Quantifying the Connectivity of a Semantic Warehouse and Understanding its Evolution over Time

Michalis Mountantonakis\textsuperscript{1,2}, Nikos Minadakis\textsuperscript{3}, Yannis Marketakis\textsuperscript{1}, Pavlos Fafalios\textsuperscript{1,2}, Yannis Tzitzikas\textsuperscript{1,2}

\textsuperscript{1} Institute of Computer Science, FORTH-ICS, GREECE, and
\textsuperscript{2} Computer Science Department, University of Crete, GREECE
\texttt{\{mountant, minadakn, marketak, fafalios, tzitzik\}@ics.forth.gr}

Abstract

In many applications one has to fetch and assemble pieces of information coming from more than one source for building a semantic warehouse offering more advanced query capabilities. In this paper we describe the corresponding requirements and challenges, and we focus on the aspects of quality and value of the warehouse. For this reason we introduce various metrics (or measures) 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, as well as on synthetically produced warehouses. The proposed metrics allow someone to get an overview of the contribution (to the warehouse) of each source and to quantify the value of the entire warehouse. Consequently, these metrics can be used for advancing data/endpoint profiling and for this reason we use an extension of VoID (for making them publishable). Such descriptions can be exploited for dataset/endpoint selection in the context of federated search. In addition, we show how the metrics can be used for monitoring a semantic warehouse after each reconstruction reducing thereby the cost of quality checking, as well as for understanding its evolution over time.

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 sources. These pieces are then used for constructing a warehouse, offering thereby more complete and efficient browsing and query services (in comparison to those offered by the underlying sources). We 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. In general, we can distinguish domain independent warehouses, like the Sindice RDF search engine [39] or the Semantic Web Search Engine (SWSE) [28], but also domain specific, like TaxonConcept [5] and the MarineTLO-based warehouse [44]. Domain specific semantic 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)?
- How to understand the evolution of the warehouse?
- How to measure the contribution of each source to the warehouse, and hence deciding which sources to keep or exclude?

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. Most past works have focused on the notion of conflicts (e.g., [36]), 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. Moreover connectivity is a notion which can be exploited in the task of dataset or endpoint selection.

To this end, in this paper we introduce and evaluate upon real and synthetic datasets several metrics for quantifying the connectivity of a warehouse. What we call metrics could be also called measures, i.e. they should not be confused with distance functions. 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. This paper extends the metrics first proposed in [46] whose publishing according to an extension of VoID is described in [38]. With respect to that work, in this paper:

- We extensively discuss related work and the placement of the current work.
- We generalize the “source-to-source” metrics, and we introduce metrics that involve more than two sources, as well as a lattice-based visualization of these metrics.
- We introduce various single-valued metrics for quantifying the contribution of a source or the value of the warehouse. To make easier and faster the identification and inspection of pathological cases (redundant sources or sources that do not contribute new information) we propose a single-valued metric that characterizes the overall contribution of a source. This value considers the contribution (in terms of triples and connectivity) of a source and its size, and it indicates the degree up to which the source contributes to the warehouse.
- We propose methods that exploit these metrics for understanding and monitoring the evolution of the warehouse.
- We propose an alternative way of modeling the values of the proposed metrics using 3D models. This allows quickly understanding the situation of a warehouse and how the underlying sources contribute and are connected. The 3D visualization can also be used for modeling the evolution of a warehouse over time (using animations).

The rest of this paper is organized as follows: Section 2 describes the main requirements, and provides the context by briefly describing the process used for constructing such warehouses. Section 3 describes related work and what distinguishes the current

---

one. Section 4 introduces the quality metrics and demonstrates their use. Section 5 discusses how these metrics can be used for monitoring and understanding the evolution of the warehouse over time. Section 6 describes how we calculate and exploit the connectivity metrics. Finally, Section 7 concludes the paper and identifies directions for future research.

2 Context, Requirements and Integration Process

2.1 Context and Requirements

The spark for this work was the recently completed iMarine project (and the ongoing BlueBRIDGE project) that offers an operational distributed infrastructure that serves hundreds of scientists from the marine domain. As regards semantically structured information, the objective was to integrate information from various marine sources, specifically from:

- **WoRMS** [6]: it is a marine registry containing taxonomic information and lists of common names and synonyms for more than 200 thousand species in various languages.
- **Ecoscope** [1]: it is knowledge base containing geographical data, pictures and information about marine ecosystems.
- **FishBase** [2]: is a global database of fish species, containing information about the taxonomy, geographical distribution, biometrics, population, genetic data and many more.
- **FLOD** [3]: is a network of marine linked data containing identification information using different code lists.
- **DBpedia** [10]: is a knowledge base containing content that has been converted from Wikipedia, that by the time of writing this paper, the English version contained more than 4.5 million resources.

The integrated warehouse\(^2\) is operational and it is exploited in various applications, including the gCube infrastructure [11], or for enabling exploratory search services (e.g., X-ENS [19] that offers semantic post-processing of search results). For the needs of the iMarine and BlueBRIDGE projects, the materialized (warehouse) integration approach was more suited because it offers (a) flexibility in transformation logic (including ability to curate and fix problems), (b) decoupling of the release management of the integrated resource from the management cycles of the underlying sources, (c) decoupling of access load from the underlying sources, and (d) faster responses (in query answering but also in other tasks, e.g., in entity matching). 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**

\(^2\)The warehouse can be accessed from https://i-marine.d4science.org/.
• **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 level 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).

2.2 **Context: 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 contents, while Figure 2 sketches the construction process. For the construction (and the update) of the warehouse, we have used the tool *MatWare* [47], that automates the entire process.

![Figure 1: Overview of the warehouse.](image)

(1) **Define the requirements.** 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.

It is always a good practice to have (select or design) a top-level schema/ontology as it alleviates the schema mapping effort (avoiding the combinatorial explosion of pairwise mappings) and allows formulation of the competency queries using that ontology (instead of using elements coming from the underlying sources, which change over time). For our case in iMarine, we used the MarineTLO [44] ontology.

(2) Fetch. The next step is to fetch the data from each source and this requires using various access methods (including 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/RDF files from its Web page, information about species (ranks, scientific and common names) from WoRMS by accessing a specific-purpose service, called Species Data Discovery Service (SDDS) (provided by gCube infrastructure [11]), information about species from DBpedia’s SPARQL endpoint, and finally information about species, water areas, ecosystems and countries from the relational tables of FishBase.

(3) Transform and (4) Ingest. 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 [50] (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, hasNameAssignment, 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 [47]. 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

Figure 2: The process for constructing and evolving the warehouse.
in our case is to make evident the different values and their provenance, instead of making decisions, enabling thereby 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.

(5) Inspect/Test the Connectivity. 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 in a real dataset or warehouse (due to errors, omissions, etc.), it is beneficial to adopt an IR-inspired evaluation, i.e., compute the precision and recall defined as:

$$\text{precision} = \frac{|Pos \cap ans(q)|}{|ans(q)|}, \quad \text{recall} = \frac{|Pos \cap ans(q)|}{|Pos|}.$$  

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.

(6) Formulate rules for Instance Matching. 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 $\text{sameAs}$ relationships for obtaining the desired connections. For this task we employ the tool SILK$^3$ [48].

(7) Apply the rules and (8) Ingest the derived $\text{sameAs}$ relationships. Then, we apply the instance matching rules (SILK rules in our case) for producing, and then ingesting to the warehouse, $\text{sameAs}$ relationships.

(9) Create and Ingest VoID triples. For publishing and exchanging the characteristics of the warehouse (e.g., number of triples, endpoint, publisher, etc.), as well as the several metrics that we will introduce, we create and ingest to the warehouse triples based on an extension of VoID [30] that is described in [38] (VoID is an RDF-based schema that allows metadata about RDF datasets to be expressed).

(10) Test/Evaluate the Warehouse. Finally, we have to test the produced repository and evaluate it after ingesting the produced $\text{sameAs}$ relationships. This is done through the competency queries and through the metrics that we will introduce.

Above we have described the steps required for the first time. After that, the warehouse is reconstructed periodically for getting refreshed content using 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 positively or negatively the quality (connectivity) of the warehouse.

3 Related Work

Data quality is commonly conceived as fitness of use for a certain application or use case [33,49]. The issue of data quality, especially for the case of a data warehouse, is older than the RDF world, e.g., the database community has studied it in the relational world [8,42]. Connectivity, as defined in the Introduction, can be considered as a dimension of data quality in the context of a Semantic Warehouse. Recall that a Semantic Warehouse refers to a read-only set of RDF triples fetched and transformed from different sources that aims

3http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/
at serving a particular set of query requirements. Thereby, in this context connectivity is an important aspect of data quality aiming to measure the degree up to which the contents of the warehouse are connected (and thus satisfy its query requirements).

Fürber and Hepp [20] investigated data quality problems for RDF data originating from relational databases, while a systematic review of approaches for assessing the data quality of Linked Data is presented by Zaveri et al. [53]. In that work, the authors surveyed 21 approaches and extracted 26 data quality dimensions (such as completeness, provenance, interlinking, reputation, accessibility, and others) along with the corresponding metrics.

Below, we first (in §3.1) discuss some quality aspects that are especially useful for the case of a Semantic Warehouse, we report approaches that have tried to address them and we place our work in the literature. In §3.2 we compare (based on different perspectives) several frameworks and systems that automate quality assessment for the RDF world.

3.1 Quality Aspects

Completeness. Completeness refers to the degree up to which all required information is presented in a particular dataset [53]. In the RDF world, completeness can be classified (according to Zaveri et al. [53]) as: schema completeness (degree to which the classes and properties of an ontology are represented), property completeness (measure of the missing values for a specific property), population completeness (percentage of all real-world objects of a particular type that are represented in the datasets), and interlinking completeness (degree to which instances in the dataset are interlinked).

The problem of assessing completeness of Linked Data sources was discussed by Harth and Speiser [24]. Darari et al. [14] introduce a formal framework for the declarative specification of completeness statements about RDF data sources and underline how the framework can complement existing initiatives like VoID. They also show how to assess completeness of query answering over plain and RDF/S data sources augmented with completeness statements, and they present an extension of the completeness framework for federated data sources.

Provenance. Provenance focuses on how to represent, manage and use information about the origin of the source to enable trust, assess authenticity and allow reproducibility [53]. Hartig [25] presents a provenance model for Web data which handles both data creation and data access. The author also describes options to obtain provenance information and analyzes vocabularies to express such information. Hartig and Zhao [26] propose an approach of using provenance information about the data on the Web to assess their quality and trustworthiness. Specifically, the authors use the provenance model described in [25] and propose an assessment method that can be adapted for specific quality criteria (such as accuracy and timeliness). This work also deals with missing provenance information by associating certainty values with calculated quality values. In [27], the same authors introduce a vocabulary to describe provenance of Web data as metadata and discuss possibilities to make such provenance metadata accessible as part of the Web of Data. Furthermore, they describe how this metadata can be queried and consumed to identify outdated information. Given the need to address provenance, the W3C community has standardised the PROV Model\(^4\), a core provenance data model for building representations of the entities, people and processes involved in producing a piece of data or thing in the world. The PROV Family of Documents\(^5\) [37] defines the model, corresponding serializations and other supporting definitions to enable the interoperable interchange of provenance information in heterogeneous environments such as the Web.

