, state, Country_{(13)}) \rightarrow$$
$$Aggpower(Country) = 12.3$$
$$Fuel = (Powerplant_{(11994)}, fuel\_type, Fuel, label, string_{(27)} \rightarrow$$
$$Aggpower(Fuel) = 444.2$$

In order to exactly calculate the aggregation power, one must execute the SPARQL query that populates the dimension type, and then calculate the ratio between the number of facts and the number of distinct dimension values. In the next section, we show an approximation to calculate the aggregation power of a dimension type that does not need to execute a SPARQL query.

To summarize, the elements of the data model presented in Defs. 4.1 and 4.2 refer to the schema part of the MDAS, that is, $S = (F_T, D_T)$ are the fact type and dimension types, whereas elements defined in Defs. 4.3 to 4.5 refer to the instance data that will populate an MDAS, that is, the facts, dimensions and fact-dimension relations. The MO presented in Def. 4.6 is the element that groups together the

schema and the instance data of the star. Recall that the main aim of this paper is to automatically discover MDAS, that is, the schema part of a MO, as the population with instance data can be easily done by means of SPARQL queries once we know the schema.

Notice that a brute-force algorithm that searches any possible combination of fact type and dimension types is unfeasible as the search space of the problem is combinatorial. Also, the approach is not scalable because SPARQL query processors suffer from scalability issues when queries imply many joins.

Thus, we have devised a probabilistic approach for generating MDAS that guides the exploration of the search space for finding fact and dimension types. The statistical nature of the proposed model allows us to deal with the incompleteness and heterogeneity inherent in LD sets.

5. Statistical model

In this section we show the statistical model devised to provide an estimation of "good" candidate MDAS. The first part shows the statistical indicators that compose the model, in the second part, we provide the algorithms that make use of the statistical model in order to generate MDAS, finally, we comment on the computational complexity of the implementation.

Intuitively, a candidate MDAS should keep a good balance between the number of dimensions that compose it (i.e., *dimensionality*), and the number of facts that populate it (i.e., *cardinality*). Those candidate MDAS that cannot eventually be populated with facts are called *empty schemas*. In this way, a good statistical model should produce a low rate of empty schemas and should rank higher schemas with high cardinality, at the same time that it should avoid schemas with extreme dimensionality (either too low or too high).

5.1. Components of the model

In order to deal with the incompleteness and heterogeneity of LD we build a statistical model on top of the RDF instance graph and RDF schema graph that will estimate MDAS as defined in Section 4. We assume that at least *type* information is available in the RDF schema graph.

First of all, we generate what we call *typified triples*. Given a LD set with its associated schema triples $\mathcal{T}_S$ and instance triples $\mathcal{T}_I$, $\forall(x\ r_1\ y) \in \mathcal{T}_I$ we derive a typified triple (c_1, r_1, c_2) iff $\mathcal{T}_S \models (x\ a\ c_1) \wedge (y\ a\ c_2)$, $c_1, c_2 \in C$, $r_1 \in P$. We assume the existence of a reasoning system able to infer the classes c_1 and c_2 of instances x and y when not explicitly set [6].

A direct way to estimate the probability of an MDAS consists in defining random walks over the observed transitions between classes and properties in the typified triples. That is, we first estimate the conditional probabilities $p(r_1|c_1)$, $p(c_2|r_1)$ and $p(c_2|c_1)$ for the triple random variables subject (c_1), property (r_1) and object (c_2). In [27], we proposed a preliminary model for estimating the joint probability of a star schema directly over these probabilities, mainly in heterogeneous datasets. Unfortunately, the resulting statistical model is too simple to capture the implicit data constraints. As a consequence, the method produces numerous empty schemas. In this work we propose a more accurate statistical model aimed at capturing the existing data constraints.

In the following, we introduce the components of the statistical model that will guide the generation of MDAS by taking into account both the instance and the schema graph by means of the typified triples.

To estimate fact types, according to Def. 4.1, the probability of a class c acting as a fact type given a dataset $\mathcal{T}_S \cup \mathcal{T}_I$, that is $p(c|\mathcal{T}_S \cup \mathcal{T}_I)$, can be estimated by simply counting the typified triples associated to that class.

Regarding dimension types, in Def. 4.2 we define the structure of a dimension type as an acyclic sequence of classes and properties $(c_1, r_1, c_2, \cdots, r_{n-1}, c_n)$. As previously mentioned, the most likely sequences can be easily generated by random walking over the conditional models $p(r_i|c_i)$, $p(c_{i+1}|r_i)$ and $p(c_{i+1}|c_i)$ [27]. Additionally, in order to take into account existing data constraints between classes, we also estimate the pairwise co-occurence of chaining properties. Thus, for chaining up properties in the way (r_1, c_2, r_2), we estimate the co-occurrence of (r_1, r_2) under the class c_2 by sampling at the instance level, which is denoted as $p(r_1 \rightarrow r_2|c_2)$. Unlike random walk models, which aim at generating candidate dimension types, these distributions are used for checking whether a dimension type is likely to fulfill the instance data constraints or not.

As an MDAS is defined as a fact type and a set of dimension types, we must take care that the dimension types are likely to co-occur at the instance level so that it does not produce an empty schema. With this purpose, we estimate the co-occurrence probability of property pairs (r_i, r_j) under the fact type c by sampling at the instance level, which is denoted as $p(r_i \sim r_j|c)$. These probabilities will be used to generate tight property clusters around each fact type c so that they are likely to generate non-empty schemas.

Table 1 provides a summary of the statistical indicators that compose the model. Next section describes how these estimators can be used to guide the construction of candidate MDAS.

Table 1: Statistical indicators

Function	Statistical indicator
Estimating fact type	$p(c\|\mathcal{T}_S \cup \mathcal{T}_I)$
Generating dimension types (random walks)	$p(r_i\|c_i), p(c_{i+1}\|r_i), p(c_{i+1}\|c_i)$
Generating bases[4]of MDAS	$p(r_i \sim r_j\|c)$
Restricting chaining of props. in dim. types	$p(r_1 \rightarrow r_{i+1}\|c_{i+1})$

5.2. Implementation

In order to generate MDAS from both schema and instance data, we need to estimate likely fact types $F_T = c$ and potential dimension types T_i associated to them. Dimension types are composed by sequences of classes and properties estimated in such a way that represent likely sequences of instances and properties at the instance level. We must ensure as much as possible that there is a fact-dimension relation between the set of facts (F) and dimensions (D). This means that there must be a path connecting each fact $f \in F$ with each dimension value $e \in D_v$ at the instance level. For this matter we use the statistical indicators of the previous section aimed at capturing instance data constraints.

