Towards Analytical MD Stars from Linked Data

Victoria Nebot and Rafael Berlanga
Computer Languages and Systems, Universitat Jaume I, Castellón, Spain

Keywords: Linked Data, RDF, Multidimensional Models, Statistical Models.

Abstract: While the Linked Data (LD) initiative has given place to open, large amounts of semi-structured and rich data published on the Web, effective analytical tools that go beyond browsing and querying are still lacking. To address this issue, we propose the automatic generation of multidimensional (MD) analytical stars. The success of the 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. Our method is based on probabilistic graphical models and makes use of the statistics about the instance data to generate the MD stars. We present a first implementation, and the preliminary results with large LD sets are encouraging to further work in this direction.

1 INTRODUCTION

During the last years, communities from different areas have published data in the cloud of Linked Data (LD) following the publication guidelines, providing the basis for creating and populating the Web of Data. Currently, there are approximately 13 billion triples over 200 datasets.

The increasing availability of these semi-structured and semantically enriched datasets has prompted the need for new tools able to explore, query, analyze and visualize these semi-structured data (Dadzie and Rowe, 2011). While several different tools such as graph-based query builders, semantic browsers and exploration tools (Auer and Lehmann, 2007; Berners-Lee et al., 2006; Heim et al., 2010; Araújo and Schwabe, 2009) have emerged to aid the user in querying, browsing and exploring LD, these approaches have a limited ability to summarize, aggregate and display data in the form that a scientific or business user expects, such as tables and graphs. Moreover, they fall short when it comes to provide the user an overview of the data that may be of interest from an analytical viewpoint.

LD constitutes a valuable source of knowledge worth exploiting using analytical tools. Business Intelligence (BI) uses the multidimensional (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, exploring the datasets using the available browsers and tools to find MD patterns is cumbersome due to the semi-structured nature of the data and the lack of support for obtaining summaries of the data. Moreover, as the datasets are dynamic their structure may change or evolve, making the one-time MD design approach unfeasible.

In this paper, we aim at discovering candidate MD patterns hidden in LD by suggesting the user MD analytical stars using a statistical approach. A MD analytical star is a MD star-shaped pattern at the concept level that encapsulates an interesting MD analysis (Nebot and Berlanga, 2012). These stars reflect the most relevant patterns in the dataset, as they are calculated from the instance data. Moreover, we ensure that each MD analytical star has a minimum aggregation power, that is, it is able to provide a summary of the data that it represents. By suggesting the user these MD stars from large LD sets we are freeing the user from the cumbersome task of browsing and exploring the data to find interesting analytical patterns.

We summarize our contribution as follows:

- We define the concept of MD analytical star as a mapping of the MD model to LD. That is, we identify the subject of analysis, dimensions and measures that compose a MD analytical star in LD.

Nebot V. and Berlanga R..
Towards Analytical MD Stars from Linked Data.
DOI: 10.5220/0005128701170125
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2014), pages 117-125
ISBN: 978-989-758-048-2
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

- We model the problem of automatically discovering MD analytical stars by means of probabilistic graphical models and use a statistical framework based on instance data to implement it.
- We introduce the notion of aggregation power (similar to the notion of functionality between facts and dimensions in traditional data warehousing) and make an estimation to filter MD analytical stars according to this score.
- We present the first results of our method to extract MD analytical stars over two large and well-known LD datasets.

The structure of the paper is as follows. In Section 2 we review the literature related to the problem of analyzing LD. Section 3 presents the main foundations that underlie our approach. In Section 4 we present a model for MD analytical stars over LD sources. Section 5 contains the implementation and in Sections 6 and 7 we present the first results and give some conclusions and future work.

2 RELATED WORK

We have performed a thorough review on the literature from an analytical viewpoint to find out that the majority of approaches use querying, exploration and only light-weight analytics over LD.

For querying LD, SPARQL has become the de-facto standard. However, directly querying a dataset using SPARQL interface cannot be considered an end user task as it requires familiarity with its syntax and the structure of the underlying data. Graph-based query builders such as (Auer and Lehmann, 2007) can help users build triple patterns by using auto-completion to express queries. However, users do not always have explicit queries upfront, but need to explore the available data first in order to find out what information might be interesting to them. Sgvizler[1] allows to render results of SPARQL queries as charts, maps, etc. However, it requires SPARQL knowledge and focuses only on the visualization part.

