Quantifying the Connectivity of a Semantic Warehouse

Yannis Tzitzikas[1,2], Nikos Minadakis[1], Yannis Marketakis[1],
Pavlos Fafalios[1,2], Carlo Allocca[1], Michalis Mountantonakis[1,2]
[1] Institute of Computer Science, FORTH-ICS, GREECE, and
[2] Computer Science Department, University of Crete, GREECE
{tzitzik,minadakn,marketak,fafalios,carlo,mountant}@ics.forth.gr

ABSTRACT

In many applications one has to fetch and assemble pieces of information coming from more than one SPARQL endpoints. In this paper we describe the corresponding requirements and challenges, and then we present a process for constructing such a semantic warehouse. We focus on the aspects of *quality* and *value* of the warehouse, and for this reason we introduce various metrics for quantifying its *connectivity*, and consequently its ability to answer complex queries. We demonstrate the behavior of these metrics in the context of a real and operational semantic warehouse. The results are very promising: the proposed metrics-based matrixes allow someone to get an overview of the contribution (to the warehouse) of each source and to quantify the benefit of the entire warehouse. The later is useful also for monitoring the quality of the warehouse after each reconstruction.

1. INTRODUCTION

An increasing number of datasets are already available as Linked Data. For exploiting this wealth of data, and building domain specific applications, in many cases there is the need for fetching and assembling pieces of information coming from more than one SPARQL endpoints. These pieces are then used for constructing a *warehouse*, for offering more complete browsing and query services (in comparison to those offered by the underlying sources).

We shall use the term *Semantic Warehouse* (for short warehouse) to refer to a read-only set of RDF triples fetched (and transformed) from different sources that aims at serving a particular set of query requirements.

We can distinguish *domain independent* warehouses, like the Sindice RDF search engine [9], or the Semantic Web Search Engine (SWSE) [4], but also *domain specific*, like TaxonConcept[1] and marineTLO-based warehouse [12].

In this paper we focus on the requirements for building domain specific semantic warehouses. Such warehouses aim to serve particular needs, for particular communities of users, consequently their "quality" requirements are more strict. It is therefore worth elaborating on the process that can be used for building such warehouses, and on the related difficulties and challenges. In brief, for building such a warehouse one has to tackle various challenges and questions, e.g. how to define the objectives and the scope of such a warehouse, how to *connect* the fetched pieces of information (common URIs or literals are not always there), how to tackle the various issues of provenance that arise, how to keep the warehouse fresh, i.e. how to automate its construction or refreshing. In this paper we focus on the following questions:

- How to measure the value and quality (since this is important for e-science) of the warehouse?
- How to monitor its quality after each reconstruction or refreshing (as the underlying sources change)?

We have encountered these questions in the context of a real semantic warehouse for the *marine* domain which harmonizes and connects information from different sources of marine information. Past works have focused on the notion of conflicts, and have not paid attention to connectivity. We use the term *connectivity* to express the degree up to which the contents of the semantic warehouse form a connected graph that can serve, ideally in a correct and complete way, the query requirements of the semantic warehouse, while making evident how each source contributes to that degree. To this end in this paper we introduce and evaluate several metrics for quantifying the connectivity of the warehouse. These metrics allow someone to get an overview of the contribution (to the warehouse) of each source (enabling the discrimination of the important from the non important sources) and to quantify the benefit of such a warehouse

The paper is organized as follows: Section 2 describes the main requirements, related works, and what distinguishes the current work. Section 3 provides the context by describing the process used for constructing such warehouses. Section 4 introduces the quality metrics and demonstrates their use. Finally, Section 5 concludes the paper.

2. REQUIREMENTS AND RELATED WORK

The context of this work is the ongoing *iMarine* project[2] that offers an operational distributed infrastructure that serves hundreds of scientists from the marine domain. As regards semantically structured information, the objective is to integrate information from various marine sources, specif-

[1]`http://www.taxonconcept.org/`

[2]FP7, Research Infrastructures, `http://www.i-marine.eu/`

ically from WoRMS[3], Ecoscope[4], FishBase[5], FLOD[6] and DBpedia[7].

The integrated warehouse (its first version is described in [12])[8] is now operational and it is exploited in various applications, including generators of fact sheets (e.g. TunaAtlas[9]), or for enabling exploratory search services (e.g. `X-ENS` [3] that offers semantic post-processing of search results). Below we list the main functional and non functional requirements for constructing such warehouses.

Functional Requirements

- *Multiplicity of Sources.* Ability to query SPARQL endpoints (and other sources), get the results, and ingest them to the warehouse.
- *Mappings, Transformations and Equivalences.* Ability to accommodate schema mappings, perform transformations and create `sameAs` relationships between the fetched content for connecting the corresponding schema elements and entities.
- *Reconstructibility.* Ability to reconstruct the warehouse periodically (from scratch or incrementally) for keeping it fresh.

Non Functional Requirements

- *Scope control.* Make concrete and testable the scope of the information that should be stored in the warehouse. Since we live in the same universe, everything is directly or indirectly connected, therefore without stating concrete objectives there is the risk of continuous expansion without concrete objectives regarding its contents, quality and purpose.
- *Connectivity assessment.* Ability to check and assess the connectivity of the information in the warehouse. Putting triples together does not guarantee that they will be connected. In general, connectivity concerns both schema and instances and it is achieved through common URIs, common literals and `sameAs` relationships. Poor connectivity affects negatively the query capabilities of the warehouse. Moreover, the contribution of each source to the warehouse should be measurable, for deciding which sources to keep or exclude (there are already hundreds of SPARQL endpoints).
- *Provenance.* More than one levels of provenance can be identified and are usually required, e.g. warehouse provenance (from what source that triple was fetched), information provenance (how the fact that the x species is found in y water area was produced), and query provenance (which sources and how contributed to the answer of this query).
- *Consistency and Conflicts.* Ability to specify the desired consistency level of the warehouse e.g. do we want to tolerate an association between a fish commercial code and more than one scientific names? Do we want to consider this as inconsistency (that makes the entire warehouse, or parts of it, unusable), or as resolvable (through a rule) conflict, or as a normal case (and allow it as long as the provenance is available).