The structure of a MDAS is shown in Figure 3. We approach the problem of generating MDAS in two phases. Given a likely fact type $F_T = c$, the first phase consists in estimating *bases* for c, that is, the core part in bold, which is composed by c and a set of ranked properties that co-occur given c. Each property in the base is ranked according to a measure of the cohesion with the previous properties in the rank. The second phase, shown in dotted lines, estimates potential dimension types. As the inferred MDAS are statistical approximations, we follow this 2-step approach to ensure as much as possible that final stars configured by the user will be populated with facts and dimensions that are connected. The calculation of the base of a star with the ranked properties increases the chances that the different dimensions of each dimension type will overlap at the same fact type instances and result in non-empty schemas. In the following sections, we explain each phase in more detail.

Figure 3: Structure of a MDAS. The bold part corresponds to the base of the star and the dotted part to the dimension types.

[4]The definition of base is given in Def. 5.1

5.2.1. First phase: Generation of bases of MDAS

Generating possible bases of a MDAS consists in estimating sets of ordered properties that most likely co-occur given the fact type class c. We formalize this notion as follows:

Definition 5.1. We define a *base* as a tuple $B = (c, rep, core, opt, sc)$, such that:

- c is the fact type,
- $rep = (r_1, r_2) \in P$ is the representative property pair of the base,
- $core = (R, \prec)$ is the *core* of the base where R is the set of ground properties $r_i \in P$ and $\prec$ is a partial order of R such that $\forall s,t \in R$, $s \prec t$ iff $min(\{p(s \sim r_i|c)_{r_i \in core}\}) < min(\{p(t \sim r_i|c)_{r_i \in core}\})$. Properties in R also satisfy that $\forall r_i, r_j \in core$, $p(r_i \sim r_j|c) >= \alpha \cdot p(r_1 \sim r_2|c)_{r_1,r_2 \in rep}$,
- $opt = (R, \prec)$ is the *optional* part of the base where R is the set of ground properties $r_i \in P$ and $\prec$ is a partial order of R such that $\forall s,t \in R$, $s \prec t$ iff $min(\{p(s \sim r_i|c)_{r_i \in core}\}) < min(\{p(t \sim r_i|c)_{r_i \in core}\})$. Properties in R also satisfy that $\forall r_i, r_j \in opt$, $\beta \cdot p(r_1 \sim r_2|c)_{r_1,r_2 \in rep} <= p(r_i \sim r_j|c) < \alpha \cdot p(r_1 \sim r_2|c)_{r_1,r_2 \in rep}$,
- $sc = min(\{p(r_i \sim r_j|c)\})_{r_i,r_j \in core}$ is the probabilistic score assigned to the base.

From the previous definition we observe that each base has a representative property pair *rep*. The properties that belong to the base are divided into two disjoint ordered sets, the *core* properties, and the *optional* properties. In both sets, properties are ordered by their cohesion score (i.e., co-occurrence probability) with the rest of preceding properties. This provides the analyst a valuable estimation of the degradation in the number of facts as properties further down in the rank are selected. Properties that belong to the core have a higher degree of cohesion among themselves than properties in the optional set, that is, they must satisfy the restriction that their co-occurrence probability pairwise is not inferior to a percentage (α) of the co-occurrence probability of the representative pair. Properties that belong to the optional set have a lower degree of cohesion with the properties of the core, that is, they must satisfy that their co-occurrence probability pairwise with the core properties is in between two percentages of the probability of the representative pair (α and β). These restrictions that enforce cohesion at two different levels are set to aid the analyst in the selection of properties and ensure as much as possible that the resulting base will be populated with

facts. The α and β parameters can be set by the analyst and mark the degradation on the cohesion (i.e., number of facts) which one is willing to accept, that is, if the analyst selects top properties in the core, the resulting star will probably have more facts than if properties from the optional set are selected.

Next, we formally define the cardinality of a base.

Definition 5.2. The *cardinality* of a base B is defined as the number of facts that have all the properties in the core set. That is, $|\{f \in c \,/\, \forall r_i \in core,\, \mathcal{T}_i \models (f\ r_i\ ?x)\}|$

Obviously, the estimation of bases as proposed in Def. 5.1 is an optimistic approximation because we are reducing the probability of generating a set of properties from a fact type to the probability of generating those properties from the fact type pairwise. Moreover, this probability measures the degree of cohesion of the properties of the base rather than the cardinality that eventually will populate the base, which can only be calculated by actually querying the dataset. Nevertheless, results show that this measure is a good indicator of the cardinality of the base.

The algorithm for generating the bases given a fact type c is shown in Algorithm 1. Given a potential base B, we use the functions $rep(B)$, $core(B)$, $opt(B)$ and $sc(B)$ to access its components. The general idea of this algorithm is to create clusters of ordered properties (i.e., bases) by adding to the clusters new property pairs that share a common property with the cluster but with the restriction that the co-occurrence probability of generating the new property together with the rest of the cluster properties pairwise must be over a percentage of the co-occurrence probability of the representative pair of the cluster. Inside a cluster, properties are classified into two groups: core properties, whose cohesion pairwise is higher (above α), and optional properties, whose cohesion with the core properties pairwise is lower (in between α and β).

The algorithm requires a list of property pairs ordered by their importance with respect to the fact type c. Then, we process each property pair (r_1, r_2) and check each candidate base B_i that contains either r_1 or r_2 (lines 6-21). For each candidate, we have the possible new property r_{new} (line 7) to be added. Line 8 checks if the co-occurrence probability of r_{new} with each of the properties in the core of the candidate base is over a percentage w.r.t. the representative pair of the base. If that applies, r_{new} together with its associated score is added to the core of the base (abusing notation) and the score of the base is recalculated (lines 10-12). Otherwise, lines 13 to 19 check if r_{new} can be added to the optional set of properties in the base. If we do not find candidate bases that contain either r_1 or r_2 or the pair cannot be added to a candidate base, we create a new base (lines 23-27). Each base contains the representative pair, the core properties set, the

optional properties set and a score. Line 28 adds the newly created base to the set of bases. As the goal of the paper is to generate useful MDAS, that is, non-empty schemas (with facts associated), we rate each base with a score that is a good estimator of the cardinality of the base. Such score is the minimum co-occurrence probability of the properties pairwise in the core of the base (see lines 11 and 27).

5.2.2. Second phase: Generation of dimension types

The second phase generates dimension types $T_i = (c_1, r_1, c_2, ..., r_{n-1}, c_n)$ that are likely to be populated by instance data. Starting form a concept c_1, we generate each possible dimension type (i.e., path) of length n by applying the statistical model devised in Section 5.1. That is, we generate the paths by random walking over the conditional models $p(r_i|c_i), p(c_{i+1}|r_i), p(c_{i+1}|c_i)$. Additionally, at each transition $(c_i\ r_i)$, we check the co-occurrence of (r_{i-1}, r_i) under the class c_i, that is $p(r_{i-1} \rightarrow r_i|c_i)$, to make sure that we are chaining properties that fulfill the instance data constraints (i.e., they have instance data associated). Through the random walk, we set up the minimum threshold δ_t to avoid walking over transitions that are too low to be considered relevant.