Amount-of-data. Amount-of-data is defined as the extent to which the volume of data is

\(^4\)http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/
\(^5\)http://www.w3.org/TR/prov-overview/
is appropriate for the task at hand [9,53]. This dimension can be measured in terms of general dataset statistics like number of triples, instances per class, internal and external links, but also coverage (scope and level of detail) and metadata “richness”. Tsiflidou and Manouselis [43] carried out an analysis of tools that can be used for the valid assessment of metadata records in a repository. More specifically, three different tools are studied and used for the assessment of metadata quality in terms of statistical analysis. However, such works do not consider the characteristics of RDF and Linked Data. Auer et al. [7] describe LODStats, a statement-stream-based approach for gathering comprehensive statistics (like classes/properties usage, distinct entities and literals, class hierarchy depth, etc.) about RDF datasets. To represent the statistics, they use VoID and the RDF Data Cube Vocabulary. The RDF Data Cube Vocabulary⁶ [12] provides a means to publish multi-dimensional data (such as statistics of a repository) on the Web in such a way that it can be linked to related datasets and concepts. Hogan et al. [29] performed analysis in order to quantify the conformance of Linked Data with respect to Linked Data guidelines (e.g., use external URIs, keep URIs stable). They found that in most datasets, publishers followed some specific guidelines, such as using HTTP URIs, whereas in other cases, such as providing human readable metadata, the result were disappointing since only a few publishers created metadata for their datasets.

**Accuracy.** Accuracy is defined as the extent to which data is correct, that is, the degree up to which it correctly represents the real world facts and is also free of errors [53]. Accuracy can be measured by detecting outliers, conflicts, semantically incorrect values or poor attributes that do not contain useful values for the data entries. Fürber and Hepp [21] categorize accuracy into semantic and syntactic accuracy. Semantic accuracy checks whether the data value represents the correct state of an object, whereas syntactic accuracy checks if a specific value violates syntactical rules. For measuring accuracy, the authors used three rules and four formulas, whereas the results were evaluated by using precision and recall measures. They managed to detect syntactic and semantic errors such as invalid country combinations, rules for phone numbers and so forth. OD-CleanStore [32,36] names conflicts the cases where two different quads (e.g., triples from different sources) have different object values for a certain subject and predicate. 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). Finally, Knap and Michelfeit [31] describe various quality metrics for scoring each source based on conflicts, as well for assessing the overall outcome.

**Relevancy.** Relevancy refers to the provision of information which is accordant with the task at hand and suitable to the users’ query [53]. The existence of irrelevant data can have negative consequences for the query performance, while it will be difficult for the user to explore this data, since the user expects to receive the correct information. Zaveri et al. [52] divide relevancy (for DBpedia) into the following sub-categories: (i) extraction of attributes containing layout information, (ii) image related information, (iii) redundant attribute values, and finally (iv) irrelevant information. The existence of a number of different properties for a specific subject-object pair is an example of redundant information.

**Dynamics / Evolution.** Dynamics quantifies the evolution of a dataset over a specific period of time and takes into consideration the changes occurring in this period. Dividino et. al [18] lists probably all works related to the dynamics of LOD datasets. A related quality perspective, identified by Tzitzikas et al. [45], 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.

**Interlinking.** Interlinking refers to the degree to which entities that represent the same concept are linked to each other [53]. This can be evaluated by measuring the

---
⁶http://www.w3.org/TR/2013/PR-vocab-data-cube-20131217/
existence of sameAs links and chains, the interlinking degree, etc. Zaveri et al. [52] classify interlinking into two different categories: (i) external websites (checking whether there are links among sources which are not available), and (ii) interlinks with other datasets (trying to detect incorrect mappings and links which do not provide useful information).

Our Placement: 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. The proposed connectivity metrics reflect the query capabilities of a warehouse as a whole (so they are important for evaluating its value), but also quantify the contribution of the underlying sources allowing evaluating the importance of each source for the warehouse at hand. 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 capabilities of the warehouse are determined by the connectivity.

In the related literature, the aspect of connectivity is not covered sufficiently and regards mainly the existence of sameAs links and chains (e.g., [52]). If we would like to associate connectivity with the existing quality dimensions, we could say that it is predominantly interlinking and secondly relevancy and amount-of-data. Of course, it can be exploited together with approaches that focus on completeness (e.g., [14]), provenance (e.g., [26]), accuracy, (e.g., [31]), etc. Regarding relevancy, we should stress that, by construction, a Semantic Warehouse as created by the proposed process (see Figure 2) does not contain irrelevant data since the data has been fetched based on the requirements defined in terms of competency queries. Furthermore, the proposed metrics can even detect redundant sources and sources containing data which are not connected with data found in the other sources. Compared to existing approaches on amount-of-data (like LODStats [7]), the proposed connectivity metrics can be used to gather statistics that regard more than one source (like common URIs, common literals, etc.) Finally, as regards dynamics/evolution, existing works (e.g., [18]) concern atomic datasets, not warehouses comprising parts of many datasets.

3.2 Frameworks/Systems for Quality Assessment

Here, we discuss frameworks/systems that automate quality assessment. At first we give a brief description of each framework/system and what quality aspects it can handle, and then we compare them regarding several aspects.

ODCleanStore [31,32,36] 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 in terms of accuracy, consistency, conciseness and completeness.

Sieve [35] is part of the Linked Data Integration Framework (LDIF) [4] and proposes metrics for assessing the dimensions in terms of schema completeness, conciseness and consistency. The role of this tool is to assess the quality by deciding which values to keep, discard or transform according to a number of metrics and functions which are configurable via a declarative specification language.

RDFUnit [34] measures the accuracy and the consistency of a dataset containing Linked Data. More specifically, it checks the correct usage of vocabularies according to a number of constraints (e.g., cardinality restriction on a property). One can use some custom SPARQL queries to quantify the quality of a specific dataset for the aforementioned aspects.
LinkQA [22] uses a number of metrics to assess the quality of Linked Data mappings regarding the dimensions of interlinking and completeness. This tool can be used for detected pathological cases, such as bad quality links, before they are published.

Luzzu [16] is a framework for assessing the quality of Linked data for 10 different dimensions, such as availability, provenance, consistency and so forth. In particular, by using this tool one can perform quality evaluation either by using some of the 25 available metrics or by defining his own metrics.

SWIQA [21] is a framework for the validation of the values of semantic resources based on a set of rules. This framework allows the calculation of quality scores for various dimensions, such as completeness, timeliness and accuracy, in order to identify possible problems with the data values. Moreover, it is also applicable on top of relational databases with the support of wrapping technologies (i.e., D2RQ).

Finally, MatWare [47] is a tool that automates the process of constructing semantic warehouses by fetching and transforming RDF triples from different sources. MatWare computes and visualizes the connectivity metrics that are proposed in this paper.

Figure 3 illustrates the dimensions that the aforementioned approaches and MatWare measure.

Table 1 categorizes the aforementioned frameworks according to various aspects like data format, number of sources, output kind, output format, computability and extensibility. Firstly, LinkQA [22] takes as an input a set of Linked data Mappings and produces an HTML page containing the results of the measures which were computed through a Java Application. On the contrary, the input of Luzzu [16] is a specific dataset and the output is produced in RDF format by using the Dataset Quality Ontology [15] and the Quality Problem Report Ontology (QPRO)7. In particular, they use qpro:QualityReport and qpro:QualityProblem in order to represent problems that emerged during the assessment of quality on a specific dataset. It is a common point with our approach, since MatWare can produce the results in RDF format by using an extension of VoID [38]. Concerning ODCleanStore [31,32,36], a common point with MatWare is that it computes the metrics for aggregated RDF data (not only for a specific source) and it produces the results in HTML and RDF. Regarding Sieve [35], it relies on a set of configurable metrics in order to assess the quality of an integrated dataset while the results are produced in the form of Quads. On the contrary, SWIQA [21] and RDFUnit [34] offer generalized SPARQL query templates for assessing the quality of the data, therefore, these metrics are domain-independent. Finally, compared to other tools, MatWare [47] offers a variety of output formats, including 3D visualization, whereas it can also be used for any domain.

7http://butterbur04.iai.uni-bonn.de/ontologies/qpro/qpro
Moreover, we could mention systems for keyword searching over RDF datasets. Such systems can be very helpful for creating domain independent warehouses and offering user-friendly search and browsing capabilities. However, they are not closely related with semantic warehouses, since they aim at collecting everything, and do not have specific requirements as regards quality of data and query answering. Additionally, they do not measure quality aspects. For example, the Semantic Web Search Engine (SWSE) [28] adapts the architecture of the common Web search engines for the case of structured RDF data (it offers crawling, data enhancing, indexing and a user interface for search, browsing and retrieval of RDF data). Swoogle [17] is a crawler-based indexing and retrieval system for the Semantic Web that uses multiple crawlers to discover Semantic Web Documents (SWDs) through meta-search and link-following, analyzes SWDs and produces metadata, computes ranks of SWDs using a rational random surfing model, and indexes SWDs using an information retrieval system. Sindice [39] is an indexing infrastructure with a Web front-end and a public API to locate Semantic Web data sources such as RDF files and SPARQL endpoints. It offers an automatic way to locate documents including information about a given resource, which can be either a URI or full-text search. Finally, Watson [13] is a Semantic Web search engine providing various functionalities not only to find and locate ontologies and semantic data online, but also to explore the content of these semantic documents.

### Table 1: Categorizing Existing Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Input</th>
<th>Num of Sources</th>
<th>Output Kind</th>
<th>Output Format</th>
<th>Computability</th>
<th>Extendable</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkQA</td>
<td>RDF/XML</td>
<td>One Source</td>
<td>Numeric values</td>
<td>HTML</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Luzzu [16]</td>
<td>RDF/XML</td>
<td>Set of Sources</td>
<td>Numeric values</td>
<td>RDF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ODCleanStore</td>
<td>RDF/XML</td>
<td>One Source</td>
<td>Collection of Quads</td>
<td>HTML</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sieve [35]</td>
<td>RDF/XML</td>
<td>One (integrated) Source</td>
<td>Numeric values</td>
<td>RDF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SWRQA [21]</td>
<td>RDF/XML</td>
<td>One Source</td>
<td>Numeric values</td>
<td>RDF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RDFUnit [34]</td>
<td>RDF/XML</td>
<td>One Source</td>
<td>numeric values</td>
<td>RDF, HTML</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MatWare [47]</td>
<td>RDF/XML</td>
<td>Set of Sources</td>
<td>Numeric values</td>
<td>RDF, HTML, 3D</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Moreover, we could mention systems for keyword searching over RDF datasets. Such systems can be very helpful for creating domain independent warehouses and offering user-friendly search and browsing capabilities. However, they are not closely related with semantic warehouses, since they aim at collecting everything, and do not have strict requirements as regards quality of data and query answering. Additionally, they do not measure quality aspects. For example, the Semantic Web Search Engine (SWSE) [28] adapts the architecture of the common Web search engines for the case of structured RDF data (it offers crawling, data enhancing, indexing and a user interface for search, browsing and retrieval of RDF data). Swoogle [17] is a crawler-based indexing and retrieval system for the Semantic Web that uses multiple crawlers to discover Semantic Web Documents (SWDs) through meta-search and link-following, analyzes SWDs and produces metadata, computes ranks of SWDs using a rational random surfing model, and indexes SWDs using an information retrieval system. Sindice [39] is an indexing infrastructure with a Web front-end and a public API to locate Semantic Web data sources such as RDF files and SPARQL endpoints. It offers an automatic way to locate documents including information about a given resource, which can be either a URI or full-text search. Finally, Watson [13] is a Semantic Web search engine providing various functionalities not only to find and locate ontologies and semantic data online, but also to explore the content of these semantic documents.