[3]`http://www.marinespecies.org/`

[4]`http://www.ecoscopebc.ird.fr/EcoscopeKB/ShowWelcomePage.action`

[5]`http://www.fishbase.org/`

[6]`http://www.fao.org/figis/flod/`

[7]`http://dbpedia.org/`

[8]URL of the warehouse (restricted access): `http://virtuoso.i-marine.d4science.org:8890/sparql`

[9]`http://vmecoscopebc-proto.mpl.ird.fr:8080/semantic-atlas/ShowWelcomePage`

2.1 Related Approaches

Below we refer and discuss in brief the more related systems, namely *ODCleanStore* and *Sieve*.

ODCleanStore [8, 6, 5] is a tool that can download content (RDF graphs) and offers various transformations for cleaning it (deduplication, conflict resolution), and linking it to existing resources, plus assessing the quality of the outcome. It names *conflicts* the cases where two different quads (e.g. sources) have different object values for a certain subject s and predicate p. To such cases conflict resolution rules are offered that either select one or more of these conflicting values (e.g. ANY, MAX, ALL), or compute a new value (e.g. AVG). [5] describes various quality metrics (for scoring each source based on conflicts), as well for assessing the overall outcome.

Another related system is *Sieve* [7] which is part of the Linked Data Integration Framework (LDIF)[10]. This work proposes metrics like *schema completeness* and *conciseness*. However, such metrics are not useful for the case of domain specific warehouses that have a top-level ontology, in the sense that the schema mappings and the transformation rules can tackle these problems. This is true in our warehouse (it is also assumed in the scenarios of *ODCleanStore*).

Overall, we can say that the quality metrics introduced by other works focus more on conflicts. The aspect of connectivity, is not covered sufficiently. The aspect of connectivity is important in warehouses whose schema is not small, and consequently the queries contain paths. The longer such paths are, the more the query capability of the warehouse is determined by the connectivity.

Of course, the issue of data warehouse quality is older than the RDF world, e.g. [10, 1]. A discussion of related works for the RDF world is available in [2], that also focuses on describing data sources in terms of their completeness in query answering. Another quality perspective identified in [13] is that of the *specificity* of the ontology-based descriptions under ontology evolution, an issue that is raised when ontologies and vocabularies evolve over time.

Finally, we could mention that works like [11], which focus on the statistical evaluation of the metadata elements of a repository, are not directly related, since they do not consider the characteristics of RDF and Linked Data, nor they try to evaluate the contribution of the underlying sources.

3. THE INTEGRATION PROCESS

For making clear the context, here we describe in brief the steps of the process that we follow for creating the warehouse. Figure 1 shows an overview of the warehouse's contents, while Figure 2 sketches the construction process[11].

The first step is to *define requirements* in terms of *competency queries*. It is a set of queries (provided by the community) indicating the queries that the warehouse is intended to serve. Some indicative queries are given in Appendix A,

[10]`http://www4.wiwiss.fu-berlin.de/bizer/ldif/`

[11]Extra material is available at `http://www.ics.forth.gr/isl/MarineTLO/#applications`.

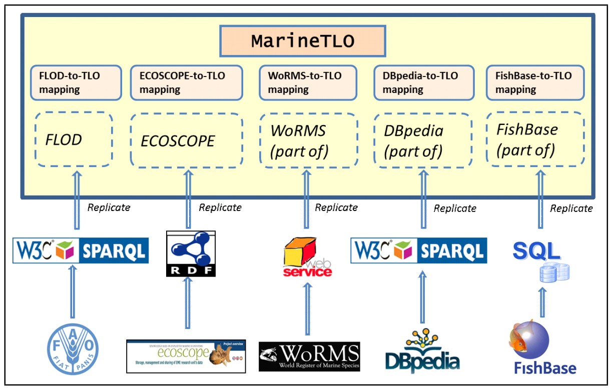
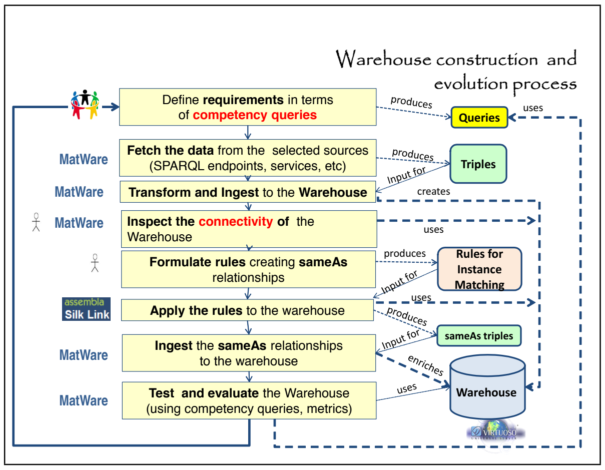


Figure 1: Overview of the warehouse

the full list is web accessible[12]. It is always a good practice to have (select or design) a *top-level schema/ontology* as it alleviates the schema mapping effort (avoids the combinatorial explosion of pair-wise mappings) and allows formulating the competency queries using that ontology (instead of using elements coming from the underlying sources, which change over time). For our case in iMarine, the ontology is called *MarineTLO* [12][13].

The next step is to *fetch the data* from each source and this requires using various access methods (SPARQL endpoints, HTTP accessible files, JDBC) and specifying what exactly to get from each source (all contents or a specific part). For instance, and for the case of the iMarine warehouse, we fetch all triples from FLOD through its SPARQL endpoint, all triples from Ecoscope obtained by fetching OWL files from its web page, information about species (ranks, scientific and common names) from WoRMS, information about species from DBpedia's SPARQL endpoint, and finally information about species, water areas, ecosystems and countries from the relational tables of FishBase.