We assign to each new dimension type generated with the previous algorithm the following score:

$$sc(T) = \prod_{n=1}^{len(T)-1} p(r_i|c_i) \cdot p(r_i \rightarrow r_{i+1}|c_{i+1}) \cdot p(c_{i+2}|r_{i+1})$$

We discard dimension types whose score is below some threshold δ_{path} (empirically set), as we consider these dimension types not statistically significant.

Grouping of dimensions. The number of possible dimension types $(c_1, r_1, ..., c_n)$ starting from a fact type can be very large, as datasets tend to be very heterogeneous and we generate each possible combination of classes and properties reflected in the instance data that are statistically significant.

This can overwhelm the analyst, who is in charge of selecting a base for a star and then, a set of dimension types that he/she considers interesting for analysis. Thus, in an attempt to facilitate the task of selecting dimension types and deal with the heterogeneity, we group dimension types into *grouped dimension types*. That is, we group together into the same group all dimension types that share the same fact type c_1 and properties r_i and leave as wildcards the rest of the intermediate classes $c_2, ..., c_n$. The intuition is that paths under the same group express the same relation between the facts in the fact type and the dimension values with only slight variations due to the heterogeneity of the dataset (e.g., using different

Algorithm 1 Generation of bases of MDASs for a fact type $F_T = c$

Require: L: list of property pairs (r_1, r_2) ordered by $p(r_1 \sim r_2|c) \cdot p(r_1|c) \cdot p(r_2|c)$
Ensure: *Bases*: a set of bases

1: $Bases = \emptyset$
2: **for all** $(r_1, r_2) \in L$ **do**
3: $added = False$
4: $B_{cand} = \{B_i \in Bases$ such that $core(B_i) \cap \{r_1, r_2\} \neq \emptyset\}$
5: **if** $B_{cand} != \emptyset$ **then**
6: **for all** $B_i \in B_{cand}$ **do**
7: $r_{new} = \{r_1, r_2\} \setminus core(B_i)$
8: $checkCore = \{r_i \in core(B_i)$ such that $p(r_i \sim r_{new}|c) > \alpha \cdot p(r_1 \sim r_2|c)_{r_1, r_2 \in rep(B_i)}\}$
9: **if** $|checkCore| == |core(B_i)|$ **then**
10: $core(B_i).add((r_{new}, sc(r_{new}, B_i)))$
11: $sc(B_i) = min(\{p(r_i \sim r_j|c)$ such that $r_i, r_j \in core(B_i) \wedge i \neq j\})$
12: $added = True$
13: **else**
14: $checkOpt = \{r_i \in core(B_i)$ such that $\beta \cdot p(r_1 \sim r_2|c)_{r_1, r_2 \in rep(B_i)} <= p(r_i \sim r_{new}|c) < \alpha \cdot p(r_1 \sim r_2|c)_{r_1, r_2 \in rep(B_i)}\}$
15: **if** $|checkOpt| == |core(B_i)|$ **then**
16: $opt(B_i).add((r_{new}, sc(r_{new}, B_i)))$
17: $added = True$
18: **end if**
19: **end if**
20: **end for**
21: **end if**
22: **if** $!added$ **then**
23: $B_{new} = ()$
24: $rep(B_{new}) = \{r_1, r_2\}$
25: $core(B_{new}).add((r_1, sc(r_1, B_{new}))).add((r_2, sc(r_2, B_{new})))$
26: $opt(B_{new}) = ()$
27: $sc(B_{new}) = p(r_1 \sim r_2|c)$
28: $Bases = Bases \cup B_{new}$
29: **end if**
30: **end for**
31: **return** *Bases*

subclasses to refer to the same conceptual entity). As an example, we show in Table 2 a group of dimension types for the fact type `Person` in Dbpedia, where the row in bold represents the dimension group and the rest of rows represent the dimension types that belong to that group. Notice that this a powerful mechanism to discover heterogeneity issues in LD sets and a starting point towards repairing these issues.

Table 2: Example of a grouped dimension type

c_1	r_1	c_2	r_2	c_3	score
Person	**almaMater**	*	**affiliation**	*	
Person	almaMater	University	affiliation	Agent	0.00038
Person	almaMater	CollegeOrUniversity	affiliation	Agent	0.00039
Person	almaMater	EducationalInstitution	affiliation	Agent	0.00031
Person	almaMater	EducationalOrganization	affiliation	Agent	0.00031

Estimation of the aggregation power and filtering. As previously explained, calculating the aggregation power of a dimension type implies solving a query with potentially several joins, which results impractical for large datasets. Thus, we resort to an approximation. In an off-line pre-processing step, we build an index I where we keep the number of instances associated to each typified triple $t_i = (c_{1i}, r_i, c_{2i})$. That is, for each such triple, we calculate the number of instances x of c_{1i} and the number of distinct instances or literals y of c_{2i} that satisfy the triple (i.e., $I[t_i][c_{1i}] = x$ and $I[t_i][c_{2i}] = y$). Notice that for the object of the triple we keep the number of *distinct* instances, as we are interested in discovering groups.

Then, given a dimension type $T = t_1 t_2 ... t_n$, where $t_i = (c_i, r_i, c_{i+1})$ (composed by typified triples), we make a rough estimation of its aggregation power by calculating the ratio $\frac{a}{b}$, where a is the number of instances of the root class (i.e., $I[t_1][c_{11}]$), and b has been calculated by carrying the minimum score at each triple join. That is:

$$MIN(MIN_{1 \leq i \leq n}(I[t_i][c_{2i}], I[t_{i+1}][c_{1(i+1)}]), I[t_n][c_{2n}])$$

For example, having the following statistics about typified triples:

$$I[(Powerplant, fuel\_type, Fuel)][Powerplant] = 12315$$
$$I[(Powerplant, fuel\_type, Fuel)][Fuel] = 30$$
$$I[(Fuel, label, string)][Fuel] = 27$$
$$I[(Fuel, label, string)][string] = 27$$
$$...$$

for the dimension type $(Powerplant, fuel\_type, Fuel, label, xsd:string)$, we have that $a = 12315$ and $b = MIN(MIN(30,27),27) = 27$, The final aggregation power is $\frac{12315}{27} = 456.1$. Notice that this estimation is optimistic, as we are assuming that all the instances of one side of the join will join the instances of the other side.

As explained in the following section, we use the aggregation power of a dimension type to classify it into a potential dimension, measure or attribute.