### 4 Connectivity Metrics

**Def. 1** We use the term *connectivity metric* (or *connectivity measure*) to refer to a measurable quantity that expresses 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 constituent source contributes to that degree.

What we call *metrics* could be also called *measures*. They include measures of similarity between two sources in the form of percentages (e.g. regarding common URIs), natural numbers (e.g. cardinalities of intersections), matrices of measures, means of other measures, as well as relative measures (e.g. increase of average degrees, unique contribution and others). They should not be confused with distance functions.

Such measures can assist humans on assessing in concrete terms the quality and the value offered by the warehouse. In addition they provide a summary of the contents of the warehouse which can be exploited by external applications in the context of distributed query answering.

To aid understanding in the sections that follow, after defining each metric we show the values of these metrics as computed over the MarineTLO-based warehouse which is built using data from five marine-related sources: FLOD, WoRMS, Ecoscope, DBpedia, and FishBase (cf. Figure 1). Since the warehouse is real and operational, this way of...
presentation also allows the reader to see how the metrics behave in a real setting.

This section is organized as follows: §4.1 introduces notations and discusses ways for comparing URIs, §4.2 introduces metrics for comparing pairs of sources, §4.4 introduces metrics for quantifying the value of the entire warehouse, §4.5 introduces metrics for quantifying the value of one source (in the context of one warehouse). An overview of the categories of the various metrics and measurements is given in Figure 4.

<table>
<thead>
<tr>
<th>Source-to-Source connectivity (§4.2)</th>
<th>Connectivity of Set of Sources (§4.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-to-Source connectivity (§4.2)</td>
<td>Connectivity of Set of Sources (§4.3)</td>
</tr>
<tr>
<td>Similarities between pairs of sources</td>
<td>Similarities among sources</td>
</tr>
</tbody>
</table>

Figure 4: An overview of the categories of the proposed metrics.

### 4.1 Notations and Ways to Compare URIs

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 \( \text{triples}(S_i) \). 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 \( \text{triples}(S_i) \). 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 triples of all sources of the warehouse. In general, the set of all triples of the warehouse, say \( W_{\text{All}} \), is superset of \( W \) (i.e., \( W_{\text{All}} \supset W = \bigcup_{i=1}^{k} \text{triples}(S_i) \)) because the warehouse apart from the triples from the sources, contains also the triples representing the top-level ontology, the schema mappings, the sameAs relationships, etc.

#### 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 method, or policy, for doing so. Below we distinguish three main policies:

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

- **Suffix Canonicalization.** Here we consider that \( u_1 \equiv u_2 \) if \( \text{last}(u_1) = \text{last}(u_2) \) where \( \text{last}(u) \) 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 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.
the underscore letters as well as space and special characters that might exist. According to this policy:

\[ \text{http://www.dbpedia.com/Thunnus_Albacares} \equiv \text{http://www.ecoscope.com/thunnus_albacares} \]

since their canonical suffix is the same, i.e., `thunnus_albacares`. Another example of equivalent URIs:

\[ \text{http://www.s1.com/entity#thunnus_albacares} \equiv \text{http://www.s2.org/entity/thunnusAlbacares} \]

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.

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://dbpedia.org/resource/Argentina) but also a fish genus (http://dbpedia.org/resource/Argentina_(fish)). Policy [iii] is fully aligned with the intended query behavior 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 while it requires human effort. Consequently one cannot apply policy [iii] the first time, instead policies [i] and [ii] can be applied automatically (without any human effort). Policy [ii] should be actually 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 \{ \( a \in A \mid \exists b \in B \text{ s.t. } a \equiv_{[ii]} b \} \)). Then (after applying the entity matching rules), we compute the metrics according to Policy [iii], which is actually the policy that characterizes the query behavior of the final and operational warehouse.

4.2 Metrics for Comparing two Sources

4.2.1 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:

\[
curi_{i,j} = \frac{|U_i \cap U_j|}{\min(|U_i|, |U_j|)} \tag{1}
\]

In the denominator we use \( \min(|U_i|, |U_j|) \) although one could use \( |U_i \cup U_j| \) that is used 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, while the Jaccard similarity reveals the overall contribution of a source. We now extend the above metric and consider all pairs of 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.

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

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
Table 2: Matrix of common URIs (with their percentages) using Policy [ii].

<table>
<thead>
<tr>
<th></th>
<th>FLOD</th>
<th>WoRMS</th>
<th>Ecoscope</th>
<th>DBpedia</th>
<th>FishBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_i</td>
<td>S_j</td>
<td>FLOD</td>
<td>WoRMS</td>
<td>Ecoscope</td>
<td>DBpedia</td>
</tr>
<tr>
<td>FLOD</td>
<td>173,929 (100%)</td>
<td>239 (0.3%)</td>
<td>523 (8.98%)</td>
<td>631 (9.9%)</td>
<td>887 (2.54%)</td>
</tr>
<tr>
<td>WoRMS</td>
<td>80,485 (100%)</td>
<td>200 (3.43%)</td>
<td>1,714 (10.28%)</td>
<td>3,596 (10.28%)</td>
<td></td>
</tr>
<tr>
<td>Ecoscope</td>
<td>5,824 (100%)</td>
<td>192 (3.3%)</td>
<td>70,246 (100%)</td>
<td>9,578 (27.39%)</td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td>9,578 (100%)</td>
<td>225 (4.86%)</td>
<td></td>
<td></td>
<td>34,974 (100%)</td>
</tr>
</tbody>
</table>

person-lives-placeName, a second source contributing placeName-has-postalCode, and a third one contributing postCode-addressOf-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 lives”. On the other hand, in a case where the three sources contribute 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, since 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.

Measurements after Adding the Rule-derived ‘sameAs’ Relationships and Applying the Transformation Rules

So far in the computation of the above metrics we have used policy [ii] (suffix canonical-ized URIs) when comparing URIs. Here we show the results from computing again these metrics using policy [iii] and after adding the triples as derived from the transformation rules (described in §2.2). Moreover, extra URIs have been produced due to transformation rules (e.g., in order to assign a URI to a species name). As a result, 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 have used 11 SILK rules. An indicative SILK rule is the following: “If the value of the attribute “preflabel” of an Ecoscope individual (e.g., Thunnus albacares) in lower case is the same with the attribute “label” in latin of a FLOD individual (e.g., ‘thunus albacares’@la), then these two individuals are the same (create a sameAs link between them)”. We should also note that in policy [ii], we have considered the triples as they are fetched from the sources. Computing the metrics using policy [iii], not only allows evaluating the gain achieved by these relationships, but it also reflects better the value of the warehouse since query answering considers the sameAs relationships.

Table 3 shows the matrix of the common URIs after the rule-derived relationships and the execution of the transformation rules (together with the corresponding percentages). We can see that, compared to the results of Table 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).

Furthermore, Table 4 shows the Jaccard similarity between the pairs of sources. If we compare the results between these two tables, we can see that the percentages when using Jaccard similarity table have been reduced remarkably.

4.2.2 Matrix of Percentages of Common Literals between two Sources

The percentage of common literals, between two sources $S_i$ and $S_j$ can be computed by:

$$colit_{i,j} = \frac{|Lit_i \cap Lit_j|}{\min(|Lit_i|, |Lit_j|)}$$  \hspace{1cm} (2)
Table 3: Matrix of common URIs (and their percentages) using Policy [iii].

<table>
<thead>
<tr>
<th></th>
<th>FLOD</th>
<th>WoRMS</th>
<th>Ecoscope</th>
<th>DBpedia</th>
<th>FishBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>190,749 (100%)</td>
<td>1,738 (2.64%)</td>
<td>869 (11.2%)</td>
<td>4,127 (5.46%)</td>
<td>6,053 (17.31%)</td>
</tr>
<tr>
<td>WoRMS</td>
<td>65,789 (100%)</td>
<td>809 (10.43%)</td>
<td>1,807 (2.75%)</td>
<td>4,373 (12.5%)</td>
<td></td>
</tr>
<tr>
<td>Ecoscope</td>
<td>7,759 (100%)</td>
<td>1,117 (14.4%)</td>
<td>2,171 (27.98%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td>75,518 (100%)</td>
<td>10,388 (20.7%)</td>
<td>34,973 (100%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FishBase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Matrix of percentages of common URIs using Policy [iii] and Jaccard Similarity.

<table>
<thead>
<tr>
<th></th>
<th>FLOD</th>
<th>WoRMS</th>
<th>Ecoscope</th>
<th>DBpedia</th>
<th>FishBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>1</td>
<td>0.68%</td>
<td>0.44%</td>
<td>1.56%</td>
<td>2.69%</td>
</tr>
<tr>
<td>WoRMS</td>
<td>1</td>
<td>1.11%</td>
<td>1.29%</td>
<td>4.5%</td>
<td></td>
</tr>
<tr>
<td>Ecoscope</td>
<td>1</td>
<td>1.36%</td>
<td>5.35%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td>1</td>
<td>10.31%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FishBase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Matrix of common Literals (and their percentages).

<table>
<thead>
<tr>
<th></th>
<th>FLOD</th>
<th>WoRMS</th>
<th>Ecoscope</th>
<th>DBpedia</th>
<th>FishBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>111,164 (100%)</td>
<td>3,624 (7.1%)</td>
<td>1,745 (12.37%)</td>
<td>5,668 (5.1%)</td>
<td>9,505 (8.55%)</td>
</tr>
<tr>
<td>WoRMS</td>
<td>51,076 (100%)</td>
<td>382 (2.71%)</td>
<td>2,429 (4.76%)</td>
<td>4,773 (9.34%)</td>
<td></td>
</tr>
<tr>
<td>Ecoscope</td>
<td>14,102 (100%)</td>
<td>369 (2.76%)</td>
<td>222 (2.99%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FishBase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Matrix of the Harmonic Mean of Common URIs and Literals

\[ eUriscLiti,j = \frac{2 \times curi_{i,j} \times colit_{i,j}}{curi_{i,j} + colit_{i,j}} \] (3)

Table 6 presents the results of this metric.

4.3 Metrics for Comparing a Set of Sources (Lattice-based)

The measurements described in §4.2 are measurements between pairs of sources. However, we can generalize the present metrics between any subset of the sources of the warehouse, e.g., the number of common literals in 4 sources.

The idea is to provide measurements for each subset of the set of sources, i.e., for every element of \( \mathcal{P}(S) \) where \( \mathcal{P}(S) \) denotes the powerset of \( S \). For visualizing (and understanding) these measurements, we propose a partial set-like visualization. Specifically, we propose constructing and showing the measurements in a way that resembles the Hasse Diagram of the poset (partially ordered set) \((S, \subseteq)\).

Let \( R \) be any nonempty subset of \( S \) (i.e., \( R \subseteq S \)). It is not hard to generalize the aforementioned metrics for every such subset. For example, consider the metric
common triples. The nodes at the lower level of the Hasse Diagram (corresponding to the singletons of $S$), actually show the number of triples of each source in $S$, while the nodes in the level above correspond to pairs of sources, i.e., they correspond to what the matrices that we have introduced show. At the topmost level, we have the intersection between all sources in $S$ (i.e., the value $|U_1 \cap \ldots \cap U_k|$).