The review in (Dadzie and Rowe, 2011) about visualization and exploration of LD concludes that most of the tools are designed for technical users and do not provide an overview or summary on the data.

LD browsers such as (Berners-Lee et al., 2006; Araújo and Schwabe, 2009; Zviedris and Barzdins, 2011) are designed to display one entity at a time and do not support the user in aggregation tasks. Most of them use faceted filtering to better guide the user in exploration tasks. However, the user gets overview of only a small part of the dataset. On the other hand, browsers such as (Schraefel et al., 2004) and (Stadler et al., 2012) provide a more powerful browsing environment, but are tailored to a specific application.

Graph-based tools such as RDF-Gravity[2], IsaViz[3] or Relfinder (Heim et al., 2010) provide node-link visualizations of the datasets and the relationships between them. Although this approach can help obtain a better understanding of the data structure, in some cases graph visualization does not scale well to large datasets.

The CODE Query Wizard and Vis Wizard developed under the CODE project[4] are a web-based visual analytics platform that enables non-expert users to easily perform exploration and lightweight analytic tasks on LD. Still, the user has to browse the data to find interesting analytical queries. Payola (Klimek et al., 2013) is a framework that allows any expert user to access a SPARQL endpoint, perform analysis using SPARQL queries and visualize the results using a library of visualizers.

We claim that existing tools for exploration and analysis of LD provide little or no support for summaries so that the user can have an idea of the structure of the dataset and the parts that seem more interesting for analysis. In that line, we have also looked into approaches that provide graph summaries over LD using different techniques such as bisimulation and clustering (Alzogbi and Lausen, 2013; Khatchadourian and Consens, 2010). However, these graph summaries are produced without an analytical focus, therefore, the resulting summaries may not be useful for analysis purposes.

Recently, there have been some attempts to analyze LD that go beyond querying and browsing. (Nebot and Berlanga, 2012) proposes MD analysis over LD under the OWL formalism. Other approaches (Kämpgen and Harth, 2011; Etcheverry and Vaisman, 2012) have proposed MD analysis over LD relying on the previous manual annotation of the MD elements (dimensions and measures) and using previously defined MD vocabularies.

3 FOUNDATIONS

In this section, we review the main foundations that underlie our approach.

[1]http://dev.data2000.no/sgvizler/

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

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

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

3.1 Linked Data

LD is a set of common practices and general rules to contribute to the Web of Data (Heath and Bizer, 2011). The basic principles are that each entity should be assigned 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.

The most adopted standard to implement the Web of Data is RDF (Klyne and Carroll., 2004), which allows us to make statements about entities. It assumes data modeled as triples with three components: subject, predicate and object. We consider only valid RDF triples using URIs (U), blank nodes (B) and literals (L). These triples can also be viewed as graphs, where vertices correspond to subjects and objects, while labeled edges represent the triples themselves. SPARQL (Prudhommeaux and Seaborne, 2008) has become the standard for querying RDF data and it is based on the specification of triple patterns.

In RDF there is no technical distinction between the schema and the instance data, even though it provides terminology to express class membership (`rdf:type`). The RDFS extension allows to create taxonomies of classes and properties. It also extends definitions for some of the elements of RDF, for example it sets the domain and range of properties and relates the RDF classes and properties into taxonomies using the RDFS vocabulary. OWL extends RDFS and allows for expressing further schema definitions in RDF. The formal semantics of RDFS and OWL enrich RDF with implicit information that can be reasoned over. Throughout the paper, we refer both to the explicit and implicit triples, which have been derived using some reasoning mechanism. We use the naming convention of OWL referring to classes, properties and individuals to homogenize terminology.

3.2 Multidimensional Models

The MD model is the conceptual abstraction mostly used in BI. The observations or facts are analyzed in terms of dimensions and measures (Kimball and Ross, 2011). They focus on a subject of analysis (e.g., sales) and define a series of dimensions or different analysis perspectives (e.g., location, time, product), which provide contextual information. Facts are aggregated in terms of a series of measures (e.g., average sales). As a result, analysts are able to explore and query the resulting data cube applying OLAP operations. A typical query would be to display the evolution of the sales during the current year of personal care products by city.

BI has traditionally been applied to internal, corporate and structured data, which is extracted, transformed and loaded into a pre-defined and static MD model. The relational implementation of the MD data model is typically a star schema, where the fact table containing the summarized data is in the center and is connected to the different dimension tables by means of a functional relation.