5.2.3. Composing MDAS

The algorithms presented in this section provide the analyst all the pieces to build MDAS interesting for analysis. Given a fact type of interest, the analyst is presented with a ranking of all bases that have been generated for such fact type ordered by the score. Properties inside a base are also split into two groups (i.e., core and optional properties) and ranked according to their cohesion w.r.t. the preceding properties. After selecting one base and configuring the appropriate properties from its core and optional sets, the analyst can select different dimension types that complement the selected base.

Regarding the presentation of dimension types, although our MD model does not distinguish between dimensions and measures, we make a simple classification of the dimension types into the traditional dimensions, measures and attributes, for the analyst's guidance. We make use of the aggregation power of the dimension type and the type of the sink node. Dimension types ending in numeric data types (i.e., `xsd:integer`, `xsd:float`, `xsd:double`, etc.) and with low aggregation power are considered measures, dimension types ending in `xsd:integer` or `xsd:string` with low aggregation power are considered attributes and the rest (i.e., dimension types ending in classes or datatypes with high aggregation power) are considered dimensions. Dimension types are presented to the analyst in groups as previously explained and ordered by their score.

Once the schema of the MDAS has been selected (i.e., fact type, base properties and dimension types), the population with instance data is trivial, as it comes down to translating the star to a SPARQL query.

5.3. Computational complexity

Here we present a discussion about the computational complexity of the whole process of generating MDASs.

The dataset instance triples $\mathcal{T}_I$ must be scanned in order to generate their typified triples and then to calculate all the statistical indicators presented in Section 5.1. The generation of typified triples depends on the reasoning method used. If all the triples are materialized, the process is $O(|\mathcal{T}_I| \times |C|^2)$, with $|\mathcal{T}_I|$ the number of instance triples and $|C|$ the number of classes, as the inference of the classes for each instance is $O(1)$ but each triple can generate up to $|C|^2$ typified triples ($|C|$ parents for the subject multiplied by $|C|$ parents for the object). However, such a high number of typified triples per triple is highly unlikely in practice. Otherwise, we must also consider the reasoning time for each triple.

The generation of all the statistical indicators of the statistical model for the typified triples is linear with the number of typified triples.

The generation of the bases of the stars is based on the clustering algorithm explained in Section 5.2.1. It is quadratic w.r.t. the number of property pairs (r_1, r_2), as we must ensure before adding a new pair to a cluster that there is probabilistic evidence over a threshold of the pair with the rest of properties in the cluster. Thus, in the worst case it is $O(\binom{|P|}{2}^2) \approx O((k \cdot |P|)^2)$, with P being the number of properties in the dataset.

The generation of dimension types explained in Section 5.2.2 is $O(C^2 \cdot P)$, as it is based on random walks over the transitions of classes and properties.

6. Experiments

In this section we present several experiments that assess the probabilistic method devised to build MDAS from LD sources. We have selected two LD sets with different features to test our method, which are described in turn. Enipedia has already been described in Section 2. Dbpedia 3.9[5] is a community effort to extract structured information from Wikipedia. They also make the information available in RDF. Each dataset has different size and structure. Notice that the collaborative nature of these datasets makes them very heterogeneous in their structure, as well as incomplete. Table 3 shows some statistics about the datasets. We show both the number of triples, and the number of typified triples, which demonstrates the big scale of the scenario.

[5] http://wiki.dbpedia.org/Downloads39

All experiments were run on a server with 96 GB of RAM, eight physical CPUs (Intel Xeon at 2.4 GHz, 32 threads) and using Ubuntu.

Table 3: Datasets statistics

	# triples	# typified triples
Enipedia	4,463,909	8,721,813
Dbpedia	25,896,867	253,599,827

6.1. Generation of the statistical model

The statistical model that we build on top of the RDF/S sources is based on the abstraction provided by the *typified triples*. Based on this abstraction, we are able to generate different statistical indicators. This has been implemented in Python. Table 4 shows the execution times for the generation of the different statistical indicators. As this is a one time process, we have not optimized the calculations. However, the process can be easily parallelized using map-reduce techniques. The execution times are linear w.r.t. the number of typified triples.

Table 4: Execution times for the indicators of the statistical model (in seconds).

Statistical indicators	Enipedia	Dbpedia
Generation of typified triples	176.2	3156.8
$p(c_{i+1}\|c_i)$	85	1603
$p(r_i\|c_i)$	37	1939
$p(c_{i+1}\|r_i)$	38	1894
$p(r_i \sim r_j\|c)$	248	1510
$p(r_1 \rightarrow r_2\|c_2)$	43	340

6.2. Generation of base stars

In the following, we perform several experiments that involve the generation of bases of stars in order to study the behavior of our statistical model when dealing with different datasets.

From both datasets, we have selected a set of representative classes and have generated all the bases of the stars, along with several measures. The parameters α and β can be set up by the analyst as a way to enforce cohesion to a certain limit among properties of the core set and the optional set of a base, respectively. For

the experiments, we set $\alpha = 0.8$ and $\beta = 0.5$. This means that the core properties of a base maintain the co-occurrence probability pairwise higher than the 80% of the co-occurrence probability of the representative pair of the core. On the other hand, properties that belong to the optional set have a co-occurrence probability in between 50% and 80% w.r.t. to the representative pair.

Results are shown in Tables 5 and 6 for Enipedia and Dbpedia datasets, respectively. Columns are explained in turn. The column *Class* shows the class acting as fact type and the level in the class hierarchy as an indicator of the specificity of the class, which is also related to its heterogeneity. The second column, $Total$, shows the total number of bases generated by our algorithm for each class. To assess if the score given to a base is a good indicator of the cardinality of a base (see Def. 5.2), we estimate the correlation of both variables by means of a linear regression model, that is, $Corr(sc(B), |B|)$. The correlation coefficient is shown in the column *Corr*. Based on the estimated parameters of the linear regression (slope and intercept), the column # *pos* shows the number of generated bases whose estimated cardinality according to the regression model is greater than 0, that is, bases whose score indicates that they will contain facts. From this number, we calculate the false positives in column FP, that is, the number of bases estimated to have facts but that actually have no facts associated. Similarly, the column # *neg* shows the number of generated bases whose estimated cardinality is 0, and from these, we calculate false negatives in column FN, that is, the number of bases estimated to have 0 facts but that eventually have more than 50 facts associated. We set the threshold of false negatives to 50 facts per base as a lower number of facts is not representative. Finally, we show in the last column, Sp, the Spearman rank correlation of the score of a base w.r.t. its cardinality, that is $Sp(sc(B), |B|)$, as an alternative to the linear correlation coefficient because this rank takes into account any type of correlation, not only linear.