Figure 5 presents the lattice concerning the common URIs according to policy [ii]. One can see that the number of common URIs of DBpedia, FishBase, and Ecoscope are more than the number of common URIs among the subsets of the same level, while 74 common URIs are included in all sources.

This approach can be used for all metrics, e.g., for common URIs, for common literals, as well for the metrics that we will introduce later for the entire warehouse (i.e., for the average degree, for the complementarity factor, etc). The diagram contains $2^{|S|}$ nodes.

### 4.4 Metrics for Evaluating the Entire Warehouse

Here we introduce metrics that measure the quality of the entire warehouse.

#### 4.4.1 Increase in the Average Degree

Now we introduce another metric for expressing the degree of a set of nodes, where a node can be either a URI or a blank node$^9$. Let $E$ be the entities of interest (or the union of all URIs and blank nodes).

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 $\overline{\deg_T(E)} = \frac{1}{|E|} \sum_{e \in E} \deg_T(e)$. Now for each source $S_i$, we can compute the average degree of the elements in $E$ considering $\text{triples}(S_i)$. 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 $\overline{\deg_W(E)}$), 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. Furthermore, we can define a normalized metric for average degree increment i.e., a metric whose value approaches 1 in the best case, and 0 in the worst. To this end, we define

$$\text{DegIncr}(S_i, W) = \frac{\deg_W(U_i) - \overline{\deg_S(U_i)}}{\overline{\deg_W(U_i)}}$$

For each source $S_i$, Table 7 shows the average degree of its URIs and blank nodes, and the average degree of the same URIs and blank nodes in the warehouse graph. It also reports the increment percentage, and the normalized metric for the average degree increment. The last row of the table shows the average values of each column. We observe that the average degree is increased from 9.39 to 15.29.

$^9$In our case the size of blank nodes in the warehouse is much bigger than the size of unique triples (about twice).

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>FLOD</th>
<th>WoRMS</th>
<th>Ecoscope</th>
<th>DBpedia</th>
<th>FishBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>1</td>
<td>1.61%</td>
<td>10.44%</td>
<td>1.53%</td>
<td>3.93%</td>
</tr>
<tr>
<td>WoRMS</td>
<td>1</td>
<td>4.45%</td>
<td>11.02%</td>
<td>21.86%</td>
<td></td>
</tr>
<tr>
<td>Ecoscope</td>
<td>1</td>
<td>3.08%</td>
<td>3.82%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td>1</td>
<td></td>
<td></td>
<td>15.85%</td>
<td></td>
</tr>
<tr>
<td>FishBase</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Harmonic Mean of Common URIs and Literals.
Measurements after Adding the Rule-derived ‘sameAs’ Relationships and Applying the Transformation Rules

Table 8 shows the average degree of the URIs and blank nodes of each source $S_i$, and the average degree of the same URIs and blank nodes in the warehouse graph, when policy [iii] is used. It also reports the increment percentage, and the normalized metric for the average degree increment. 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 7, the increase is from 2 to 9 times bigger. This means that we achieve a great increase in terms of the connectivity of the information in the warehouse.

4.4.2 Unique Triples Contribution

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 $|\text{triples}(S_i)|$. We can also define its “unique contribution” by excluding from $\text{triples}(S_i)$ those belonging to the triples returned by the other sources.
Table 7: Average degrees in sources and in the warehouse using policy [ii].

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$\text{avg deg}_S(U_i)$</th>
<th>$\text{avg deg}_W(U_i)$</th>
<th>Increase %</th>
<th>$\text{DegIncr}(S_i, W)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>5.82</td>
<td>6.85</td>
<td>16.84%</td>
<td>0.14</td>
</tr>
<tr>
<td>WoRMS</td>
<td>4.14</td>
<td>4.24</td>
<td>2.46%</td>
<td>0.02</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>21.37</td>
<td>47.56</td>
<td>122.52%</td>
<td>0.55</td>
</tr>
<tr>
<td>DBpedia</td>
<td>6.9</td>
<td>6.99</td>
<td>1.36%</td>
<td>0.01</td>
</tr>
<tr>
<td>FishBase</td>
<td>8.75</td>
<td>10.89</td>
<td>24.46%</td>
<td>0.19</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>9.39</td>
<td>15.29</td>
<td>62.89%</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Table 8: Average degrees in sources and in the warehouse using Policy [iii].

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$\text{avg deg}_S(U_i)$</th>
<th>$\text{avg deg}_W(U_i)$</th>
<th>Increase %</th>
<th>$\text{DegIncr}(S_i, W)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>5.82</td>
<td>52.01</td>
<td>739.64%</td>
<td>0.89</td>
</tr>
<tr>
<td>WoRMS</td>
<td>4.14</td>
<td>8.19</td>
<td>97.94%</td>
<td>0.49</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>21.37</td>
<td>90.52</td>
<td>323.51%</td>
<td>0.76</td>
</tr>
<tr>
<td>DBpedia</td>
<td>6.9</td>
<td>42.97</td>
<td>523.23%</td>
<td>0.84</td>
</tr>
<tr>
<td>FishBase</td>
<td>8.71</td>
<td>18.99</td>
<td>117.19%</td>
<td>0.54</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>9.39</td>
<td>42.53</td>
<td>353%</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Formally, for the $k$ sources of the warehouse, we can define:

$$\text{triplesUnique}(S_i) = \text{triples}(S_i) \setminus (\bigcup_{1 \leq j \leq k, j \neq i} \text{triples}(S_j))$$

(5)

It follows that if a source $S_i$ provides triples which are also provided by other sources, then we have $\text{triplesUnique}(S_i) = \emptyset$. Consequently, and for quantifying the contribution of each source to the warehouse, we can compute and report the number of its triples $|\text{triples}(S_i)|$, the number of the unique triples $|\text{triplesUnique}(S_i)|$, the unique contribution of each source as:

$$\text{UniqueT Contrib}(S_i) = \frac{|\text{triplesUnique}(S_i)|}{|\text{triples}(S_i)|}$$

(6)

Obviously, it becomes 0 in the worst case and 1 in the best case. 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 $\text{triplesUniques}$ be the union of the unique triples of all sources, i.e., $\text{triplesUniques} = \bigcup_i \text{triplesUnique}(S_i)$. This set can be proper subset of $W$ (i.e., $\text{triplesUniques} \subset W$), since it does not contain triples which have been contributed by two or more sources.

Table 9 shows for each source the number of its triples $|\text{triples}(S_i)|$, the number of unique triples $|\text{triplesUnique}(S_i)|$, and the unique triples contribution of that source $(\text{UniqueT Contrib}(S_i))$. We can see that every source contains a very high (> 99%) percentage of unique triples, so we can conclude that all sources are important.

| $S_i$ | $|\text{triples}(S_i)|$ | $|\text{triplesUnique}(S_i)|$ | $\text{UniqueT Contrib}(S_i)$ |
|-------|------------------------|-------------------------------|-----------------------------|
| FLOD  | 665,456                | 664,703                       | 99.99%                      |
| WoRMS | 461,230                | 460,741                       | 99.99%                      |
| Ecoscope | 54,027               | 53,641                       | 99.29%                      |
| DBpedia | 450,429              | 449,851                       | 99.97%                      |
| FishBase | 1,425,283            | 1,424,713                     | 99.96%                      |

Table 9: (Unique) triple contributions of the sources using policy [ii].

Measurements after Adding the Rule-derived ‘sameAs’ Relationships and Applying the Transformation Rules

As regards the unique contribution of each source using Policy [iii], Table 10 shows the number of the triples of each source $|\text{triples}(S_i)|$, the number of unique triples...
triplesUnique(S_i)), and the unique triples contribution \(UniqueT\text{ Contrib}(S_i)\). We observe that the values in the first column are increased in comparison to Table 9. 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, the unique triples contribution of each source is decreased. This happens because the transformation rules and the same-as relationships have turned previously different triples, the same.

| S_i  | |triples(S_i)| |triplesUnique(S_i)| |UniqueT\text{ Contrib}(S_i) |
|------|-----------------|-----------------|-----------------|
| FLOD | 810,301         | 798,048         | 98.49%          |
| WoRMS| 528,009         | 527,358         | 99.88%          |
| Ecoscope | 138,324 | 52,936 | 38.27% |
| DBpedia | 826,046 | 517,242 | 98.33% |
| FishBase | 1,425,283 | 1,340,968 | 94.08% |

Table 10: (Unique) triple contributions of the sources using Policy [iii].

### 4.4.3 Complementarity of Sources

We now define another metric for quantifying the value of the warehouse for the entities of interest. With the term “entity” we mean any literal or URI that contains a specific string representing a named entity, like the name of a fish or country. The set of triples containing information about the entity of interest can be defined as \(triples_W(e) = \{<s, p, o>| s = e \text{ or } o = e\}\). Specifically we define the complementarity factor for an entity \(e\), denoted by \(cf(e)\), as the percentage of sources that provide unique material about \(e\). It can be defined declaratively as:

\[
cf(e) = \frac{|\{i | triples_W(e) \cap triplesUnique(S_i) \neq \emptyset\}|}{|S|}
\]

where \(S\) is the set of underlying sources.

Note that if \(|S| = 1\) (i.e., we have only one source), then for every entity \(e\) we will have \(cf(e) = 1.0\). If \(|S| = 2\), i.e., if we have two sources, then we can have the following cases:

- \(cf(e) = 0.0\), if both sources have provided the same triple (or triples) about \(e\) or no source has provided any triple about \(e\),
- \(cf(e) = 0.5\), if the triples provided by the one source (for \(e\)) are subset of the triples provided by the other, or if only one source provide triple(s) about \(e\),
- \(cf(e) = 1.0\), 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 11 shows (indicatively) the complementarity factors for a few entities which are important for the problem at hand. We can see that for the entities “Thunnus” and “Shark” each source provides unique information \((cf = 1.0)\). For the entity “Greece” and “Astrapogon” we obtain unique information from three sources \((cf = 3/5 = 0.6)\). The fact that the complementarity factor is big means that the warehouse provides unique information about each entity from many/all sources. Moreover, Table 12 shows the average complementarity factor of the species that are native to Greece. One can observe that there are no species with very small complementarity factor, which means that at least 2 sources provide unique information for each species \(cf(e) \geq 0.4\). Indeed, exactly 2 sources provide unique information for 116 species, while for 35 species we get unique data from all the sources. In general the average complementarity factor for all species that are native in Greece is approximately 0.63 (3.15/5) (meaning that at least 3 sources contain unique information for such species).
Table 11: Complementarity factor (cf) of some entities.

<table>
<thead>
<tr>
<th>Kind of Entity</th>
<th>cf()</th>
<th>No. of Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thunnus</td>
<td>1.0 (5/5)</td>
<td>0</td>
</tr>
<tr>
<td>Greece</td>
<td>0.6 (3/5)</td>
<td>116</td>
</tr>
<tr>
<td>Shark</td>
<td>1.0 (5/5)</td>
<td>180</td>
</tr>
<tr>
<td>Astrapogon</td>
<td>0.6 (4/5)</td>
<td>113</td>
</tr>
</tbody>
</table>

Table 12: cf of species that are native to Greece.