3.3 Bayesian Networks

A Bayesian network provides a graph theoretic representation to compactly represent the joint probability distribution of random variables in a problem domain. Formally, a Bayesian network consists of two components: structure and parameters. The structure is represented as a directed acyclic graph (DAG) G which consists of a set of vertices $V$and a set of edges E that connects these vertices, that is, $G = (V,E)$. The set of vertices corresponds to random variables in a problem domain while the set of edges defines certain types of conditional dependency among these variables. Parameters, on the other hand, describe conditional probability distribution of each variable given its parents in G. These conditional probabilities together with the Markov property assumption, that is , each variable x_i is independent of its non-descendent given its parents, simplifies the computation of joint probability distribution of these variables.

$$P(x_1,x_2,...,x_n) = \prod_{i=1}^{n} P(x_i|pa(x_i)) \qquad (1)$$

where $pa(x_i)$ represents the set of parents of variable x_i.

4 MD ANALYTICAL STARS

In this section we explain how we model MD analytical stars from LD.

We formalize the representation of an RDF graph using graph notation.

Definition 4.1. (RDF graph) An RDF graph G is a labeled directed graph $G = \langle V,E,\lambda \rangle$ where:

- V is the set of nodes, let V^0 denote the nodes in V having no outgoing edge, and let $V^{>0} = V \backslash V^0$;
- $E \subseteq V \times V$ is the set of directed edges;
- $\lambda : V \cup E \to U \cup B \cup L$ is a labeling function such that $\lambda_{|V}$ is injective, with $\lambda_{|V^0} : V^0 \to U \cup B \cup L$ and $\lambda_{|V^{>0}} : V^{>0} \to U \cup B$, and $\lambda_{|E} : E \to U$.

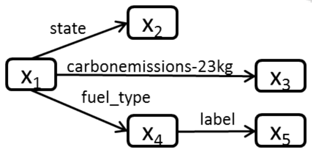
Typical analysis usually involves investigating a set of particular facts according to relevant criteria (dimensions) and measurable attributes (measures). Here, we use the notion of basic graph pattern (BGP) queries, which is a well-known a subset of SPARQL. A BGP is a set of triple patterns, where each triple has a subject, predicate and object, some of which can be variables. We are specially interested in rooted BGP queries, as they resemble the star-shaped pattern typical of MD analysis.

Definition 4.2. (Rooted query) Let q be a BGP query, $G = \langle V, E, \lambda \rangle$ its graph and $v \in V$ a node that is a variable in q. The query q is rooted in v iff G is a connected graph and any other node $v' \in V$ is reachable from v following the directed edges in E.

Example 4.1. (Rooted query) The query q is a rooted BGP query, with x_1 as root node.

$$q(x_1,x_2,x_3,x_5) \text{ :- } x_1 \text{ carbonemissions23kg } x_3, \; x_1 \text{ state } x_2, \; x_1 \text{ fuel\_type } x_4, \; x_4 \text{ label } x_5$$

The query's graph representation below shows that every node is reachable from the root x_1.



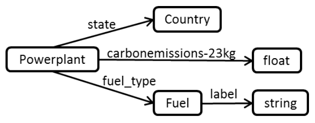
Even though rooted queries express data patterns by means of the predicate chains, these are still vague as the variable nodes can match any element in $U \cup B \cup L$. To narrow down the scope of the patterns we define the notion of typified rooted queries as follows:

Definition 4.3. (Typified rooted query) A typified rooted query q' is a rooted query with graph $G = \langle V, E, \lambda \rangle$ where each variable node $v_x \in V$ has an associated class or datatype. That is, each variable v_x has an outgoing edge (v_x, v_y) such that $\lambda((v_x, v_y)) =$ `rdf:type` and $\lambda(v_y) \in U$ and v_y has an outgoing edge (v_y, v_z) such that $\lambda((v_y, v_z)) =$ `rdf:type` and $\lambda(v_z) \in \{$`rdfs:Class`, `rdfs:Datatype`$\}$.