Table 5: Generation of bases for representative Enipedia class

Class (L)	Total	Corr	# pos	FP	# neg	FN	Sp
Powerplant (1)	133	1	70	0 (0%)	63	6 (9.5%)	0.94
Facility (1)	114	0.88	93	5 (5.4%)	21	0 (0%)	0.77
ChemManufacturer (1)	50	0.70	50	1 (2%)	0	0 (0%)	0.80
Country (1)	47	0.83	47	0 (0%)	0	0 (0%)	0.92
Refinery (1)	14	1	14	0 (0%)	0	0 (0%)	1
Company (1)	9	1	4	0 (0%)	5	0 (0%)	0.86
Fuel (1)	6	1	6	0 (0%)	0	0 (0%)	1

In Table 5 we observe both a high linear correlation and Spearman rank coefficient of the scores associated to the bases of Enipedia classes w.r.t. their true cardinality. This proves that our probabilistic score assigned to a base is a good indicator of its cardinality. Because of the high linear correlation of both variables, the number of estimated FP and FN is also very small, being 0 in most of the cases.

Table 6: Generation of bases for representative Dbpedia classes

Class (L)	Total	Corr	# pos	FP	# neg	FN	Sp
Agent (1)	4359	0.46	4359	1013 (23%)	0	0 (0%)	0.84
Person (2)	1781	0.47	1781	402 (23%)	0	0 (0%)	0.83
Artist (3)	300	0.69	300	60 (20%)	0	0 (0%)	0.89
Actor (4)	155	0.94	100	9 (9%)	55	0 (0%)	0.85
Athlete (3)	624	0.86	624	101 (16.2%)	0	0 (0%)	0.89
Boxer (4)	35	0.97	35	5 (14.3%)	0	0 (0%)	0.96
AmateurBoxer (5)	14	0.97	14	0 (0%)	0	0 (0%)	0.95
Country (2)	71	0.94	44	0 (0%)	27	0 (0%)	0.96
Organization (2)	2497	0.74	2497	605 (24%)	0	0 (0%)	0.84
University (4)	230	0.93	111	0 (0%)	119	5 (4.5%)	0.93
AcademicJournal (4)	58	0.98	58	5 (8.6%)	0	0 (0%)	0.95
Biomolecule (1)	22	0.98	22	1 (4.5%)	0	0 (0%)	0.99
Gene (2)	14	0.98	14	0 (0%)	0	0 (0%)	0.96
HumanGene (3)	12	1	12	0 (0%)	0	0 (0%)	1

We now analyze the results for Dbpedia classes, shown in Table 6. Regarding the linear correlation of the score associated to a base and its real cardinality, bases from more general classes (up in the class hierarchy) have a much lower linear correlation than more specific classes because of their heterogeneity. These classes do not have a clear schema but a mixture of different schemas from its subclasses. Therefore, these classes are more problematic in the sense that they will generate more bases to account for the different schema possibilities. Incompleteness of the schemas is also a factor to be taken into account. General classes have more chances of having a higher rate of incompleteness in their schemas simply due to the variety of schemas they account for. The Spearman rank correlation shows instead a higher correlation for all the classes' bases, meaning that the score of the base and the cardinality have a relationship other than linear. The estimation of FP according to the linear regression model calculated for each class also verifies the initial intuition of general classes being more heterogeneous schema-wise.

Therefore, the rate of FP is higher for bases of these classes. From the analysis of estimated FP and FN, we can conclude that our algorithm for generating bases is optimistic, as the rate of FP is much higher than the rate of FN.

6.3. Generation of dimension types

In this experiment, we generate all possible dimension types of length up until 5 by random walking as explained in Section 5.2.2. The configuration of thresholds has been empirically set up to the following values: $\delta_t = 0.01$, $\delta_{path} = 0.0001$ and $Aggpower = 5$.

Table 7 shows statistics about the generation of dimension types for both datasets. We show the number of dimension types, and the classification into dimensions, measures and attributes according to Section 5.2.3. We also calculate the number of grouped dimensions types, that is, the number of groups that contain dimension types with similar semantics. These groups are built to facilitate the user the selection of dimension types given a base star, as the heterogeneity given in a dataset can give place to a large number of dimension types with similar semantics.

Table 7: Statistics about generation of dimension types (D = dimension, M=measure, A=attribute)

Dataset	# dims	# D	# M	# A	# grouped dims.	time
Dbpedia	10,539	8,870	632	1,037	6,517	7.514s
Enipedia	1,025	462	535	880	28	0.525s

6.4. Examples of MDAS *(bases and dimension types)*

In this section we show examples of bases generated from different concepts, as well as examples of dimension types.

6.4.1. Examples for the Enipedia dataset

Tables 8 and 9 show three different bases rooted in Enipedia concepts. The first two lines of the table show the name of the class acting as fact type, the score associated to the base, *sc*, its real cardinality, *real card*, and its estimated cardinality according to the linear regression model calculated, *est card*. The next block of the table shows the core properties of the base ranked according to their co-occurrence score. As we have previously proven that the score is linearly correlated with the cardinality, for each property in the ranking we show the estimated

Table 8: Base for Powerplant (Enipedia)

Powerplant, $sc = 0.001395$ *real card* = 67703, *est card*=66715	
Core properties	**est card**
Country	72418
Ownercompany	72418
wikiPageSortKey	67338
label	67338
isDefinedBy	67338
Modification_date-23aux	67338
Longitude	67335
Point	67334
Latitude	67333
CarmaId	67229
Intensity2000_kg_CO2_per_MWh_elec	67206
Intensitynextdecade_kg_CO2_per_MWh_elec	67205
Intensity_kg_CO2_per_MWh_elec	67204
Annual_Energyoutput_MWh	67028
Annual_Energyoutputnextdecade_MWh	67028
Annual_Carbonemissions_kg	67028
Annual_Carbonemissions2000_kg	67027
Annual_Carbonemissionsnextdecade_kg	67027
Annual_Energyoutput2000_MWh	67027
State	66715
Optional properties	**est card**
City	57866
Continent	47986

Table 9: Bases for Facility and Country (Enipedia)

Facility, $sc = 0.00197$ *real card* = 105, *est card* = 74	
Core properties	**est card**
label	85
Location	85
isDefinedBy	85
Name	85
Point	84
Longitude	84
Latitude	84
wikiPageSortKey	74
Modification_date-23aux	74
Optional properties	**est card**
Site_size	61
City	58
Website	58
PostCode	58
Telephone	58
Employees	56
Address	49
Uses	44

Country, $sc = 0.00197$ *real card* = 129, *est card* = 70	
Core properties	**est card**
ISO_3166-1_Alpha-3_code	70
NaturalGasImport_m3	70
NaturalGasProduction_m3	70
TechnicallyRecoverableShaleGasResources_m3	70
NaturalGasExport_m3	70
NaturalGasConsumption_m3	70
NaturalGasProvenResources_m3	70
Optional properties	**est card**
GINI	63
OilConsumption_MMbbl	56
OilProvenReserves_MMbbl	56
OilProduction_MMbbl	56

cardinality of the base if it were composed by the upper properties in the ranking and itself, in order to point out the possible degradation in the cardinality of the resulting base as we move down the ranking. Finally, the last block of the table shows the optional properties of the base, which are ranked according to the same criteria as the core properties.