Warehouse construction and evolution process
Define requirements in terms of competency queries
produces
Queries
uses
MatWare
Fetch the data from the selected sources (SPARQL endpoints, services, etc)
produces
Triples
input for
MatWare
Transform and Ingest to the Warehouse
creates
MatWare
Inspect the connectivity of the Warehouse
uses
Formulate rules creating sameAs relationships
produces
Rules for Instance Matching
input for
uses
assembla Silk Link
Apply the rules to the warehouse
produces
sameAs triples
MatWare
Ingest the sameAs relationships to the warehouse
input for
enriches
Warehouse
MatWare
Test and evaluate the Warehouse (using competency queries, metrics)
uses

Figure 2: The process for constructing and evolving the warehouse

The next step is to *transform and ingest* the fetched data. Some data can be stored as they are fetched, while others have to be transformed, i.e. a *format* transformation and/or a *logical* transformation has to be applied for being compatible with the top-level ontology. For example, a format transformation may be required to transform information expressed in DwC-A (a format for sharing biodiversity data), to RDF. A logical transformation may be required for transforming a string literal to a URI, or for splitting a literal for using its constituents, or for creating intermediate nodes (e.g. instead of `(x,hasName,y)` to have `(x,hasNameAssignement,z),(z,name,y),(z,date,d)`, etc.

This step also includes the definition of the required *schema mappings* that are required for associating the fetched data with the schema of the top level ontology. Another important aspect for domain specific warehouses, is the management of provenance. In our case we support what we call "warehouse"-provenance, i.e. we store the fetched (or fetched and transformed) triples from each source in a separate *graphspace* (a graphspace is a named set of triples which can be used for restricting queries and updates in a RDF triple store). In this way we know which source has provided what facts and this is exploitable also in the queries. As regards conflicts (e.g. different values for the same properties), the adopted policy in our case is to make evident the different values and their provenance, instead of making decisions, enabling in this way the users to select the desired values, and the content providers to spot their differences. The adoption of separate graphspaces also allows refreshing parts of the warehouse, i.e. the part that corresponds to one source. Furthermore, it makes feasible the computation of the metrics that are introduced in the next section.

The next step is to *inspect and test the connectivity* of the "draft" warehouse. This is done through the competency queries as well as through the metrics that we will introduce. The former (competency queries) require manual inspection, but automated tests are also supported. In brief, let q be a query in the set of competency queries. Although we may not know the "ideal" answer of q, we may know that it should certainly contain a particular set of resources, say Pos, and should not contain a particular set of resources, say Neg. Such information allows automated testing. If $ans(q)$ is the answer of q as produced by the warehouse, we would like to hold $Pos \subseteq ans(q)$ and $Neg \cap ans(q) = \emptyset$. Since these conditions may not hold, it is beneficial to adopt an IR-inspired evaluation, i.e. compute the *precision* and *recall* defined as: $precision = 1 - \frac{|Neg \cap ans(q)|}{|ans(q)|}$, $recall = \frac{|Pos \cap ans(q)|}{|Pos|}$. The bigger the values we get the better (ideally 1). The better we know the desired query behaviour, the bigger the sets Pos and Neg are, and consequently the more safe the results of such evaluation are.

Based also on the results of the previous step, the next step is to *formulate rules for instance matching*, i.e. rules that can produce `sameAs` relationships for obtaining the desired connections. For this task we employ the tool SILK[14][14]. Then, we *apply the instance matching rules* (SILK rules in our case) for producing (and then ingesting to the warehouse) `sameAs` relationships.

Finally we have to test the produced repository and evaluate it. This is done through the competency queries and through the metrics that we will introduce.

Periodic Reconstruction Above we have described the

[12]http://www.ics.forth.gr/isl/MarineTLO/competency_queries/MarineTLO_Competency_Queries_Version_v3.pdf
[13]http://www.ics.forth.gr/isl/MarineTLO
[14]http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/

steps required for the first time. After that the warehouse is reconstructed periodically for getting refreshed content. This is done *automatically* through a tool that we have developed called **MatWare**. The metrics that we will introduce are very important for *monitoring* the warehouse after reconstructing it. For example by comparing the metrics in the past and new warehouse, one can understand whether a change in the underlying sources affected negatively the quality (e.g. connectivity) of the warehouse.

4. CONNECTIVITY METRICS

The objective is to define *metrics* for assisting humans on assessing in concrete terms the quality and the value offered by the warehouse.

To aid understanding, after defining each metric we show the values of these metrics as computed over the iMarine warehouse which is built using data from FLOD, WoRMS, Ecoscope, DBpedia, and FishBase. The warehouse is real and operational[15]. This also allows testing whether the metrics are successful.

At first we introduce some required notations. Let $S = S_1, \ldots S_k$ be the set of underlying sources. Each contributes to the warehouse a set of triples (i.e. a set of subject-predicate-object statements), denoted by $triples(S_i)$. This is not the set of all triples of the source. It is the subset that is contributed to the warehouse (fetched mainly by running SPARQL queries). We shall use U_i to denote the URIs that appear in the triples in $triples(S_i)$. Hereafter, we consider only those URIs that appear as *subjects* or *objects* in a triple. We do not include the URIs of the properties because they concern the schema and this integration aspect is already tackled by the top level schema.

Let W denote the triples in the warehouse, i.e. $W = \cup_{1..k} triples(S_i)$.