<table>
<thead>
<tr>
<th>cf()</th>
<th>No. of Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2 (1/5)</td>
<td>0</td>
</tr>
<tr>
<td>0.4 (2/5)</td>
<td>116</td>
</tr>
<tr>
<td>0.6 (3/5)</td>
<td>180</td>
</tr>
<tr>
<td>0.8 (4/5)</td>
<td>113</td>
</tr>
<tr>
<td>1.0 (5/5)</td>
<td>35</td>
</tr>
</tbody>
</table>

Average: 0.63 (3.15/5) Sum: 444

4.5 Metrics for Evaluating a Single Source (in the context of a warehouse)

In this section we focus on metrics for quantifying the value that a source brings to the warehouse. Such metrics should also allow identifying pathological cases (e.g., redundant or irrelevant sources). In particular, §4.5.1 provides examples of such cases and introduces rules for identifying them, while §4.5.2 introduces a single-valued metric based on these rules for aiding their identification by a human.

4.5.1 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 three artificial sources, let us call them Airports, CloneSource and AstrapogonSource. The Airports source contains triples about airports which were fetched from the DBpedia public SPARQL endpoint, the CloneSource is a subset of Ecoscope’s and DBpedia’s triples as they are stored in the warehouse, and the AstrapogonSource contains only 1 URI and 4 triples for the entity Astrapogon. In the sequel, we computed the metrics for 8 sources.

Table 13 shows the unique triples and Table 14 shows the average degrees. The metrics were calculated according to policy [iii]. As regards Airports, we can see that its unique contribution is 100% (all the contents of that source are unique). As regards CloneSource we got 0 unique contributions (as expected, since it was composed from triples of existing sources). Finally, concerning the AstrapogonSource, we noticed that although the number of triples contribution is very low, all its triples are unique.

Table 13: (Unique) triple contributions of the sources.

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>(triplen($S_i$))</th>
<th>(triplenUnique($S_i$))</th>
<th>UniqueT Contrib($S_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>810,301</td>
<td>798,048</td>
<td>99.49%</td>
</tr>
<tr>
<td>WoRMS</td>
<td>528,009</td>
<td>527,358</td>
<td>99.88%</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>1,58,324</td>
<td>17,901</td>
<td>12.9%</td>
</tr>
<tr>
<td>DBpedia</td>
<td>526,016</td>
<td>505,013</td>
<td>96%</td>
</tr>
<tr>
<td>FishBase</td>
<td>1,425,283</td>
<td>1,340,968</td>
<td>94.08%</td>
</tr>
<tr>
<td>AstrapogonSource</td>
<td>4</td>
<td>4</td>
<td>100.00%</td>
</tr>
<tr>
<td>CloneSource</td>
<td>56,195</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Airports</td>
<td>31,628</td>
<td>31,628</td>
<td>100%</td>
</tr>
</tbody>
</table>

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 of the sources (this is the case of Airports where we had only 0.1% 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).
To further ease the inspection of pathological cases (and the quantification of the contribution of each source), we can define a single (and single-valued) measure. One method is to use the **harmonic mean** of the unique contribution, and the increment in the average degree (the harmonic mean takes a high value if both values are high). Therefore, we can measure the harmonic mean of the above two metrics and define the value of a source $S_i$, denoted by $value_0(S_i, W)$, as:

$$
value_0(S_i, W) = \frac{2 \times UniqueTContrib(S_i) \times \text{DegIncr}(S_i, W)}{UniqueTContrib(S_i) + \text{DegIncr}(S_i, W)}
$$

(8)

Table 15 shows these values for all sources of the warehouse, including the artificial ones, in decreasing order. We can see that the problematic sources have a value less than 0.04 while the good ones receive a value greater than 0.2. However, $AstrapogonSource$ has the highest score although it contains only 4 triples. The two reasons why this source seems the best according to this metric are that all the triples are unique and the only instance that it contains has a lot of properties in other sources. Therefore, the degree increment of this source is almost 1. Consequently this metric makes evident the contribution of each source to the warehouse.

**Table 15: The value of a source in the Warehouse (using $value_0(S_i, W)$).**

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$UniqueTContrib(S_i)$</th>
<th>$\text{DegIncr}(S_i, W)$</th>
<th>$value_0(S_i, W)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>0.9849</td>
<td>0.89</td>
<td>0.935</td>
</tr>
<tr>
<td>WoRMS</td>
<td>0.9988</td>
<td>0.49</td>
<td>0.6575</td>
</tr>
<tr>
<td>FishBase</td>
<td>0.129</td>
<td>0.76</td>
<td>0.2206</td>
</tr>
<tr>
<td>Airports</td>
<td>1</td>
<td>0.001</td>
<td>0.02</td>
</tr>
<tr>
<td>CloneSource</td>
<td>0</td>
<td>0.89</td>
<td>0</td>
</tr>
</tbody>
</table>

For tackling this issue, we need an analogous metric for the size of a specific source in the warehouse, specifically we can define $S_i\text{SizeInW}(S_i, W) = \frac{\text{triples}(S_i)}{\text{triples}(W)}$. We can now compute the harmonic mean of these three metrics and define the value of a source $S_i$, denoted by $value_1(S_i, W)$, as:

$$
value_1(S_i, W) = \frac{3}{\frac{1}{UniqueTContrib(S_i)} + \frac{1}{\text{DegIncr}(S_i, W)} + \frac{1}{S_i\text{SizeInW}(S_i, W)}}
$$

(9)

Table 16 shows these values for all sources of the warehouse, including the artificial ones, in decreasing order. Now we can see that FishBase is the most useful source, and the score of AstrapogonSource is very low (almost 0).
Consequently, the first metric can be used for deciding whether to include or not a source in the warehouse, while second for inspecting the importance of source for the warehouse. In case of adding a huge out-of-domain source in our warehouse while there exist a lot of useful sources which are much smaller, the values of the useful sources will remain almost stable for the first metric. On the contrary, their values will be decreased for the second metric. Regarding, the value of the out-of-domain source, it will be low in both metrics, since the increase of the average degree for this source will be almost 0. Therefore, both metrics will show that the new source should be removed from the warehouse, however, the second metric will not show the real value for each of the remaining sources in this case.

<table>
<thead>
<tr>
<th>S_i</th>
<th>UniqueTContrib(S_i)</th>
<th>DegIncr(S_i, W)</th>
<th>S_iSizeInW(S_i, W)</th>
<th>value1(S_i, W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FishBase</td>
<td>0.9408</td>
<td>0.54</td>
<td>0.405</td>
<td>0.5572</td>
</tr>
<tr>
<td>FLOD</td>
<td>0.9849</td>
<td>0.89</td>
<td>0.2304</td>
<td>0.463</td>
</tr>
<tr>
<td>Dipedia</td>
<td>0.96</td>
<td>0.84</td>
<td>0.1496</td>
<td>0.3384</td>
</tr>
<tr>
<td>WoRMS</td>
<td>0.9988</td>
<td>0.49</td>
<td>0.1501</td>
<td>0.3091</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>0.129</td>
<td>0.76</td>
<td>0.0393</td>
<td>0.0869</td>
</tr>
<tr>
<td>Airports</td>
<td>1</td>
<td>0.001</td>
<td>0.0089</td>
<td>0.0027</td>
</tr>
<tr>
<td>AstrapogonSource</td>
<td>1</td>
<td>0.93</td>
<td>0.000001</td>
<td>0.000001</td>
</tr>
<tr>
<td>CloneSource</td>
<td>0</td>
<td>0.89</td>
<td>0.0089</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 16: The value of a source in the warehouse (using value1(S_i, W)).

4.6 Summary of the Metrics

Table 17 shows all the definitions of the metrics that were described in this section.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Metric Definition</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common URIs between two sources S_i and S_j</td>
<td>(</td>
<td>U_i \cap U_j</td>
</tr>
<tr>
<td>Percentage of common URIs between S_i and S_j</td>
<td>( curi,j = \frac{</td>
<td>U_i \cap U_j</td>
</tr>
<tr>
<td>Common literals between S_i and S_j</td>
<td>(</td>
<td>Lit_i \cap Lit_j</td>
</tr>
<tr>
<td>Percentage of common literals between S_i and S_j</td>
<td>( coliti,j = \frac{</td>
<td>Lit_i \cap Lit_j</td>
</tr>
<tr>
<td>Harmonic mean of common URIs and literals</td>
<td>( cUrisLit_{i,j} = \frac{2 \times curi,j \times coliti,j}{curi,j + coliti,j} )</td>
<td>mean</td>
</tr>
<tr>
<td>Increase in the average degree</td>
<td>( DegIncr(S_i, W) = \frac{degW(E) - degS(E)}{degS(E)} )</td>
<td>relative measure</td>
</tr>
<tr>
<td>Unique triples of S_i</td>
<td>( \text{triplesUnique}(S_i) = \text{triples}(S_i) \setminus \bigcup_{1 \leq j \leq k, j \neq i} \text{triples}(S_j) )</td>
<td>relative measure</td>
</tr>
<tr>
<td>Unique triples contribution of a source</td>
<td>( \text{UniqueTContrib}(S_i) = \frac{</td>
<td>\text{triplesUnique}(S_i)</td>
</tr>
<tr>
<td>Value of a source</td>
<td>( value1(S_i, W) = \frac{UT + DI + SW}{3} )</td>
<td>mean</td>
</tr>
<tr>
<td>where</td>
<td>UT = UniqueTContrib(S_i), DI = DegIncr(S_i, W), SW = S_iSizeInW(S_i, W)</td>
<td></td>
</tr>
<tr>
<td>Complementarity factor of an entity e</td>
<td>( cf(e) = \frac{1 + \text{triplesUnique}(e)}{</td>
<td>\text{triples}(e)</td>
</tr>
</tbody>
</table>

Table 17: Connectivity Metrics.

5 Warehouse Evolution

The objective here is to investigate how we can understand the evolution of the warehouse and how we can detect problematic cases (due to changes in the remote sources, mistakes in the equivalence rules, addition of a redundant or a “useless” source etc). Let v denote
a version of the warehouse and \( v' \) denote a new version of the warehouse. A number of questions arise:

- Is the new version of the warehouse better than the previous one? From what aspects, the new warehouse is better than the previous one, and from what aspects it is worse?
- Can the comparison of the metrics of \( v \) and \( v' \), aid us in detecting problems in the new warehouse, e.g., a change in an underlying source that affected negatively the new warehouse?

It is also useful to compare a series of versions for:

- understanding the evolution of the entire warehouse over time
- understanding the evolution of the contribution of a source in the warehouse over time

To tackle these questions, we first (in §5.1) describe the datasets (real and synthetic) that were used, and then (in §5.2) we focus on how to inspect a sequence of versions. Finally §5.3 summarizes the drawn conclusions.

## 5.1 Datasets Used

To understand the evolution we need several series of warehouse versions. To this end, we used both real and synthetically derived datasets.

### 5.1.1 Real Datasets

We used 3 real versions of the MarineTLO-based warehouse. Specifically we considered the following versions:

- MarineTLO-based warehouse version 2 (July 2013): 1,483,972 triples
- MarineTLO-based warehouse version 3 (December 2013): 3,785,249 triples
- MarineTLO-based warehouse version 4 (June 2014): 5,513,348 triples

### 5.1.2 Synthetic Datasets

We created a series of synthetic datasets in order to test various aspects, e.g., source enlargements, increased or reduced number of \texttt{sameAs} relationships, addition of new sources (either relevant or irrelevant to the domain), addition of erroneous data, etc.