Example 4.2. (Typified rooted query) The previous query q can be typified as follows:

$q(x_1,x_2,x_3,x_5)$:- x_1 rdf:type Powerplant,
Powerplant rdf:type rdfs:Class, x_1 state x_2,
x_2 rdf:type Country, Country rdf:type rdfs:Class,
x_1 carbonemissions23kg x_3, x_3 rdf:type xsd:float,
xsd:float rdf:type rdfs:Datatype,x_1 fuel_type x_4,
x_4 rdf:type Fuel, Fuel rdf:type rdfs:Class,
x_4 label x_5, x_5 rdf:type xsd:string,
xsd:string rdf:type rdfs:Datatype

From now on, we omit the type edges and represent typified rooted queries with the (data)type's name in the variable node.



It is immediate to see that a typified rooted query is composed by a set of typified paths that go from the root to a sink node (node with no outgoing edges). The root node represents a class and the sink node represents either a class or a datatype. We formalize this notion next:

Definition 4.4. (Typified path) Given a typified rooted query q with x_1 as root node and graph $G = \langle V, E, \lambda \rangle$, a typified path is a sequence $p = c_1 - r_1 - c_2 - r_2 - ... - r_{n-1} - c_f$ where $\lambda(x_1) = c_1$ is the root class, c_f is a sink class or datatype, every r_i is a property and every c_i has an associated class.

Example 4.3. (Typified path) In the previous query q we can identify the following typified paths:

(Powerplant, state, Country)
(Powerplant, carbonemissions-23kg, float)
(Powerplant, fuel_type, Fuel, label, string)

In MD modeling it is important the many-to-one relation between facts and dimensions to ensure aggregation power. That is, one fact must be associated with one dimension value, whereas a dimension value can and should be associated to multiple facts. In our LD scenario, we define the aggregation power of a typified path as follows:

Definition 4.5. (Aggregation power) Given a typified path $p = c_1 - r_1 - c_2 - r_2 - ... - r_{n-1} - c_f$, the aggregation power is calculated as the ratio between the number of individuals of the root class c_1 and the number of different individuals (or literals) of the sink class (or datatype) c_f that satisfy the path.

Example 4.4. (Aggregation power) Given the number of individuals (or literals) in parenthesis that satisfy the underlying queries of the paths, we show their aggregation power:

(Powerplant$_{(160)}$, state, Country$_{(13)}$) $\rightarrow$ 12.3
(Powerplant$_{(11994)}$, fuel_type, Fuel,
label, string$_{(27)}$ $\rightarrow$ 444.2

Notice that, in order to exactly calculate the aggregation power, one must execute the query corresponding to the typified path, which can imply several joins.

As this is expensive and impractical, we have devised a rough estimation of the aggregation power that will be shown in the next section.

We are now ready to introduce MD analytical stars. For this, we make use of traditional data warehousing terminology. We use the notion of *classifier* to denote the level of data aggregation, that is, the classifier defines the dimensions according to which the facts will be analyzed. The *measure* allows obtaining values to be aggregated using *aggregation functions*.

Definition 4.6. (MD analytical star) Given an RDF graph $G = \langle V, E, \lambda \rangle$, a MD analytical star rooted in the node $x \in V$ is a triple: $S = \langle c(x, d_1, ..., d_n), m(x, v), \bigoplus \rangle$ where:

- $c(x, d_1, ..., d_n)$ is a typified query rooted in the node r_c of its graph G_c, with $\lambda(r_c) = x$ and each path $x - ... - d_i$ is a typified path. This is the classifier of x w.r.t. the n dimensions $d_1, ..., d_n$. The node x is the subject of analysis.
- $m(x, v)$ is a typified query rooted in the node r_m of its graph G_m, with $\lambda(r_m) = x$. This query is only composed by a typified path $x - ... - v$[5]. This is called the measure of x.
- $\bigoplus$ is an aggregation function over a set of values, that is, the aggregator for the measure of x w.r.t. its classifier.
- Each of the typified paths of the classifier has an aggregation power over a threshold δ.

Notice that typified rooted queries (and therefore, typified paths) are the building block to suggest MD analytical stars.