On Comparing URIs

For computing the metrics that are defined next, we need methods to compare URIs coming from different sources. There are more than one methods, or policies, for doing so. Below we distinguish three main policies:

i *Exact String Equality.* We treat two URIs u_1 and $u2$ as equal, denoted by $u_1 \equiv u_2$, if $u_1 = u_2$ (i.e. strings equality).

ii *Suffix Canonicalization.* Here we consider that $u_1 \equiv u_2$ if $last(u_1) = last(u_2)$ where $last(u)$ is the string obtained by (a) getting the substring after the last "/" or "#", and (b) turning the letters of the picked substring to lowercase and deleting the underscore letters that might exist. According to this policy `http://www.dbpedia.com/Thunnus_Albacares` $\equiv$ `http://www.ecoscope.com/thunnus_albacares` since their canonical suffix is the same, i.e. `thunnusalbacares`. Another example of a equivalent URIs: `http://www.s1.com/entity#thunnus_albacares` $\equiv$ `http://www.s2.org/entity/thunnusAlbacares`.

iii *Entity Matching.* Here consider $u_1 \equiv u_2$ if u_1 `sameAs` u_2 according to the entity matching rules that are (or will be eventually) used for the warehouse. In general such rules create `sameAs` relationships between URIs. In our case we use SILK for formulating and applying such rules.

[15]In the evaluation of related tools, like *Sieve* [7] and *ODCleanStore* [8], real datasets have been used but not "real" operational needs. In our evaluation we use an operational warehouse with concrete (query) requirements which are described by the competency queries.

Note that if two URIs are equivalent according to policy [i], then they are equivalent according to [ii] too. Policy [i] is very strict (probably too strict for matching entities coming from different sources), however it does not produce any false-positive. Policy [ii] achieves treating as equal entities across different namespaces, however false-positives may occur. For instance, `Argentina` is a *country* (`http://www.fishbase.org/entity#Argentina`) but also a *fish genus* (`http://www.marinespecies.org/entity#WoRMS:125885/Argentina`). Policy [iii] is fully aligned with the intended query behaviour of the warehouse (the formulated rules are expected to be better as regards false-negatives and false-positives), however for formulating and applying these entity matching rules, one has to know the contents of the sources. Consequently one cannot apply policy [iii] the first time, instead policies [i] and [ii] can be applied automatically without requiring any human effort. We could also note that policy [ii] can be used for providing hints regarding what entity matching rules to formulate.

Below we define and compute the metrics assuming policy [ii], i.e. whenever we have a set operation we assume equivalence according to [ii] (e.g. $A \cap B$ means $\{ a \in A \mid \exists\, b \in B$ s.t. $a \equiv_{[ii]} b\}$. Then, in Section 4.1, we report results according to policy [iii].

Matrix of Percentages of Common URIs

The number of *common URIs* between two sources S_i and S_j, is given by $|U_i \cap U_j|$. We can define the *percentage of common URIs* (a value ranging [0..1]), as follows: $curi_{i,j} = \frac{|U_i \cap U_j|}{\min(|U_i|,|U_j|)}$. In the denominator we use $\min(|U_i|, |U_j|)$ instead of $|U_i \cup U_j|$ as in the Jaccard similarity. With Jaccard similarity the integration of a small triple set with a big one would always give small values, even if the small set contains many URIs that exist in the big set. For this reason we use *min*.

We now extend the above metric and consider *all* sources aiming at giving an overview of the warehouse. Specifically, we compute a $k \times k$ matrix where $c_{i,j} = curi_{i,j}$. The higher values this matrix contains, the more glued its "components" are. However note that we may have 3 sources, such that each pair of them has a high *curi* value, but the intersection of the URIs of all 3 sources is empty. This is not necessarily bad, for example, consider a source contributing triples of the form `person-lives-placeName`, a second source contributing `placeName-has-postalCode`, and a third one contributing `postCode-isAddressOf-cinema`. Although these three sources may not contain even one common URI, their hosting in a warehouse allows answering queries: *"give me the cinemas in the area where the x person leaves"*.

On the other hand, in a case where the three sources were contributing triples of the form `person-lives-placeName`, `person-worksAt-Organization` and `person-owns-car`, then it would be desired to have common URIs in all sources, as that would allow having more complete information for many persons. Finally, one might wonder why we do not introduce a kind of average path length, or diameter, for the warehouse. Instead of doing that, we inspect the paths that are useful for answering the queries of the users, and this is done through the competency queries.

For the warehouse at hand, Table 1 shows the matrix of the common URIs, while Table 2 shows the matrix of the common URI percentages. The percentages range from 0.3% to 27.39%. We can see that in some cases we have a significant percentage of common URIs between the different sources. The biggest intersection is between FishBase and DBpedia.

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	173,929	239	523	631	887
WoRMS		80,485	200	1,714	3,596
Ecoscope			5,824	192	225
DBpedia				70,246	9,578
FishBase					34,974

Table 1: Common URIs ($|U_i \cap U_j|$)

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	1	0.3%	8.98%	0.9%	2.54%
WoRMS		1	3.43%	2.44%	10.28%
Ecoscope			1	3.3%	3.86%
DBpedia				1	27.39%
FishBase					1

Table 2: Common URIs % ($curi_{i,j} = \frac{|U_i \cap U_j|}{\min(|U_i|,|U_j|)}$)

Percentage of Common literals between two sources

The *percentage of common literals*, between two sources S_i and S_j can be computed by $clit_{i,j} = \frac{|Lit_i \cap Lit_j|}{\min(|Lit_i|,|Lit_j|)}$. To compare 2 literals coming from different sources, we convert them to lower case, to avoid cases like comparing "Thunnus" from one source and "thunnus" from another.

Table 3 shows the matrix of the common literals, while Table 4 shows the percentages. We can see that as regards the literals the percentages of similarity are even smaller than the ones regarding common URIs. The percentages range from 2.71% to 12.37%.