A closer look to the base in Table 8 reveals that this base rooted in class *Powerplant* includes a large set of core properties with a strong cohesion (exactly 20 properties). Such a large base is an indicator of the nature of the fact type class, meaning that resources of type *Powerplant* do not suffer strongly from heterogeneity or incompleteness issues.

The bases shown in Table 9 reveal a similar nature for the classes *Facility* and *Country*, respectively. From the results, we can conclude that such classes also have a well-defined set of core properties associated.

Overall, the experiments performed with Enipedia show that even though this data set has some heterogeneity issues, classes usually have sets of core properties associated, which are well-captured and displayed as bases by our statistical approximation. Notice that the generated bases show a good compromise between the cardinality and the complexity of the schema (number of associated properties).

Regarding examples of dimension types, Table 10 shows potential dimension types for some of the core properties of the *Powerplant* base shown in Table 8. The first column shows the path of the dimension type, the second column shows the aggregation power of the dimension type and the third column shows the multidimensional nature, which can be a dimension (D), attribute (A) or measure (M) according to the classification explained in Section 5.2.3. Notice that dimension types that form a *grouped dimension type* according to Section 5.2.2 are shown together. This grouped visualization is a valuable tool for the analyst when the data set has heterogeneity issues, as it gathers together all the different syntactical options that express a same semantic dimension type.

6.4.2. Examples for the Dbpedia dataset

Table 11 shows several bases that have been generated from the class *Agent* in Dbpedia. This class is specially difficult, as it is used as a wildcard for the subject to express subject-predicate-object relations. Therefore, it has associated many properties, some of them semantically belonging to more specific classes. However, our statistical model is able to differentiate the latent schemas (i.e., sets of properties) that belong to more specific classes. Thus, we have selected four bases as examples, generated from *Agent* with both high score and cardinality, and

Table 10: Possible dimension types for the Powerplant base of Table 8

Dimension type	Agg. power	Type
Country/*		
Country/Country	373	D
Country/Americas	608	D
Country/...	...	D
Ownercompany/*		
Ownercompany/Company	6.85	D
Ownercompany/Energy_Company	6.85	D
OwnerCompany/...	...	D
State/*		
State/State	134	D
State/USState	241	D
State/...	...	D
label/string	1	A
CarmaId/string	1	A
isDefinedBy	1	A
Annual_Carbonemissions_kg/double	1.27	M
Annual_Energyoutput_MWh/decimal	2.66	M
Annual_Energyoutput2000_MWh/decimal	3.57	M

from the semantics of the core properties we are able to infer the potential subclass to which such set of properties refer. The name of the class preceded by the $\sim$ symbol is the potential class that the core properties refer to. In this sense, our statistical model could be used to repair the dataset and associate to the subjects of the triples more specific classes than the ones originally defined.

Regarding other parameters of each of the bases generated in Table 11, we observe that the estimated cardinality is much lower than the real cardinality. This is due to the low correlation coefficient between the score and the cardinality shown in Table 6 for the class *Agent*, whose main reason is the heterogeneity in the usage of the class, that is, the misuse of the class as a wildcard instead of using more specific classes to define relations.

Table 12 shows four examples of bases rooted in the classes *Artist*, *President*, *University* and *AcademicJournal*, respectively. Notice that all the bases shown contain a set of core properties interesting for analysis, at the same time that their estimated cardinality along the ranking is preserved. As previously explained, the deviation between the real and the estimated cardinality depends on the cor-

Table 11: Bases for Agent (Dbpedia)

∼ Singer/Band, $sc = 0.00044$ $real\ card. = 17126,\ est.\ card. = 1854$	
Core properties	**est. card.**
activeYearsStartYear	1967
associatedBand	1967
associatedMusicalArtist	1967
hometown	1854
Optional properties	**est. card.**
background	1657

∼ BaseballPlayer, $sc = 0.000125$ $real\ card. = 17094,\ est.\ card. = 1327$	
Core properties	**est. card.**
formerTeam	1428
throwingSide	1328
position	1328
battingSide	1327
Optional properties	**est. card.**
activeYearsStartDate	1285

∼ Politician, $sc = 0.000084$ $real\ card. = 6127,\ est.\ card. = 1259$	
Core properties	**est. card.**
activeYearsStartDate	1431
party	1306
almaMater	1259
termPeriod	1259
Optional properties	**est. card.**
nationality	1193

∼ MilitaryPerson, $sc = 0.00025$ $real\ card. = 13477,\ est.\ card. = 1536$	
Core properties	**est. card.**
battle	1536
serviceEndYear	1536
Optional properties	**est. card.**
award	1379

relation coefficient of the score and the real cardinality. Thus, higher correlation leads to more accurate cardinality estimations. This is an indicator of the heterogeneity and possible incompleteness of the latent schema/s for a class. In the examples, the classes *President* and *AcademicJournal* have more accurate estimations, meaning that their latent schemas are more homogeneous than those from the other classes.

Regarding dimension types from Dbpedia, Tables 13 and 14 show examples of possible dimension types along with their aggregation power and classification for the bases *AcademicJournal* and *University*, respectively.

7. Related work

The development of new tools and applications that enable the access and exploitation of the ever increasing amounts of LD published on the Web has been recently an active research area. We review two main lines of research, namely (1) general exploitation, querying and visualization of LD; and (2) MD analysis of LD using SW standards.

As far as querying LD is concerned, the SPARQL query language is the defacto standard. However, directly expressing queries in SPARQL is difficult for a non-expert end user because of the complexity and high expressivity of the language, as well as the need to know the structure of the data in advance. To aid the end user in querying tasks over LD, several graph-based query builders have been

Table 12: Bases for Artist, President, University and AcademicJournal

Artist, $sc = 0.00062$ $real\ card. = 356,\ est.\ card. = 1098$	
Core properties	**est. card.**
influencedBy	1760
genre	1144
birthPlace	1098
influenced	1098
Optional properties	**est. card.**
deathPlace	933

President, $sc = 0.00938$ $real\ card. = 1396,\ est.\ card. = 1694$	
Core properties	**est. card.**
birthPlace	1933
orderInOffice	1933
activeYearsStartDate	1812
activeYearsEndDate	1711
successor	1694
Optional properties	**est. card.**
deathPlace	1070

University, $sc = 0.00350$ $real\ card. = 1474,\ est.\ card. = 3913$	
Core properties	**est. card.**
affiliation	5012
type	4437
homepage	4416
country	3913
motto	3913
Optional properties	**est. card.**
state	2779
campus	2681
numberOfStudents	2104
officialSchoolColour	1996

AcademicJournal, $sc = 0.00687$ $real\ card. = 3495,\ est.\ card. = 3048$	
Core properties	**est. card.**
issn	5225
publisher	3258
firstPublicationYear	3258
academicDiscipline	3136
homepage	3136
frequencyOfPublication	3048
Optional properties	**est. card.**
abbreviation	2359
oclc	2183
impactFactorAsOf	1688
impactFactor	1679

developed such as [5], which allow the user to build triple patterns that are internally translated into SPARQL. Still, the main problem is that users do not know the structure of the data upfront, which makes it difficult to express interesting queries. This issue is reinforced by the review performed in [10] about visualization and exploration tools for LD, which states that the majority of the reviewed tools fall short when providing an overview of the data that aids the querying process, at the same time that require technical knowledge.