Example 4.5. (MD analytical star) The MD analytical star below asks for the average of carbon emission of powerplants, classified by country and fuel type.

$$\langle c(x, x_1, x_3), m(x, x_4), average \rangle$$

where the classifier and measure queries are:

$$\begin{aligned} c(x, x_1, x_3) :- \; & x \text{ rdf:type Powerplant}, \\ & x \text{ state } x_1,\; x_1 \text{ rdf:type Country}, \\ & x \text{ fuel\_type } x_2,\; x_2 \text{ rdf:type Fuel}, \\ & x_2 \text{ label } x_3,\; x_3 \text{ rdf:type xsd:string} \\ m(x, x_4) :- \; & x \text{ rdf:type Powerplant }, \\ & x \text{ carbonemissions-23kg } x_4, \\ & x_4 \text{ rdf:type xsd:float} \end{aligned}$$

The answer to an MD analytical star is a set of tuples of dimension values found in the answer of the classifier query, together with the aggregated result of the measure query. Therefore, it can be represented as a cube of n dimensions, where each cell contains the aggregated measure. However, query processing and answering is out of the scope of this paper.

[5]For the sake of simplicity, we assume that an MD analytical star has only one measure.

5 IMPLEMENTATION

In order to discover MD analytical stars we have to find both the classifier and measure typified queries rooted in a potential class acting as subject of analysis. These typified queries are composed by typified paths (from now on we call them simply paths) with a certain aggregation power. In this section we summarize all the steps followed to obtain MD analytical stars.

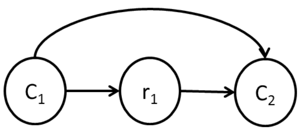
5.1 Generation of Typified Triples

The paths that will form the MD analytical stars are composed by *typified triples*. These are generated in a pre-processing step as follows:

Given a LD set D, $\forall c_1, c_2 \in D$ such that $D \models c_1(x), c_2(y), (x, r_1, y)$, the triple (c_1, r_1, c_2) is generated. We assume the existence of a reasoning system able to infer the classes c_1 and c_2 of individuals x and y.

5.2 Joint Probabilities of Typified Paths

We now focus on the automatic discovery of paths from instance data starting from any potential subject of analysis. We model the problem of finding paths using probabilistic graphical models. In particular, we model the probability of finding a path of length one (c_1, r_1, c_2), where c_1, c_2 are classes and r_1 is a property, as a Bayesian network where the random variables are the subject class c_1, the property r_1 and the object class c_2. We observe that the property depends on the subject, and the object depends on both the subject and the property.


$$\begin{tikzcd} c_1 \arrow[r] \arrow[rr, bend left] & r_1 \arrow[r] & c_2 \end{tikzcd}$$

To get the probability of observing a path, $p(c_1, r_1, c_2)$, we factorize this probability using bigrams.

$$\begin{aligned} p(c_1, r_1, c_2) = & p(r_1|c_1) * p(c_2|r_1, c_1) = \\ & p(r_1|c_1) * p(c_2|r_1) * p(c_2, c_1) \end{aligned} \quad (2)$$

To extend it to paths of arbitrary length, we simply multiply the probabilities of the subsequent paths of length one.

$$p(c_1,r_1,c_2,r_2,...,c_n) = p(c_1,r_1,c_2)* p(c_2,r_2,c_3)*...* p(c_{n-1},r_{n-1},c_n) \quad (3)$$

The joint probability of each class and property is calculated from the collection of typified triples generated in the previous step. We estimate bi-gram probabilities in the collection of typified triples through MLE as follows:

$$p(w_2|w_1) = \frac{count(w_1,w_2)}{count(w_1)} \quad (4)$$

to estimate the probabilities $p(r_1|c_1)$, $p(c_2|r_1)$ and $p(c_2,c_1)$ for each typified triple (c_1,r_1,c_2).

5.3 Typified Paths Generation

In this step we automatically build paths that will potentially compose MD analytical stars. We explore the underlying schema graph based on the previous estimated joint probabilities using a depth-first search approach, shown in Algorithm 1. Starting from each class (line 3 of `GetPaths`), the algorithm creates a one node length path, *path*, composed by the source class only and initializes the list *probs* to keep the joint probability of each element in *path*. At each successful recursion step in procedure `GetPathsRec`, the algorithm extends the current explored path, p, with a property r and a class or datatype node o based on the joint probabilities $p(r|c)$ (line 5), $p(o|r)$ (line 8) and $p(o,c)$ (line 10). If these probabilities are over some thresholds, the path is extended (line 14) and added to the set of paths with an associated score (line 16). This score is the result of the product of the joint probabilities of each element in the path. Finally, if the latter added node is a class, the path is recursively extended (line 19). In order to prune the search, we have devised several thresholds. THR_{prob} filters out joint probabilities that are too low to be considered statistically significant. THR_{sc} filters out paths whose final score is too low to be considered relevant. The relative threshold avg_{prob} (line 1) calculates at each recursion step the average of the joint probabilities of each element in the current path. It is used to not extend paths with elements whose joint probability is lower than this threshold. This way, we avoid a possible incompleteness in the results when the probability is lower than the average.