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	111,164	3,624	1,745	5,668	9,505
WoRMS		51,076	382	2,429	4,773
Ecoscope			14,102	389	422
DBpedia				123,887	14,038
FishBase					138,275

Table 3: Common Literals ($|Lit_i \cap Lit_j|$)

Increase in the Average Degree

Now we introduce another metric for expressing the degree of common URIs. Let E be the entities of interest (or all URIs). If T is a set of triples, then we can define the *degree* of an entity e in T as: $deg_T(e) = |\{(s, p, o) \in T \mid s = e \text{ or } o = e\}|$, while for a set of entities E we can define their average degree in T as $deg_T(E) = avg_{e \in E}(deg_T(e))$.

Now for each source S_i we can compute the average degree of the elements in E considering $triples(S_i)$. If the sources of the warehouse contain common elements of E, then if we compute the degrees in the graph of W (i.e. $deg_W(e)$ and $deg_W(E)$), we will get higher values. So the increase in the degree is a way to quantify the gain, in terms of connectivity, that the warehouse offers.

For each source S_i, Table 5 shows the average degree of its URIs (i.e. of those in U_i), and the average degree of the same

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	1	7.1%	12.37%	5.1%	8.55%
WoRMS		1	2.71%	4.76%	9.34%
Ecoscope			1	2.76%	2.99%
DBpedia				1	11.33%
FishBase					1

Table 4: Common Literals % ($clit_{i,j} = \frac{|Lit_i \cap Lit_j|}{\min(|Lit_i|,|Lit_j|)}$)

S_i	avg $deg_{S_i}(U_i)$	avg $deg_W(U_i)$	increase
FLOD	7.18	9.18	27.84%
WoRMS	3.3	7.33	122.36%
Ecoscope	22.84	31.18	36.56%
DBpedia	41.41	42.11	1.7%
FishBase	18.86	29.81	58.08%
AVERAGE	18.72	23.92	27.78%

Table 5: Average degrees in sources and in the warehouse

URIs in the warehouse graph. It also reports the increment percentage (computed by warehouse/source * 100). The last row of the table shows the average values of each column. We observe that the average degree is increased from 18.72 to 23.92

Restricting the Metrics (to the Entities of Interest)

The above metrics can be refined so that to consider not all URIs, but only those that serve the purpose of the warehouse. For example, one could define the above metrics by considering only URIs that are instances of a particular class or classes (e.g. Persons, Locations), or those returned by the competency queries. In general, we can consider that the set of URIs (or entities) of interest is a set E that is defined extensionally (by listing its elements) or intentionally (through a query).

Complementarity of Sources

We now define metrics for quantifying the complementarity of the sources.

The "contribution" of each source S_i can be quantified by counting the triples it has provided to the warehouse, i.e. by $|triples(S_i)|$. We can also define its "unique contribution" by excluding from $triples(S_i)$ those belonging to the triples returned by the other sources. Formally, we can define $triplesUnique(S_i) = triples(S_i) \setminus (\cup_{1 \leq j \leq k, j \neq i} triples(S_j))$. It follows that if a source S_i provides triples which are also provided by other sources, then we have $triplesUnique(S_i) = \emptyset$. Consequently, and for quantifying the contribution of each source to the warehouse, we can compute and report the number of its triples $|triples(S_i)|$, the number of unique triples $|triplesUnique(S_i)|$, and the percentage of unique triples $\frac{|triplesUnique(S_i)|}{|triples(S_i)|}$. To count the unique triples of each source, for each triple of that source we perform suffix canonicalization on its URIs, convert its literals to lower case, and then we check if the resulting (canonical) triple exists in the canonical triples of a different source. If not, we count this triple as unique.

Let $triplesUniques$ be the union of the unique triples of all sources, i.e. $triplesUniques = \cup_i triplesUnique(S_i)$. This set can be proper subset of W (i.e. $triplesUniques \subset W$), since it does not contain triples which have been contributed by two or more sources.

Table 6 shows for each source the number of its triples

$|triples(S_i)|$, the number of unique triples $|triplesUnique(S_i)|$, and the percentage of unique triples $\frac{|triplesUnique(S_i)|}{|triples(S_i)|}$. We can see that every source contains a very high ($> 99\%$) percentage of unique triples, so we can conclude that all sources are important.

S_i	$a = \|triples(S_i)\|$	$b = \|triplesUnique(S_i)\|$	b/a
FLOD	665,456	664,703	99.89%
WoRMS	461,230	460,741	99.89%
Ecoscope	54,027	53,641	99.29%
DBpedia	450,429	449,851	99.87%
FishBase	1,425,283	1,424,713	99.96%

Table 6: (Unique) triple contributions of the sources

We now define another metric for quantifying the value of the warehouse for the entities of interest. Specifically we define the *complementarity factor for an entity* e, denoted by $cf(e)$, as the number of sources that provided unique material about e. It can be defined declaratively as:

$$cf(e) = |\{ \ i \mid triples_W(e) \cap triplesUnique(S_i) \neq \emptyset \}|$$

i.e. it is the number of sources which have provided unique content for e. Note that if $k = 1$, i.e. if we have only one source, then for every entity e we will have $cf(e) = 1$. If $k = 2$, i.e. if we have two sources, then we can have the following cases:
– $cf(e) = 0$ if both sources have provided the same triple (or triples) about e,
– $cf(e) = 1$ if the triples provided by the one source (for e) are subset of the triples provided by the other,
– $cf(e) = 2$ if each source has provided at least one different triple for e (of course they can also have contributed common triples).

Consequently for the entities of interest we can compute and report the average *complementarity factor* as a way to quantify the value of the warehouse for these entities.

Table 7 shows (indicatively) the *complementarity factors* for a few entities which are important for the problem at hand. We see that for the entities "Thunnus" and "Shark" each source provides unique information (with the term entity we mean any literal or URI that contains the word "thunnus" for example). For the entity "Greece" and "Astrapogon" we take unique information from three sources. The fact that the complementarity factor is big means that the warehouse provides information about each entity from all/many sources.