Tables 18 and 19 show the 9 different versions and their size in triples, while Table 20 shows the changes from version to version. For each version from 1 to 4, we add new “useful” data, allowing in this way to check not only how the integration of the warehouse is affected by the new useful data, but also to observe the extent up to which the value of a source is changed every time. Versions 5, 6, contain “not so good” sources, hence we would expect to see low values for these two sources. Specifically, version 5 includes a source from a different domain, while version 6 contains a redundant source (this source includes triples from Fishbase and WoRMS). Concerning version 7, we have created a possible error that one could face. In particular, we have replaced in the prefix of DBpedia’s URIs ‘/’ with ‘#’, whereas in version 8, we have used an invalid SILK Rule in order to produce some wrong \texttt{sameAs} relationships. Taking all these into account, we expect that the following experiments will allow us to detect all the above errors and will show the real importance of each source in every version.

## 5.2 Inspecting a Sequence of Versions

Here we discuss how one can inspect a sequence comprising more than two versions. Suppose that we have \( n \) versions, \( v_1, v_2, \ldots, v_n \). We can get the big picture by various plots each having in the \( X \) axis one point for each warehouse version. Below we describe several useful plots. For reasons of space some plots are given in the appendix of this paper.
### Table 18: Triples of the synthetically derived versions (versions 1-5).

<table>
<thead>
<tr>
<th>Source</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>904,238</td>
<td>904,238</td>
<td>904,238</td>
<td>904,238</td>
<td>904,238</td>
</tr>
<tr>
<td>WoRMS</td>
<td>62,439</td>
<td>62,439</td>
<td>1,383,168</td>
<td>1,383,168</td>
<td>1,383,168</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>61,597</td>
<td>61,597</td>
<td>61,597</td>
<td>61,597</td>
<td>61,597</td>
</tr>
<tr>
<td>DBpedia</td>
<td>394,373</td>
<td>394,373</td>
<td>394,373</td>
<td>394,373</td>
<td>394,373</td>
</tr>
<tr>
<td>FishBase</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2,332,893</td>
<td>2,332,893</td>
</tr>
<tr>
<td>Clone Source</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Airports</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>121,113</td>
</tr>
<tr>
<td>All</td>
<td>1,422,637</td>
<td>1,422,637</td>
<td>2,743,376</td>
<td>5,076,269</td>
<td>5,197,382</td>
</tr>
</tbody>
</table>

### Table 19: Triples of the synthetically derived versions (versions 6-9).

<table>
<thead>
<tr>
<th>Source</th>
<th>V6</th>
<th>V7</th>
<th>V8</th>
<th>V9</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>904,238</td>
<td>904,238</td>
<td>904,238</td>
<td>904,238</td>
</tr>
<tr>
<td>WoRMS</td>
<td>1,383,168</td>
<td>1,383,168</td>
<td>1,383,168</td>
<td>1,383,168</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>61,597</td>
<td>61,597</td>
<td>61,597</td>
<td>61,597</td>
</tr>
<tr>
<td>DBpedia</td>
<td>394,373</td>
<td>488,989</td>
<td>645,228</td>
<td>782,469</td>
</tr>
<tr>
<td>FishBase</td>
<td>2,332,893</td>
<td>2,332,893</td>
<td>2,332,893</td>
<td>2,332,893</td>
</tr>
<tr>
<td>Clone Source</td>
<td>1,036,194</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Airports</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>All</td>
<td>6,112,463</td>
<td>5,170,885</td>
<td>5,327,124</td>
<td>5,464,365</td>
</tr>
</tbody>
</table>

### Table 20: Description of how each synthetic warehouse version was derived.

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low number of sameAs relationships</td>
</tr>
<tr>
<td>2</td>
<td>Addition of more sameAs relationships</td>
</tr>
<tr>
<td>3</td>
<td>Increase of WoRMS triples</td>
</tr>
<tr>
<td>4</td>
<td>Addition of a new source (FishBase)</td>
</tr>
<tr>
<td>5</td>
<td>Addition of an out-of-domain source (Airports)</td>
</tr>
<tr>
<td>6</td>
<td>Deletion of Airports source and addition of a redundant source (CloneSource)</td>
</tr>
<tr>
<td>7</td>
<td>Deletion of CloneSource and update of DBpedia where the policy of its URIs changed.</td>
</tr>
<tr>
<td>8</td>
<td>Update of DBpedia and production of 4000 wrong sameAs relationships due to an invalid SILK rule</td>
</tr>
<tr>
<td>9</td>
<td>The final version without errors</td>
</tr>
</tbody>
</table>
For each \( v_i \) we plot \(|\text{triples}(W_{vi})|\). Figure 6 shows the resulting plot for the real datasets.

For each \( v_i \) we plot \(|U_{W_{vi}}|\) and \(|\text{Lit}_{W_{vi}}|\) where \( U_{W_{vi}} \) is the set of all URIs and \( \text{Lit}_{W_{vi}} \) is the set of all Literals in the warehouse of that version. Figure 7 shows the resulting plot for the real datasets.

For each \( v_i \) we plot the average degree of the URIs of the warehouse \( \text{deg}_W(U_{W_{vi}}) \) and the average degree of the blank nodes and URIs of the warehouse \( \text{deg}_W(U_{W_{vi}} \cup BN_{W_{vi}}) \). Figure 8 shows the resulting plot for the real datasets.

For each \( v_i \) and for each source \( S_j \) we plot \( value_1(S_i, W) \) as defined in §4.5.2 (one diagram with \( k \) plots one for each of the \( k \) sources). Figure 9 shows how the contribution of the sources in the warehouse evolves, for the real datasets.

For each source \( S_i \) and version \( j \) we plot:

(a) The Jaccard similarity between the set of triples in \((S_i)_{j-1}\) and \((S_i)_j\) (e.g., see the 1st column of Figure 21 in Appendix B)

(b) The normalized number of the triples in every version, defined as the division of the number of a source’s triples with the number of the biggest source’s triples in the warehouse (e.g., see the 2nd column of Figure 21 in Appendix B):

\[
\text{NormalizedTriples}(S_i)_j = \frac{|\text{triples}(S_i)_j|}{\max_{m \in 1..k} |\text{triples}(S_m)_j|} \tag{10}
\]

(c) The normalized average degree defined as the division of the average degree of a source by the maximum average degree among all the sources in the warehouse (e.g., see the 3rd column of Figure 21 in Appendix B):

\[
\text{NormalizedAverageDegree}(S_i)_j = \frac{\text{avgdeg}_S(U_i)}{\max_{m \in 1..k} \text{avgdeg}_S(U_m)} \tag{11}
\]

(d) The value in each version, as it has defined in §4.5.2 (e.g., see the 4th column of Figure 21 in Appendix B):

The first three (i-iii) concern the warehouse per se, while (iv) and (v) show how the contribution of the source in the warehouse evolves. Finally, Table 21 shows the average degree increment of the URIs and blank nodes of each source for the 3 different versions of the real datasets.

<table>
<thead>
<tr>
<th>Source</th>
<th>Version2</th>
<th>Version 3</th>
<th>Version 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOD</td>
<td>465.59%</td>
<td>793.64%</td>
<td>797.61%</td>
</tr>
<tr>
<td>WoRMS</td>
<td>548.67%</td>
<td>97.82%</td>
<td>103.61%</td>
</tr>
<tr>
<td>Ecoscope</td>
<td>108.97%</td>
<td>325.58%</td>
<td>396.84%</td>
</tr>
<tr>
<td>DBpedia</td>
<td>271.84%</td>
<td>522.75%</td>
<td>505.65%</td>
</tr>
<tr>
<td>FishBase</td>
<td>——</td>
<td>117.02%</td>
<td>58.43%</td>
</tr>
</tbody>
</table>

Table 21: Average degree increment percentages for the URIs and blanks nodes of each source in every version.

Results over the Synthetic Datasets

Common URIs/Literals. Concerning common URIs, Figure 10 shows how the percentages of common URIs (according to policy [iii]) changed from version to version among DBpedia and all the other sources. As we can see, the percentages increased in version 2 comparing to version 1, because of the production of the \texttt{sameAs} relationships.
On the contrary, in version 7 the percentages of common URIs were heavily reduced, since the change in DBpedia’s URIs affected negatively the production of the sameAs relationships. Moreover, the percentages increased in version 8, predominantly because of the wrong sameAs relationships while in the last version, the increase in the common URIs percentage between DBpedia and FishBase is remarkable.

Regarding common Literals, as we can see in Figure 11 the percentages didn’t change so much from version to version. However, it is clear that the highest percentage exists between DBpedia and FishBase, exactly as it happened with common URIs percentage. Therefore, it is obvious that DBpedia shares more common information with FishBase than with all the other sources.

Average Degree Increment. Figure 12 shows the normalized increment degree percentage of each source. As we can see, it is evident that a source containing useful data has a positive effect on the percentages of all the other sources. For instance, when we added the source FishBase (in version 4), the percentages of all the other sources were notably increased, while when we added the Airports source (version 5), the percentages remained almost the same.

In version 7, the change of the URI policy in DBpedia reduced the sameAs relationships in the warehouse. As a result, it affected negatively the percentages of all the
sources (predominantly those from DBpedia). Despite the fact that there are considerable benefits to see big values in this measure, sometimes there is also an important drawback that cannot be ignored. For instance, in version 8, as we can see better in Figure 13, the percentages were highly increased because of an invalid SILK rule which produced 4,000 wrong sameAs relationship, leading to significant negative consequences for the warehouse. For instance, the queries will return a lot of imprecise results. Therefore, having big values for integration has no meaning if the precision is low.

Unique Triples Percentage. As regards unique triples, Figure 14 shows the unique triples percentage of each source. The addition of CloneSource in version 6 had as a result the reduction of WoRMS and FishBase percentages. In fact, CloneSource shared a lot of triples with these two sources. Therefore, through this diagram one can easily observe that CloneSource is redundant, while one can see the remarkable change in the unique triples percentage of WoRMS because of the addition of FishBase in version 4.

Complementarity Factor. Regarding complementarity factor, Figure 15 shows how the updates affected the complementarity factor of a specific entity. In versions 1-3, only 2 (out of 4) sources provided unique triples about “Astrapogon”. In version 4, the complementarity factor of this entity increased because FishBase contained unique data about it. On the other hand, the out-of-domain source that we added in version 5 had negative impact for this measure. It is worth-mentioning that the CloneSource, which included all the triples about Astrapogon from WoRMS and FishBase, heavily reduced the complementarity factor in version 6. Finally, the addition of the new triples of DBpedia and the deletion of CloneSource had considerable advantages for the value of this measure. Therefore, in the last 3 versions, 4 sources (out of 5) provided unique data about “Astrapogon”.