Algorithm 1: Typified paths discovery.

```
Procedure GETPATHS(D)
Input: D:dataset
Output: paths: tuples (path, score)
 1: global $paths = \{\}$
 2: global $D$
 3: for $c \in D$, c is concept do
 4:     $path = c$
 5:     $probs = []$
 6:     GETPATHSREC($c$,$path$,$probs$)
 7: return $paths$
Procedure GETPATHSREC(c,p,probs)
Input: c: concept, p: current path, probs: path probs
 1: $avg_{prob} = getAvg(probs)$
 2: if $c$ in $p$ then                                   ▷ cycle
 3:     $paths = paths \cup (p, sc)$
 4: for $r \in D$, r is role do
 5:     $p_{rc} = p(r|c)$
 6:     if $p_{rc} < THR_{prob} \vee p_{rc} < avg_{prob}$ then return
 7:     for $o \in D$, o is concept do
 8:         $p_{or} = p(o|r)$
 9:         if $p_{or} < THR_{prob} \vee p_{or} < avg_{prob}$ then return
10:         $p_{oc} = p(o|c)$
11:         if $p_{oc} < THR_{prob} \vee p_{oc} < avg_{prob}$ then return
12:         $new_{sc} = getScore(probs + [p_{rc}, p_{or}, p_{oc}])$
13:         if $new_{sc} < THR_{sc}$ then return
14:         $new_p = p + [r, o]$
15:         if $(new_p, new_{sc}) \notin paths$ then
16:             $paths = paths \cup (new_p, new_{sc})$
17:         if $o$ is concept then
18:             $new_{probs} = probs + [p_{rc}, p_{or}, p_{oc}]$
19:             GETPATHSREC($o$, $new_p$, $new_{probs}$)
```

5.4 Typified Paths Filtering by Aggregation Power

Once we have obtained a set of paths from a source class, we filter out those which do not pass the aggregation power threshold δ, which has been set up to 4. As previously said, calculating the aggregation power of a path implies solving a query with potentially several joins, and it results impractical to precalculate all possible paths with their associated aggregation power score. Thus, in an off-line pre-processing step, we build an index I where we keep the number of individuals associated to each typified triple $t_i = (c_{1i}, r_i, c_{2i})$. That is, for each such triple, we calculate the number of individuals x of c_{1i} and the number of distinct individuals 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 we keep the number of distinct individuals, as we are interested in discovering groups.

Then, given a path $p = t_1 t_2 ... t_n$ (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 individuals 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:

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

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 path (Powerplant, fuel_type, Fuel, label, xsd:string), we have that $a = 12315$ and $b = 27$, The final aggregation power is $\frac{12315}{27} = 456.1$. Notice that this estimation is optimistic, as we are assuming that all the elements of one side of the join will join with the other elements' side.

5.5 Composing MD Analytical Stars

Finally, from the remaining paths, we select those that will compose the classifier (i.e., dimensions) and a set of possible measures. For this we use the aggregation power of the path and the type of the sink node. Paths ending in numeric datatypes (i.e., `xsd:integer`, `xsd:float`, `xsd:double`, etc.) and with low aggregation power are considered measures, whereas the rest of paths (i.e., paths ending in classes or datatypes with high aggregation power) are considered dimensions and thus, are part of the classifier query.

Currently, the generated MD analytical stars are ranked based on an eigenvector centrality algorithm. That is, stars whose subject of analysis is ranked higher are displayed first (Zhang et al., 2007).