Kind of Entity	$cf(\cdot)/5$
Thunnus	5/5
Greece	3/5
Shark	5/5
Astrapogon	3/5

Table 7: Complementarity factor (cf) of some entities

4.1 After applying the rule-derived 'sameAs' relationships and the transformation rules

So far in the computation of the above metrics we have used policy [ii] (suffix canonicalized URIs) when comparing URIs. Here we show the results from computing again these metrics using policy [iii]. This means that now when comparing URIs we consider the `sameAs` relationships that have been produced by the entity matching rules of the warehouse. In the current warehouse we use 11 SILK rules. An indicative SILK rule is the following: *"If an Ecoscope individual's attribute preflabel (e.g. Thunnus albacares) in lower case is the same with the attribute label in latin of a FLOD individual (e.g. 'thunnus albacares'@la), then these two individuals are the same".*

We should also note that previously, in policy [ii] we considered the triples as they are fetched form the sources. Here we consider the triples as derived from the transformation rules (described in §3).

Computing the metrics using policy [iii], not only allows evaluating the gain achieved by these relationships, but it also better reflects the value of the warehouse since query answering considers the `sameAs` relationships.

Table 8 shows the matrix of the common URIs after the rule-derived `sameAs` relationships and the execution of the transformation rules, and Table 9 shows the corresponding percentages. We can see that, compared to the results of Tables 1 and 2, after considering the `sameAs` relationships the number of common URIs between the different sources is significantly increased (more than 7 times in some cases).

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	190,733	434	1,897	4,009	6,732
WoRMS		80,486	805	1,754	3,596
Ecoscope			7,805	1,245	2,116
DBpedia				74,381	10,385
FishBase					34,974

Table 8: Common URIs ($|U_i \cap U_j|$)

S_i \ S_j	FLOD	WoRMS	Ecoscope	DBpedia	FishBase
FLOD	1	0.54%	24.3%	5.39%	19.25%
WoRMS		1	10.31%	2.36%	10.28%
Ecoscope			1	15.95%	27.1%
DBpedia				1	29.69%
FishBase					1

Table 9: Common URIs % ($curi_{i,j} = \frac{|U_i \cap U_j|}{\min(|U_i|,|U_j|)}$)

Table 10 shows the average degree of the URIs of each source S_i (i.e. of those in U_i), and the average degree of the same URIs in the warehouse graph. It also reports the increment percentage (computed by warehouse/source * 100). The last row of the table shows the average values of each column. We can see that the average degree, of all sources, after the inclusion of the `sameAs` relationships is significantly bigger than before. In comparison to Table 5, the increase is from 2 to almost 8 times bigger. This means that we achieve a great increase in terms of the connectivity of the information in the warehouse.

As regards the unique contribution of each source, Table 11 shows the number of the triples of each source $|triples(S_i)|$, the number of unique triples $|triplesUnique(S_i)|$, and the percentage of unique triples $\frac{|triplesUnique(S_i)|}{|triples(S_i)|}$. We observe that the values in the column *"a"* are increased in comparison to Table 6. This is because of the execution of the transformation rules after the ingestion of the data to the warehouse, which results to the creation of new triples for the majority of sources. Finally we observe that, in general,

S_i	avg $deg_{S_i}(U_i)$	avg $deg_W(U_i)$	increase
FLOD	7.18	54.31	656.51%
WoRMS	3.3	9.93	201.36%
Ecoscope	22.84	165.24	623.6%
DBpedia	41.41	84.2	103.36%
FishBase	18.86	50.6	168.32%
AVERAGE	18.72	72.86	289.21%

Table 10: Average degrees in sources and in the warehouse

the percentage of unique triples provided by each source is decreased. This happens because the transformation rules and the same-as relationships have turned previously different triples, the same.

S_i	$a = \|triples(S_i)\|$	$b = \|triplesUnique(S_i)\|$	b/a
FLOD	810,301	798,048	98.49%
WoRMS	582,009	527,358	99.88%
Ecoscope	138,324	52,936	38.27%
DBpedia	526,016	517,242	98.33%
FishBase	1,425,283	1,340,968	94.08%

Table 11: (Unique) triple contributions of the sources

4.2 Detecting Redundancies or other Pathological Cases

The metrics can be used also for detecting various pathological cases, e.g. sources that do not have any common URI or literal, or "redundant sources". To test this we created two artificial sources, let's call them *Airports* and *CloneSource*. The first contains triples about airports which were fetched from the DBpedia public SPARQL endpoint, while the second is a subset of Ecoscope's and DBpedia's triples as they are stored in the warehouse.

In the sequel, we computed the metrics for all 7 sources. Table 12 shows the unique triples and Table 13 shows the average degrees. As regards *Airports*, the percentage of common URIs was very low, and the average degree for the entities of that source was very low too (2.22% due to some common country names), while its unique contribution was 100%. As regards *CloneSource* we got 0 unique contribution (as expected, since it was composed from triples of existing sources).

S_i	$a =$ $\|triples(S_i)\|$	$b =$ $\|triplesUnique(S_i)\|$	b/a
FLOD	665,456	664,703	99.89%
WoRMS	461,230	460,741	99.89%
Ecoscope	54,027	17,951	33.23%
DBpedia	450,429	429,426	95.34%
Fishbase	1,425,283	1,424,713	99.96%
CloneSource	56,195	0	**0%**
Airports	31,628	31,628	**100%**

Table 12: (Unique) triple contributions of the sources

Rules for Detecting Pathological Cases

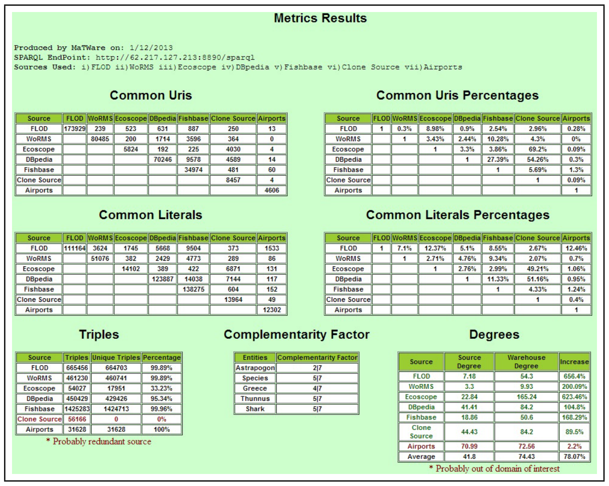
It follows that we can detect pathological cases using two rules: (a) if the average increase of the degree of the entities of a source is low, then this means that its contents are not connected with the contents of the rest sources (this is the case of *Aiports* where we had only 2.22% increase), (b) if the unique contribution of a source is very low (resp. zero), then this means that it does not contribute significantly (resp. at all) to the warehouse (this is the case of *CloneSource* where the unique contribution was zero).