Value of Sources. Figure 16 depicts the values of each source in all different versions.
Initially, there is no doubt that in the first two versions the most important source was FLOD, since it had high values for the average degree increment and the unique triples percentage, and was consisted of a large number of triples. Furthermore, we can see that in version 2 the value of each source was increased because of the production of the \texttt{sameAs} relationships. Regarding version 3, the increment of the size of WoRMS had negative effects on the values of the other sources. Therefore, the value and possibly the ranking of the sources can be affected when we add to the warehouse a bigger source with useful triples. In this way, FishBase value surpassed all the other sources values in version 4. Concerning version 5, the new source (Airports) that we added seemed to be useless because of the low average degree increment percentage. This is rational, because it provided information which is useless for our warehouse. In version 6, we added a source (CloneSource) containing approximately 1,500 new triples and a lot of triples which were already existed in the warehouse and specifically, in FishBase and WoRMS. Indeed, despite the fact that this source was consisted of a lot of triples, its value was the lowest. Moreover, by adding these triples, the values of the sources sharing common triples with CloneSource were affected, too. In fact, the reduction of the value of FishBase is remarkable although the contents of this source were exactly the same as in version 5. As regards version 7, DBpedia was affected negatively because of the change in the policy of it’s URIs. In version 8, DBpedia and FishBase were mostly benefited by the production of the wrong \texttt{sameAs} relationships, since their normalized average degree increased to a great extent comparing to the other sources. Finally, it is worth mentioning that the value of DBpedia increased in version 9, since there was added a large number of unique triples for this source.

\textbf{Measurements per Source.} In Appendix C we show the values of several metrics for each source for all versions.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure16.png}
\caption{Value of each source per version (using $value_1(S_i, W)$).}
\end{figure}

\section{Executive Summary Regarding Evolution}

Taking all the aforementioned into consideration, we managed through the above experiments to meet the main objectives. In particular, through the real datasets we saw the evolution of a real warehouse. Indeed, by observing the changes in the \textit{Average Degree Increment percentage}, we concluded that the average Degree percentage of the warehouse
URIs increased from version to version. However, by taking into account not only the URIs but also the blank nodes, we observed that the percentage decreased from version to version. Additionally, through the single-valued metric Value, we saw the most important sources in every version and the contribution of each source in the warehouse over time.

As regards the synthetic datasets, we managed to detect a number of problematic cases. Particularly, we found the problematic cases in versions 5,6,7,8 predominantly through Value, Average Degree Increment percentage and Unique Triples percentage. Specifically, we identified that Airports source was out-of-domain, because both Value and Average Degree Increment percentage of this source were very low. On the contrary, in version 6, it was easy to understand that CloneSource was redundant, since, its Value and Unique Triples percentage were very low. In addition to this, it was remarkable that the values of the sources sharing common triples with this source were affected to a great extent. In particular, the Unique Triples percentage of these sources decreased greatly.

Concerning version 7, the large decrease in the Average Degree Increment percentage of DBpedia and in the common URIs percentage between DBpedia and each of the remaining sources indicated that probably either the schema or the policy of this source’s URIs changed. However, such a large decrease can be also arisen when the content of a source changed while neither the schema nor the policy of URIs changed. For being sure that this decrease arose due to some incorrect rules (due to changes in source’s schema or URIs policy) that create sameAs relationships one can measure the common URIs percentage between the pairs of sources with policy [ii]. In particular, if the common URIs percentage between a pair of sources with policy [ii] remains high (as in a previous version), but the Average Degree Increment percentage and the common URIs percentage with policy [iii] decreased to a great extent, it means that surely the rules for creating sameAs relationships should be modified. On the contrary, in other cases a lot pairs of URIs of two different sources referring to the same real world object contains different suffixes. In such cases, only policy [iii] can show how connected is a pair of sources while policy [ii] cannot help us when there is a decrease in the results of measurements that use policy [iii].

Finally, in version 8, the huge increase in the Average Degree Increment percentage was unexpected. Indeed, the production of more and more sameAs relationships always improves the Average Degree Increment percentages. However, if the percentages are far higher than the previous versions, it is highly possible that there have been produced a lot of wrong sameAs relationships, probably due to an invalid SILK rule.

5.4 3D Visualization

Since there are many parameters the problem of understanding can be quite complex. To further alleviate the difficulty, below we propose a 3D visualization that can aid the user to get an overview.

We have adopted a quite familiar metaphor, specifically that of a urban area. The main idea is to visualize each source of the warehouse as a building, while the proximity of the buildings is determined by their common URIs and literals (note that the number of sameAs relationships between their URIs could also be considered). Finally, the number of sameAs relationships created by instance matching, can be visualized as bridges (or strings) between the buildings.

In particular, each source $S_i$ corresponds to a building $b_i$. The volume of the building corresponds to $|\text{triples}(S_i)|$. Since $|\text{triples}(S_i)| \approx (|U_i| + |Lit_i| + |BN_i|) \cdot \text{Deg}(U_i)$, (where $BN_i$ denotes the set of blank nodes of the source $i$) we decided to set the height of the building analogous to $|U_i| + |Lit_i| + |BN_i|$, and the footprint of the building analogous to $\text{degs}_i(U_i)$. Specifically, assuming square footprints:

$$\text{height}(b_i) = |U_i| + |Lit_i| + |BN_i|$$

(12)
width(b_i) = \sqrt{\text{deg}_{S_i}(U_i)} \tag{13}

In this way, the volume of the building \( b_i \) approximates \( |\text{triples}(S_i)| \); if its degree is low it will become a high building with a small footprint, whereas if its degree is high then it will become a building with big footprint but less tall. For getting building sizes that resemble those of a real urban area, a calibration is required, specifically we use an additional parameter \( K \), through which we can obtain the desired average ratio of height/width of the buildings. The new formulas are:

height(b_i) = (|U_i| + |Lit_i| + |BN_i|)/K \tag{14}

width(b_i) = \sqrt{\text{deg}_{S_i}(U_i) \times K} \tag{15}

Urban building usually have height > width, so one approach for setting automatically the value for \( K \) is to select the smallest \( K \) such \( \text{height}_{\text{avg}} > \text{width}_{\text{avg}} \). If one prefers to have 3 floor buildings he can select the smallest \( K \) such that \( h_{\text{avg}} > 3 \times \text{width}_{\text{avg}} \). In our case for a desired ratio of around 5, we used \( K = 500 \). Table 22 shows the building sizes of warehouse version 4.

| \( S_i \) | \( |\text{triples}(S_i)| \) | \( |U_i| \) | \( |\text{Lit}_i| \) | \( |BN_i| \) | \( \text{avg \ deg}_{S_i}(U_i) \) | Height | Width |
|---|---|---|---|---|---|---|---|
| FLOD | 904,691 | 159,042 | 115,573 | 27,812 | 6.28 | 604,8 | 167.9 |
| WoRMS | 1,382,748 | 51,649 | 217,759 | 271,352 | 4.43 | 1081.2 | 67.2 |
| Ecoscope | 2,332,739 | 33,089 | 490,940 | 9,981 | 1344.9 | 68.9 |
| FishBase | 88,714 | 12,027 | 25,901 | 0 | 7.55 | 75.9 | 62.5 |

Table 22: Buildings’ sizes.

Each building has a location, i.e., \( x \) and \( y \) coordinates. The more common URIs and literals two source have, the closer the corresponding buildings should be. Below we describe the approach for computing the locations of the buildings. We can define the similarity between two sources (based on their common URIs/literals and sameAs relationships), denoted by \( \text{sim}(S_i, S_j) \), as follows:

\[
\text{sim}(S_i, S_j) = \frac{1}{2} \left( \frac{|U_i \cap U_j|}{\text{min}(|U_i|, |U_j|)} + \frac{|\text{Lit}_i \cap \text{Lit}_j|}{\text{min}(|\text{Lit}_i|, |\text{Lit}_j|)} \right) \tag{16}
\]

From \( \text{sim}(S_i, S_j) \) we can compute the distance as follows:

\[
\text{dist}(S_i, S_j) = \frac{1}{\text{sim}(S_i, S_j)} \tag{17}
\]

Then we adopt a force-directed placement algorithm (similar in spirit with that of [51]) for computing \( x \) and \( y \) coordinates. Let \( \equiv_{i,j} \) denote the set of sameAs relationships between the URIs of \( S_i \) and \( S_j \). Each \( \equiv_{i,j} \) can be visualized as a bridge that connects \( b_i \) and \( b_j \). The volume of the bridge can be analogous to \( |\equiv_{i,j}| \). Three snapshots...
from different points of view of the produced 3D model are shown in Figure 17. The model can give an overview of the situation in a quite intuitive manner. The relative sizes of the buildings allow the user to understand the relative sizes in triples of the sources. The proximity of the buildings and the bridges make evident the commonalities of the sources, while the side size of each building indicates the average degree of the graph of each source. Furthermore, the user can navigate into the model using a Web browser, in this example through the following address http://62.217.127.128:8080/warehouseStatistics/Statistics.html.

![Figure 17: Three snapshots from different points of view of the produced 3D model.](image)

As future research it would be interesting to investigate using animations. For instance, each version $v_i$ of the warehouse can be considered as a state of the area $a_{vi}$. An animation that starts from $a_{vi}$ and is progressively transformed to $a_{vj}$ could be used for showing what has happened between two versions $vi$ and $vj$. For our case the sizes and the locations of the buildings were similar for the versions 3 and 4 of the warehouse, therefore we illustrate only the visualization of the latest one.

6 Implementation and Exploitation of the Metrics

Here we discuss how the proposed metrics can be computed and published. The entire life cycle of these metrics is illustrated in Figure 18. In brief, the metrics can be computed using SPARQL queries, and the resulting measurements can be represented using an extension of VoID (which is briefly described later). Subsequently, they can be stored allowing an external service to query and exploit them, e.g., for dataset and endpoint selection. In particular, such descriptions could be exploited in a virtual integration approach (for source/warehouse selection) and in distributed query processing in general. For instance, a mediator could collect these descriptions and query them for obtaining the SPARQL endpoints that satisfy certain criteria. For example, one might want to find those SPARQL endpoints that contain integrated information from the X and Y particular primary sources (e.g., WoRMS and DBpedia).
It is also worth noting that although in the previous sections we have defined the metrics over all URIs and literals, they can be straightforwardly applied also for producing various topic-based data summaries. For instance, they can be computed only for URIs that are instances of Species, or instances of WaterAreas, and so on. Such descriptions would enable a topic-based source selection algorithm to select the endpoints to query, as well as to estimate the upper bound of size of the query answer. So the metrics can be considered a kind of data summaries [23] for semantic warehouses.

Below at first (§6.1) we discuss how we can compute these metrics using SPARQL and then (in §6.2) we describe a way for publishing and exchanging these metrics.

6.1 Computing the Connectivity Metrics using SPARQL queries

Here we show how the values of the connectivity metrics can be computed using SPARQL queries (this corresponds to the process “1. SPARQL compute” of Figure 18). Some of the SPARQL queries contain some specific rules (e.g., define input:same-as "yes") these rules are used by Virtuoso10 for enabling inference for triples containing owl:sameAs relations.

Common URIs. The metric Common URIs over two sources $S_i$ and $S_j$, can be computed with the following query:

\[
\text{SELECT COUNT (DISTINCT ?o) WHERE } \{ \text{GRAPH :Si \{} \{?s1 ?p1a ?o\} \text{ UNION } \{?o ?p1b ?o1\}\text{ FILTER(isURI(?o))} \text{ GRAPH :Sj \{} \{?s2 ?p2a ?o\} \text{ UNION } \{?o ?p2b ?o2\}\text{ FILTER(isURI(?o))} \} \}
\]

In the context of the warehouse, this metric should be computed over all pairs of sources, i.e., all $(S_i, S_j)$ such that $S_i, S_j \in S$ and $i \neq j$. Note that this metric is symmetric, i.e., the value of the pair $(S_i, S_j)$ is equal to the value of $(S_j, S_i)$.