Computational complexity
Here we present a discussion about the computational complexity of the whole process of generating MD analytical stars. All the triples must be scanned in order to generate their typified triples and calculate their joint probabilities. The generation of typified triples depends on the reasoning method used. If all the triples are materialized, the process is $O(N \times C^2)$, with N the number of triples and C the number of classes, as the inference of the classes for each instance is $O(1)$ but each triple could generate up to C^2 typified triples (C parents for the subject times C parents for the object). However, this is highly unlikely in practice. Otherwise, we must also consider the reasoning time for each triple. The generation of the joint probabilities for the typified triples is linear with the number of typified triples. The generation of typified paths (Algorithm 1) is $O(C \times (R+C))$, with C being the number of classes and R be the number of properties. Such algorithm is an implementation of DFS (i.e., it has complexity $O(R+C)$) and it is invoked once for each class. Notice that our implementation does not compute all the possible paths between two classes, but only those which pass the thresholds, thus the complexity for the DFS is much lower than $O(R+C)$. The filtering of the paths according to their aggregation power is $O(n \times P)$, with P being the number of paths and n the maximum length of a path, as the estimation of the aggregation power for each path depends on its length. Finally, the composition of MD analytical stars is $O(P)$.

6 PRELIMINARY RESULTS

In this section we present the first results of our method to build MD analytical stars from LD. We have selected three LD datasets with different features to test our method. BioPAX[6] is an emerging format for sharing biological pathway data. We select the homo sapiens pathways, which are available in RDF format. Enipedia[7] is an initiative aimed at providing a collaborative environment through the use of wikis and the Semantic Web for energy and industry issues. They provide energy data from different open data sources structured and linked in RDF. Dbpedia 3.9[8] 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. Biopax is the smallest one but is richer in terms of semantics, whereas Enipedia and Dbpedia are much larger but less semantically rich. Table 1 shows some statistics about the datasets. We show both the number of triples, and the number of typified triples (see Section 5.1), which demonstrates the big scale of the scenario.

Table 1: Datasets statistics.

	# triples	# typified triples
Biopax	600,874	499,152
Enipedia	4,463,909	8,721,813
Dbpedia	25,896,867	253,599,827

To generate typified paths, we have empirically set up THR_{prob} and THR_{sc} to 10^{-2} and 10^{-5}, respectively. The aggregation power threshold δ is set up to

[6]http://www.biopax.org/
[7]http://enipedia.tudelft.nl/wiki/Main_Page
[8]http://wiki.dbpedia.org/Downloads39

Table 3: MD analytical stars (Biopax).

Subject	Paths (dimensions and measures)
Pathway	name, xsd:string
(1,113)	organism, bioSource
	pathway-components, pathwayStep, step-interactions, pathway
	xref, unificationXref, db, xsd:string
Catalysis	direction, xsd:string
(2,125)	control-type, xsd:string
	controller, physicalEntityParticipant, cellular-location, openControlledVocabulary
	controlled, biochemicalReaction
BiochemicalReaction	name, string
(4,162	right, physicalEntityParticipant
	left, physicalEntityParticipant

Table 2: Statistics about paths and MD stars.

	# typ. paths	# filt. paths	# stars
Biopax	329	129 (39%)	20
Enipedia	643	369 (57%)	54
Dbpedia	12395	5809 (47%)	377

4. Table 2 shows the number of typified paths, paths after filtering by aggregation power and the number of analytical MD stars generated for each dataset.

We emphasize the usefulness of calculating typified paths as a means to capture and group together instance data paths under the same semantic concepts, providing a summarized overview of the structure of the datasets. Yet, the amount of typified paths is too large to be managed by an analyst. The aggregation power property filters out about half of the typified paths for Enipedia and Dbpedia datasets and even more for Biopax. Finally, we observe that the number of MD analytical stars generated for each dataset is manageable, compared to the dataset's size, and provides the analyst different interesting MD analysis patterns. In Tables 3, 4 and 5 we show some excerpts of MD analytical stars for the three datasets. Notice that not all the paths conforming the MD stars are shown for space reasons. Paths in italics correspond to potential measures, whereas the rest act as dimensions. The number in between parenthesis is an estimation of the number of facts that satisfy the shown star.

Table 4: MD analytical stars (Enipedia).

Subject	Paths (dimensions and measures)
Powerplant	*Energyoutputnextdecade-23J, xsd:double*
(34,632)	*Energyoutput-23J, xsd:double*
	Carbonemissions-23kg, xsd:double
	country, Country
	ownercompany, Company
Country	*NaturalGasProvenResources-23m3, xsd:double*
(130)	*NaturalGasProduction-23m3, xsd:double*
	NaturalGasConsumption-23m3, xsd:double
	label, xsd:string
NaturalGas-	*NumberOfTanks, xsd:double*
Interconnector	*StorageCapacityLNG-23m3, xsd:double*
(106)	label, xsd:string
	GasFlowsFrom, Country
	GasFlowsTo, Country

Table 5: MD analytical stars (Dbpedia).