The exploration and visualization of LD has also been targeted among the scientific community. In particular, many LD browsers with faceted filtering features have been developed, both for general domains [7, 4, 39] and for specific applications [32, 33]. The common issue is that such interfaces neither provide summaries nor support the user in aggregation tasks. Graph-based tools such as RDF-Gravity[6], IsaViz[7] or Relfinder [15] can facilitate the understanding of the underlying data structure by providing the visualization of nodes and their relationships but this graph visualization can suffer from scalability issues.

[6] http://semweb.salzburgresearch.at/apps/rdf-gravity/

[7] http://www.w3.org/2001/11/IsaViz/

Table 13: Possible dimension types for the AcademicJournal base of Table 12

Dimension type	Agg. power	Type
publisher/*		
publisher/Agent	6.65	D
publisher/Publisher/founder/Agent	33.58	D
publisher/...	...	D
fistPublicationYear/gYear	24.67	D
frecuencyOfPublication/string	12.76	D
impactFactorAsOf/gYear	162	D
abbreviation/string	1	A
issn/string	1	A
oclc/string	1	A
impactFactor/double	1.29	M

Table 14: Possible dimension types for the University base of Table 12

Dimension type	Agg. power	Type
affiliation/Agent	5.40	D
country/Place	51.10	D
state/Place	12.04	D
officialSchoolColour/string	2.29	D
motto/string	1.12	A
numberOfStudents/nonNegativeInteger	1.89	M

The approaches mentioned so far enable exploration, querying and visualization of LD but do not address these aspects from an analytical viewpoint. Recently, web-based visualization platforms such as the CODE Query Wizard and Vis Wizard [8] and Payola [22] have been developed to enable lightweight analytic tasks on LD sets. However, the main problem of having to become acquainted with the dataset before expressing the queries still remains.

To shed some light into the problem of providing summaries to the user to aid the analysis of the datasets, we have looked into research approaches focused on summarizing LD. Some of them use bisimulation and clustering techniques [3, 19], whereas others such as [34, 37] are based on statistics and topological

[8] http://code-research.eu/

features. These approaches differ from ours in their main objective: while they are focused on identifying the relevant parts of the dataset, we look for the parts more suited for MD analysis.

The recent ontology-based data access (OBDA) technology is also partially related to this research, in the sense that it provides integration tools based on ontologies [38] to access transparently different data sources and allows end users to express complex information requirements with familiar and comprehensible terms. The work in [12] is an example of OBDA where they integrate concept browsing, facet-based search, and visual query manipulation to allow the user to express their information needs and hide the query language syntax. Behind the scenes, the system builds a SPARQL query. Even though the visual interface facilitates the formulation of queries, the tool does not provide a summary of the dataset nor focuses in analytical aspects.

Regarding MD analysis of SW data, the approach in [26] proposes a semi-automatic method to extract semantic data into a traditional MD database. The method is driven by the user requirements, making necessary some human effort, and the data sources must be enriched with a semantic layer (OWL axioms), which is not always the case.

Recently, terms like *self-service BI* [1], *Situational BI* [25] or *Exploratory OLAP* [2, 16] have emerged to refer to new methods that allow performing OLAP analysis directly over SW MD data. These methods usually rely on previous manual or semi-automatic identification and annotation of the MD elements with vocabularies fitted to that purpose, such as VOID vocabulary [36] for expressing metadata about RDF datasets and QB [9] or QB4OLAP [11] vocabularies to represent RDF MD data. The work in [17, 18] does not perform OLAP analysis directly over the sources but in a traditional MD database. Still, it also requires manual annotation of the MD elements. This contrasts with the method presented in this paper, which focuses on automatically inferring potential MD elements (i.e., dimensions and measures) and patterns for analysis.

Finally, concerning the statistical nature of our approach, we acknowledge existing research that also uses statistical methods over RDF data. The work in [29] focuses on heuristically inferring type information in noisy and incorrect RDF data by using statistical distributions. In [28] they predict properties for resources based on a statistical dataset analysis, in particular, in co-occurrence of properties. The proposals [35] and [24] infer schema axioms from RDF datasets using mining algorithms. The previous approaches are research examples that make use of statistical machinery to enrich the schema of an RDF dataset rather than to infer potential MD schemas for analysis purposes.

8. Conclusions and future work

In this paper, we have presented an automatic approach to generate useful MD analytical patterns from LD sources. The MD patterns, MDAS, are star-shaped patterns at the class level, are based on the semantics of the data (i.e., they provide a conceptual summary of the data), follow the MD model (i.e., information is modeled in terms of facts, dimensions and measures) and are generated following a statistical approach. We have also presented the foundations of the statistical framework that underlies the approach and an implementation that takes into account both the complexity of the generated star schema and its cardinality (i.e., number of facts that will populate the star). The statistical framework has been assessed and validated by a series of experiments with two popular LD sets.

As future work, we plan to address several aspects that will definitely improve the work. In the first place, we plan to take into account the class hierarchy of a LD set by using only the most specific class of an instance to generate the typified triples. This will greatly reduce the total number of typified triples. Another aspect for future research is to find a better estimator for the cardinalities, especially for heterogeneous datasets, where linear regression models do not seem to fit very well. Regarding the presentation of results, we plan to use a vocabulary such as QB4OLAP to publish the generated MDAS as MD data in the LD cloud.

In broader terms, we would like to investigate further the synergies between LD and Big Data, as we believe LD is part of the Big Data landscape. In particular, the Big Data dimension of variety is a generalization of semantic heterogeneity as studied in the field of the SW, and the idea of LD of advancing the hypertext principle from a web of documents to a web of rich data can help alleviate Big Data variability. As Big Data is also characterized by its volume, further research needs to address the analysis of LD as part of Big Data by using scalable approaches (e.g. statistical methods). In this sense, the work presented in this paper is a starting point in that direction.

9. Acknowledgements

This research has been partially funded by the "Ministerio de Economía y Competitividad" with contract number TIN2014-55335-R. Victoria Nebot was supported by the UJI Postdoctoral Fellowship program with reference PI14490.