S_i	avg $deg_{S_i}(U_i)$	avg $deg_W(U_i)$	increase
FLOD	7.18	54.31	656.51%
WoRMS	3.3	9.93	201.36%
Ecoscope	22.84	165.24	623.6%
DBpedia	41.41	84.2	103.36%
FishBase	18.86	50.6	168.32%
CloneSource	44.43	84.2	89.52%
Airports	70.99	72.56	**2.22%**
AVERAGE	29.86	74.43	149.26%

Table 13: Average degrees in sources and in the warehouse

4.3 Implementation

As regards implementation, the above metrics are computed by the tool **MatWare** that we have developed. The values of the metrics are exposed in the form of an HTML page (as shown in Figure 3) providing in this way a kind of quantitative documentation of the warehouse. As regards time, the current warehouse (containing 3,772,919 triples) takes about 7 hours to reconstruct.[16]



Metrics Results

```
Produced by MaTWare on: 1/12/2013
SPARQL EndPoint: http://62.217.127.213:8890/sparql
Sources Used: i)FLOD ii)WoRMS iii)Ecoscope iv)DBpedia v)Fishbase vi)Clone Source vii)Airports
```

Common Uris

Source	FLOD	WoRMS	Ecoscope	DBpedia	Fishbase	Clone Source	Airports
FLOD	173929	239	523	631	887	250	13
WoRMS		80485	200	1714	3596	364	0
Ecoscope			5824	192	225	4030	4
DBpedia				70246	9578	4589	14
Fishbase					34974	481	60
Clone Source						8457	4
Airports							4606

Common Uris Percentages

Source	FLOD	WoRMS	Ecoscope	DBpedia	Fishbase	Clone Source	Airports
FLOD	1	0.3%	8.98%	0.9%	2.54%	2.96%	0.28%
WoRMS		1	3.43%	2.44%	10.28%	4.3%	0%
Ecoscope			1	3.3%	3.86%	69.2%	0.09%
DBpedia				1	27.39%	54.26%	0.3%
Fishbase					1	5.69%	1.3%
Clone Source						1	0.09%
Airports							1

Common Literals

Source	FLOD	WoRMS	Ecoscope	DBpedia	Fishbase	Clone Source	Airports
FLOD	111164	3624	1745	5668	9504	373	1533
WoRMS		51076	382	2429	4773	289	86
Ecoscope			14102	389	422	6871	131
DBpedia				123887	14038	7144	117
Fishbase					138275	604	152
Clone Source						13964	49
Airports							12302

Common Literals Percentages

Source	FLOD	WoRMS	Ecoscope	DBpedia	Fishbase	Clone Source	Airports
FLOD	1	7.1%	12.37%	5.1%	8.55%	2.67%	12.46%
WoRMS		1	2.71%	4.76%	9.34%	2.07%	0.7%
Ecoscope			1	2.76%	2.99%	49.21%	1.06%
DBpedia				1	11.33%	51.16%	0.95%
Fishbase					1	4.33%	1.24%
Clone Source						1	0.4%
Airports							1

Triples

Source	Triples	Unique Triples	Percentage
FLOD	665456	664703	99.89%
WoRMS	461230	460741	99.89%
Ecoscope	54027	17951	33.23%
DBpedia	450429	429426	95.34%
Fishbase	1425283	1424713	99.96%
Clone Source	56166	0	0%
Airports	31628	31628	100%

* Probably redundant source

Complementarity Factor

Entities	Complementarity Factor
Astrapogon	2|7
Species	5|7
Greece	4|7
Thunnus	5|7
Shark	5|7

Degrees

Source	Source Degree	Warehouse Degree	Increase
FLOD	7.18	54.3	656.4%
WoRMS	3.3	9.93	200.09%
Ecoscope	22.84	165.24	623.46%
DBpedia	41.41	84.2	104.8%
Fishbase	18.86	50.6	168.29%
Clone Source	44.43	84.2	89.5%
Airports	70.99	72.56	2.2%
Average	41.8	74.43	78.07%

* Probably out of domain of interest

Figure 3: Metrics results displayed in HTML as produced by MatWare

5. SYNOPSIS AND CONCLUSION

For many applications one has to fetch and assemble pieces of information coming from more than one SPARQL endpoints. In this paper we have described the main requirements and challenges, based also on our experience so far in building a semantic warehouse for marine resources. We have presented a process for constructing such warehouses and then we introduced metrics for quantifying the connectivity of the outcome.

The results are very positive. By inspecting the proposed metrics-based matrixes one can very quickly get an overview

[16]Virtuoso and machine spec: Openlink Virtuoso V6.1, Ubuntu 12.10 64bit, Quad-Core, 4 GB RAM.

of the contribution of each source and the tangible benefits of the warehouse. The main metrics proposed are: (a) the matrix of percentages of the common URIs and/or literals, (b) the complementarity factor of the entities of interest, (c) the table with the increments in the average degree of each source, and (d) the unique triple contribution of each source. The values of (a),(b),(c) allow valuating the warehouse, while (c) and (d) mainly concern each particular source.