Subject	Paths (dimensions and measures)
Person	birthPlace, Place
(795,564)	deathPlace, Place
	team, SportsTeam
	birthDate, date
MusicGroup	hometown, Place
(52,882)	genre, Genre
	recordLabel, Company
Educational-	*numberOfStudents, xsd:integer*
Institution	city, Place
(19,741)	country, Country
	foundingYear, xsd:gYear
Work	*runtime, xsd:double*
(167,880)	starring, Person
	producer, Person
	writer, Person
	director, Person
	genre, Genre
City	*elevation, xsd:double*
(36,268)	*areaTotal, xsd:double*
	areaLand, xsd:double
	areaWater, xsd:double
	populationDensity, xsd:double
	isPartOf, Place, country, Country
Film	starring, Person
(65,826)	writer, Person
	musicComposer, Person
	director, Person
	director, Person, birthplace, Place
Automobile	*wheelbase, xsd:double*
(4,333)	*length, xsd:double*
	transmission, xsd:string
	assembly, Place
	productionStartYear, xsd:gYear
	productionEndYear, xsd:gYear
	manufacturer, Company
	engine, AutomobileEngine

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the first automatic approach towards providing useful MD analytical patterns from LD sources. The MD patterns 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 analysis dimensions and measures) and are extracted following a statistical approach. As this is preliminary work, there is much room for improvement. In a near future, we would like to explore more sophisticated and foundational relative thresholds that help prunning the generation of paths. Also, we would like to devise a ranking algorithm for the MD stars that takes into account both the semantic and analytical relevance of the star. Finally, as some of the stars are composed by many semantically similar paths, it would be interesting to further group these paths into semantic dimensions to get an even more summarized view of the generated stars.

REFERENCES

Alzogbi, A. and Lausen, G. (2013). Similar structures inside rdf-graphs. In *LDOW*, volume 996 of *CEUR Workshop Proceedings*.

Araújo, S. and Schwabe, D. (2009). Explorator: A tool for exploring rdf data through direct manipulation. In *LDOW*, volume 538 of *CEUR Workshop Proceedings*.

Auer, S. and Lehmann, J. (2007). 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.

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., and Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction*.

Dadzie, A. and Rowe, M. (2011). Approaches to visualising linked data: A survey. *Semant. web*, 2(2):89–124.

Etcheverry, L. and Vaisman, A. A. (2012). Enhancing olap analysis with web cubes. In *ESWC*, volume 7295 of *Lecture Notes in Computer Science*, pages 469–483.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers.

Heim, P., Lohmann, S., and Stegemann, T. (2010). 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.

Kämpgen, B. and Harth, A. (2011). 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.

Khatchadourian, S. and Consens, M. P. (2010). Explod: Summary-based exploration of interlinking and rdf usage in the linked open data cloud. In *ESWC (2)*, volume 6089 of *Lecture Notes in Computer Science*, pages 272–287. Springer.

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

Klimek, J., Helmich, J., and Neask, M. (2013). Payola: Collaborative linked data analysis and visualization framework. In *The Semantic Web: ESWC 2013 Satellite Events*, volume 7955 of *Lecture Notes in Computer Science*, pages 147–151.

Klyne, G. and Carroll., J. J. (2004). Resource description framework (RDF): Concepts and abstract syntax.

Nebot, V. and Berlanga, R. (2012). Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868.

Prudhommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF.

Schraefel, m. c., Shadbolt, N. R., Gibbins, N., Harris, S., and Glaser, H. (2004). CS AKTive Space: Representing computer science in the semantic web. In *WWW*, pages 384–392, New York, NY, USA. ACM.

Stadler, C., Lehmann, J., Höffner, K., and Auer, S. (2012). Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 3(4):333–354.

Zhang, X., Cheng, G., and Qu, Y. (2007). Ontology summarization based on rdf sentence graph. In *WWW*, pages 707–716. ACM.

Zviedris, M. and Barzdins, G. (2011). Viziquer: A tool to explore and query sparql endpoints. In *The Semanic Web: Research and App.*, volume 6644 of *Lecture Notes in Computer Science*, pages 441–445.