References

[1] Alberto Abelló, Jérôme Darmont, Lorena Etcheverry, Matteo Golfarelli, Jose-Norberto Mazón, Felix Naumann, Torben Pedersen, Stefano Bach Rizzi, Juan Trujillo, Panos Vassiliadis, and Gottfried Vossen. Fusion Cubes: Towards Self-Service Business Intelligence. *Int. J. Data Warehous. Min.*, 9(2):66–88, April 2013.

[2] Alberto Abelló, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga Llavori, Victoria Nebot, María José Aramburu Cabo, and Alkis Simitsis. Using Semantic Web Technologies for Exploratory OLAP: A Survey. *IEEE Trans. Knowl. Data Eng.*, 27(2):571–588, 2015.

[3] Anas Alzogbi and Georg Lausen. Similar structures inside RDF-Graphs. In *LDOW*, volume 996 of *CEUR Workshop Proceedings*, 2013.

[4] Samur Araújo and Daniel Schwabe. Explorator: A tool for exploring RDF data through direct manipulation. In *LDOW*, volume 538 of *CEUR Workshop Proceedings*, 2009.

[5] Sören Auer and Jens Lehmann. What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In *Proc. of the 4th European Conference on The Semantic Web: Research and Applications*, ESWC '07, pages 503–517, Berlin, Heidelberg, 2007.

[6] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[7] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and Analyzing linked data on the Semantic Web. In *Proceedings of the 3rd International Semantic Web User Interaction*, 2006.

[8] Dan Brickley and Ramanathan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 10, 2004.

[9] R. Cyganiak and D. Reynolds. The RDF Data Cube Vocabulary (W3C Recommendation), January 2014.

[10] Aba Dadzie and Matthew Rowe. Approaches to Visualising Linked Data: A Survey. *Semant. web*, 2(2):89–124, April 2011.

[11] Lorena Etcheverry, Alejandro A. Vaisman, and Esteban Zimányi. Modeling and Querying Data Warehouses on the Semantic Web Using QB4OLAP. In *DaWaK 2014, Munich, Germany, September 2-4, 2014.*, volume 8646 of *LNCS*, pages 45–56. Springer, 2014.

[12] Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür L. Özçep, and Riccardo Rosati. Optique: Zooming in on big data. *IEEE Computer*, 48(3):60–67, 2015.

[13] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. The dimensional fact model: A conceptual model for data warehouses. *Int. J. Cooperative Inf. Syst.*, 7(2-3):215–247, 1998.

[14] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

[15] Philipp Heim, Steffen Lohmann, and Timo Stegemann. Interactive Relationship Discovery via the Semantic Web. In *Proc. of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'10, pages 303–317, Berlin, Heidelberg, 2010.

[16] Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. Towards exploratory OLAP over linked open data - A case study. In *BIRTE 2013, Riva del Garda, Italy, August 26, 2013, and BIRTE 2014, Hangzhou, China, September 1, 2014, Revised Selected Papers*, volume 206 of *LNBIP*, pages 114–132. Springer, 2014.

[17] Benedikt Kämpgen and Andreas Harth. Transforming statistical linked data for use in OLAP systems. In *I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM Int. Conf. Proc. Series, pages 33–40, 2011.

[18] Benedikt Kämpgen, Seán O'Riain, and Andreas Harth. Interacting with statistical linked data via OLAP operations. In *The Semantic Web: ESWC 2012 Satellite Events, Heraklion, Crete, Greece, May 27-31, 2012. Revised Selected Papers*, volume 7540 of *LNCS*, pages 87–101. Springer, 2012.

[19] Shahan Khatchadourian and Mariano P. Consens. ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud. In *ESWC (2)*, volume 6089 of *LNCS*, pages 272–287. Springer, 2010.

[20] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, 2011.

[21] Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[22] Jakub Klimek, Jirí Helmich, and Martin Necaský. Payola: Collaborative Linked Data Analysis and Visualization Framework. In *The Semantic Web: ESWC 2013 Satellite Events*, volume 7955 of *LNCS*, pages 147–151. 2013.

[23] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[24] Huiying Li and Qiang Sima. Parallel mining of OWL 2 EL ontology from large linked datasets. *Knowledge-Based Systems*, 84:10 – 17, 2015.

[25] Alexander Löser, Fabian Hueske, and Volker Markl. *Situational Business Intelligence*, pages 1–11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[26] Victoria Nebot and Rafael Berlanga. Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868, 2012.

[27] Victoria Nebot and Rafael Berlanga. Towards Analytical MD Stars from Linked Data. In *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21 - 24 October, 2014*, pages 117–125, 2014.

[28] Eyal Oren, Sebastian Gerke, and Stefan Decker. Simple algorithms for predicate suggestions using similarity and co-occurrence. In *Proceedings of the 4th European Conference on The Semantic Web: Research and Applications*, ESWC '07, pages 160–174, Berlin, Heidelberg, 2007. Springer-Verlag.

[29] Heiko Paulheim and Christian Bizer. Type inference on noisy RDF data. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of *LNCS*, pages 510–525. Springer, 2013.

[30] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Inf. Syst.*, 26(5):383–423, 2001.

[31] Juan Manuel Pérez, Rafael Berlanga Llavori, María José Aramburu, and Torben Bach Pedersen. Integrating data warehouses with web data: A survey. *IEEE Trans. Knowl. Data Eng.*, 20(7):940–955, 2008.

[32] M. C. Schraefel, Nigel R. Shadbolt, Nicholas Gibbins, Stephen Harris, and Hugh Glaser. CS AKTive Space: Representing Computer Science in the Semantic Web. In *WWW*, pages 384–392, New York, NY, USA, 2004. ACM.

[33] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. LinkedGeoData: A Core for a Web of Spatial Open Data. *Semantic Web Journal*, 3(4):333–354, 2012.

[34] Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, and Dimitris Plexousakis. RDF Digest: Efficient Summarization of RDF/S KBs. In *The 12th Extented Semantic Web Conference (ESWC2015)*, May 2015.

[35] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'11, pages 124–138, Berlin, Heidelberg, 2011. Springer-Verlag.

[36] W3C. Describing Linked Datasets with the VoID Vocabulary, 2011.

[37] Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology Summarization Based on Rdf Sentence Graph. In *WWW*, pages 707–716. ACM, 2007.

[38] Nansu Zong, Sejin Nam, Jae-Hong Eom, Jinhyun Ahn, Hyunwhan Joe, and Hong-Gee Kim. Aligning ontologies with subsumption and equivalence relations in Linked Data. *Knowledge-Based Systems*, 76:30 – 41, 2015.

[39] Martins Zviedris and Guntis Barzdins. Viziquer: A tool to explore and query sparql endpoints. In *The Semanic Web: Research and App.*, volume 6644 of *LNCS*, pages 441–445. 2011.