For instance, and for the warehouse at hand, by combining the unique triples contribution (from Table 11) and the increment of the average degrees (of Table 10), we can understand that not only we get unique information *from all* sources, but also *how much* the average degree of the entities of the sources has been increased in the warehouse. Moreover, redundant sources can be spotted through their low unique contribution, while unconnected sources through their low average increase of the degree of their entities. Of course one could combine the above metrics and derive various other single-valued metrics for expressing the quality (connectivity, redundancy) of each source, as well as for the entire warehouse.

The ability to assess the quality of a semantic warehouse (using methods like those presented in this paper, as well those presented in §2.1) is very important for judging whether the warehouse can be used in e-Science. It is also important because in the long run we expect that datasets and warehouses will be peer-reviewed, evaluated and cited, and this in turn will justify actions for their future preservation.

In future we plan to continue along this direction, focusing also on methods that compare the metrics of two different warehouse versions for monitoring the evolution of the warehouse over time.

Acknowledgement

This work was partially supported by the ongoing project *iMarine* (FP7 Research Infrastructures, 2011-2014).

6. REFERENCES

[1] D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. *Communications of the ACM*, 42(1):73–78, 1999.

[2] F. Darari, W. Fariz, W. Nutt, G. Pirro, and S.Razniewski. Completeness Statements about RDF Data Sources and their Use for Query Answering. In *The Semantic Web–ISWC 2013*, pages 66–83. Springer, 2013.

[3] P. Fafalios and Y. Tzitzikas. X-ENS: Semantic Enrichment of Web Search Results at Real-Time. In *SIGIR'13*, pages 1089–1090, Dublin, Ireland, 2013. ACM.

[4] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and Browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4), 2011.

[5] T. Knap and J. Michelfeit. Linked Data Aggregation Algorithm: Increasing Completeness and Consistency of Data, `http://www.ksi.mff.cuni.cz/~knap/files/aggregation.pdf`.

[6] T. Knap, J. Michelfeit, J. Daniel, P. Jerman, D. Rychnovskỳ, T. Soukup, and M. Nečaskỳ. ODCleanStore: a Framework for Managing and Providing Integrated Linked Data on the Web. In *Web Information Systems Engineering-WISE 2012*, pages 815–816. Springer, 2012.

[7] P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123. ACM, 2012.

[8] J. Michelfeit and T. Knap. Linked Data Fusion in ODCleanStore. In *International Semantic Web Conference (Posters & Demos)*, 2012.

[9] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a Document-Oriented Lookup Index for Open Linked Data. *Int. J. Metadata Semant. Ontologies*, 3(1):37–52, 2008.

[10] G. G. Shanks and P. Darke. Understanding Data Quality and Data Warehousing: A Semiotic Approach. In *Third Conference on Information Quality (IQ'98)*, pages 292–309, 1998.

[11] E. Tsiflidou and N. Manouselis. Tools and Techniques for Assessing Metadata Quality. In *7th Metadata and Semantics Research Conference (MTSR'13)*, 2013.

[12] Y. Tzitzikas, C. Alloca, C. Bekiari, Y. Marketakis, P. Fafalios, M. Doerr, N. Minadakis, T. Patkos, and L. Candela. Integrating Heterogeneous and Distributed Information about Marine Species through a Top Level Ontology. In *Proceedings of the 7th Metadata and Semantic Research Conference (MTSR'13)*, Thessaloniki, Greece, November 2013.

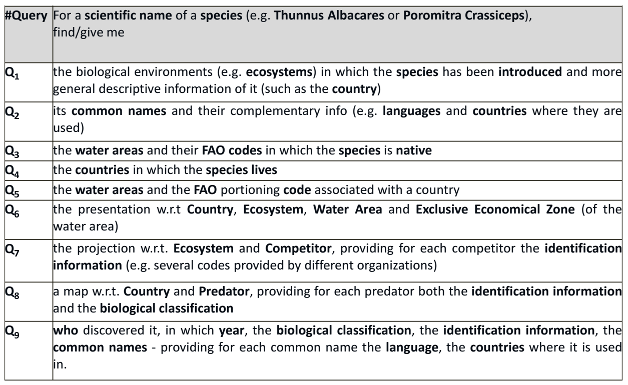
[13] Y. Tzitzikas, M. Kampouraki, and A. Analyti. Curating the Specificity of Ontological Descriptions under Ontology Evolution. *Journal on Data Semantics*, pages 1–32, 2013.

[14] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In *Proceedings of the WWW'09 Workshop on Linked Data on the Web*, 2009.

APPENDIX

A. COMPETENCY QUERIES

Figure 4 displays the textual description for some competency queries as they were supplied by the communities.



#Query	For a **scientific name** of a **species** (e.g. **Thunnus Albacares** or **Poromitra Crassiceps**), find/give me
Q_1	the biological environments (e.g. **ecosystems**) in which the **species** has been **introduced** and more general descriptive information of it (such as the **country**)
Q_2	its **common names** and their complementary info (e.g. **languages** and **countries** where they are used)
Q_3	the **water areas** and their **FAO codes** in which the **species** is **native**
Q_4	the **countries** in which the **species lives**
Q_5	the **water areas** and the **FAO** portioning **code** associated with a country
Q_6	the presentation w.r.t **Country**, **Ecosystem**, **Water Area** and **Exclusive Economical Zone** (of the water area)
Q_7	the projection w.r.t. **Ecosystem** and **Competitor**, providing for each competitor the **identification information** (e.g. several codes provided by different organizations)
Q_8	a map w.r.t. **Country** and **Predator**, providing for each predator both the **identification information** and the **biological classification**
Q_9	**who** discovered it, in which **year**, the **biological classification**, the **identification information**, the **common names** - providing for each common name the **language**, the **countries** where it is used in.

Figure 4: Some indicative competency queries