Regarding policy [iii], one can use the following query in order to collect all the matching pairs between 2 sources.

```
DEFINE input:same-as "yes"
SELECT DISTINCT (bif:concat(?o,'\t',?s)) as ?matchingPair
FROM :Si
FROM :Sj
FROM :SameAs
WHERE { 
    \{GRAPH :SameAs \{(?s ?p ?o) UNION (?o ?p ?s)\} \}
    .GRAPH :Si
    \{(?s ?p1 ?o1) UNION (?o1 ?p1 ?s) .filter(isURI(?s))\}
    .GRAPH :Sj
    \{(?o ?p2 ?o2) UNION (?o2 ?p2 ?o) \} . filter(isURI(?o)) \}
```

10http://virtuoso.openlinksw.com/
Common Literals. The Common Literals between two sources $S_i$ and $S_j$ can be computed in a similar manner, i.e.:

```sparql
SELECT COUNT DISTINCT ?o
WHERE { graph :Si { ?s ?p ?o} . FILTER(isLiteral(?o))
  graph :Sj { ?a ?b ?o} }
```

Again, this metric should also be computed over all pairs $(S_i, S_j)$ of the warehouse.

Unique Triples Contribution. To compute the unique triple contribution of a source, say $S_1$, to the warehouse $S = S_1, \ldots, S_k$, we have to count the number of triples of $S_1$ that do not intersect with the triples of any of the other sources of $S$ (i.e., with none of the sources in $S_2, \ldots, S_n$). This can be done using the following query:

```sparql
SELECT COUNT(*)
WHERE { graph :S1 { ?s ?p1 ?o} .
  FILTER NOT EXISTS { graph :S2 { ?s ?p2 ?o} } .
  \ldots
  FILTER NOT EXISTS { graph :Sn { ?s ?pn ?o} } }
```

On the contrary, concerning policy [ii] one can use the following SPARQL query in order to get the suffixes of the subject and object of each triple for every source. After that, one can use the produced sets in order to calculate the unique triples for each source.

```sparql
SELECT bif:lower(bif:regexp_substr('[^#|/]+$',?s,0)) as ?s
  bif:lower(bif:regexp_substr('[^#|/]+$',?o,0)) as ?o
FROM :Si
WHERE {?s ?p ?o}
```

Complementarity Factor. This metric is computed for a specific entity over all sources of the warehouse. In particular, the complementarity factor of an entity $e$ is increased for each source $S_i \subseteq S$ that contains at least one unique triple having the entity $e$. This means that if all sources in $S$ contain unique triples for $e$, then its complementarity factor will be 1.0. The query below gives the complementarity factor of an entity $e$ over $S$. Notice that the WHERE clause contains $n$ graph patterns. In the SELECT statement $n$ denotes the number of sources and it is provided by the user. Each graph pattern $i$ returns 1, if $S_i$ contains unique triples for the entity $e$, otherwise it returns 0.

```sparql
SELECT (?CF1+ \ldots + ?CFn)/n AS ?CF
WHERE { ( SELECT xsd:integer(COUNT(*)>0) as ?CF1
  WHERE { ( graph :S1 { ?s ?p1 ?o } )
    FILTER NOT EXISTS { graph :S2 { ?s ?p2 ?o} } .
    \ldots
    FILTER NOT EXISTS { graph :Sn { ?s ?pn ?o} }
    FILTER (regex(?s, e,'i') || (regex(?o, e,'i'))) } )
  \ldots
  ( SELECT xsd:integer(COUNT(*)>0) as ?CFn
    WHERE { ( graph :Sn { ?s ?pn ?o } )
      FILTER NOT EXISTS { graph :S1 { ?s ?p1 ?o } .
      \ldots
      FILTER NOT EXISTS { graph :Sn-1 { ?s ?pn-1 ?o } }
      FILTER (regex(?s, e,'i') || (regex(?o, e,'i'))) } )
}
```

Increase in the Average Degree. Let $E$ be a set of entities coming from a source $S_i$. To compute the increase in the average degree of these entities when they are added into the warehouse, the following query computes both average values (before and after the addition to the warehouse) and reports the increase. Note that that above query considers the “entity matching” policy [iii].
Regarding the average degree of all the URIs and blank nodes of a source, one can use the following query.

```
SELECT xsd:double(count(?incomingProperties)+count(?outgoingProperties))
/xsd:double(count (distinct ?node)) as ?sourceDegree
FROM :Si
WHERE { {?node a :SiURI} .
{(?node ?outgoingProperties ?o .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
UNION
(?o1 ?outgoingProperties ?node .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))} }
```

Finally, for the average degree of the URIs and blank nodes of a source in the warehouse, one can use the following two queries for URIs and blank nodes, respectively. We use a temporary graph to store all the URIs of each graph, since this way was proved more efficient in order to measure the degree of each source in the warehouse. Moreover, one can find the average degree of the union of its URIs and blank nodes by using the following formula:

\[
\text{avgdeg}_W(U_W \cup BN_W) = \frac{\text{avgdeg}_W(U_W) \cdot |U_W| + \text{avgdeg}_W(BN_W) \cdot |BN_W|}{|U_W| + |BN_W|} \tag{18}
\]

The following SPARQL queries can be used for computing the average degree for entities that are URIs and blank nodes respectively.

```
DEFINE input:same-as "yes"
SELECT xsd:double(count(?incomingProperties)+count(?outgoingProperties))
/xsd:double(count (distinct ?node)) as ?avgURIsDegree
count (distinct ?node) as ?URIs
FROM :Wi
WHERE { GRAPH :temp
{?node a :SiURI} .
{(?node ?outgoingProperties ?o .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
UNION
(?o1 ?outgoingProperties ?node .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))} }
```

```
DEFINE input:same-as "yes"
SELECT xsd:double(count(?incomingProperties)+count(?outgoingProperties))
/xsd:double(count (distinct ?node)) as ?avgBnodesDegree
count (distinct ?node) as ?Bnodes
FROM :Wi
WHERE { {?node a :SiURI} .
{(?node ?outgoingProperties ?o .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
UNION
(?o1 ?outgoingProperties ?node .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))} }
```

```
DEFINE input:same-as "yes"
SELECT xsd:double(count(?incomingProperties)+count(?outgoingProperties))
/xsd:double(count (distinct ?node)) as ?avgURIsDegree
FROM :Wi
WHERE { GRAPH :temp
(?node a :SiURI) .
{(?node ?outgoingProperties ?o .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
UNION
(?o1 ?outgoingProperties ?node .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))} }
```

```
DEFINE input:same-as "yes"
SELECT xsd:double(count(?incomingProperties)+count(?outgoingProperties))
/xsd:double(count (distinct ?node)) as ?avgBnodesDegree
FROM :Wi
WHERE { (?node a :SiURI) .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
UNION {?o1 ?incomingProperties ?node .
FILTER (?outgoingProperties!=<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>)} }
```
Time efficiency

Table 24 shows the query execution times (in minutes) for computing the metric Common URIs for each of the three policies, i.e., (i) Exact String Equality, (ii) Suffix Canonicalization and (iii) Entity Matching. The first row corresponds to the pure SPARQL approach that was presented earlier. The second row corresponds to a hybrid approach, where more simple queries are used for getting the resources of interest (i.e., the two sets of URIs, one for each source \(S_i, S_j\)), and Java code is used for computing their intersection. We observe that the hybrid approach is faster than the pure SPARQL, as the comparisons are implemented faster in Java. In general, we have observed that the hybrid approach loses in time efficiency when the implemented queries return a big amount of data (as in the case of Unique Triples Contribution), while it is faster (than pure SPARQL) in comparisons.

<table>
<thead>
<tr>
<th>Computation Method</th>
<th>Policy [i]</th>
<th>Policy [ii]</th>
<th>Policy [iii]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pure SPARQL</td>
<td>7</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>hybrid</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 24: Times (in min) needed to compute metrics on various approaches and policies.

Regarding policy [ii], the pure SPARQL approach becomes less efficient, as the string comparisons cost more when implemented over the endpoint. When adopting policy [iii], both approaches are increased by 1 minute. This uniform increase is reasonable as an additional graph that contains the triples with the sameAs properties is taken into account.

6.2 On Publishing and Exchanging metrics

For publishing and exchanging these metrics we have used VoID [30]. VoID is an RDF-based schema that allows expressing metadata about RDF datasets. It is intended as a bridge between the publishers and the users of RDF data by making the discovery and the usage of linked datasets an effective process. Note also that, according to [41], about 15% of LOD datasets use VoID. The design of VoID has been driven by representing a number of both domain-dependent features (e.g., which type of data it contains) and domain-independent ones (e.g., who published it). Conceptually, it has been built around the notions of void:Dataset, void:Linkset and RDF Links. A void:Dataset is a set of RDF triples that are published, maintained or aggregated by a single provider. A void:Linkset is a collection of RDF Links between two datasets. An RDF Link is an RDF triple whose subject and object are described in different void:Dataset. Based on Dublin Core [40], VoID provides properties that can be attached to both void:Dataset and void:Linkset to express metadata.

Apart from VoID, we have also used a particular extension [38], which is sufficient for describing our metrics. In brief, we could say that this extension allows the values of the various metrics to be expressed in a machine processable (and query-able) manner. If such information is exposed in a machine-readable format, it could be exploited in various ways, e.g.:

- For producing visualizations that give an overview of the contents of a warehouse.
For comparing different warehouses and producing comparative reports.
For aiding the automatic discovery of related data since software services/agents based on these metrics could decide which SPARQL endpoints to query based on time/cost constraints.
For crediting good sources since these metrics make evident, and quantifiable, the contribution of a source to the warehouse.

An illustration of the main concepts we are using from VoID and its extension is depicted in Figure 19.

We can see that there is the notion of measurement which is actually a specialization of event and therefore inherits the property date. A measurement is carried out by an agent using a specific metric according to one (or more) configurations over one (or more) datasets (atomic or composite) and produces a value (i.e., literal).

Each metric is an individual which means that it is assigned a URI and is defined by a resource (e.g., the DOI of the scientific paper that defined that metric). The notion of configuration concerns issues that explain how the measurement was done. Furthermore we use two specializations of the configuration class; the first concerns the way URI equivalence is defined, while the second concerns how the entities of interest are defined. Regarding the latter the current modeling allows someone to specify the desired set of entities by providing a SPARQL query that returns them.

7 Concluding Remarks

In many applications one has to fetch and assemble pieces of information coming from more than one source. 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. By inspecting the proposed metrics-based matrices one can very quickly get an overview 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, (d) the unique triple contribution of each source, and (e) a single-valued metric for quantifying the value of a source. The values of (a), (b), (c) allow valuating the warehouse, while (c), (d) and (e) mainly concern each particular source. For instance, by combining the unique triples contribution and the increment of the average degrees, 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. In addition we introduced metrics and plots suitable for monitoring the evolution of a warehouse. More specifically, we exploit the aforementioned metrics to understand how the warehouse evolves and how the contribution of each source changes over time. To this end, we provide a set of plots that allow someone to quickly spot anomalies. For aiding the user to get an overview of the situation in an intuitive manner, we have introduced a visualization method for these metrics that relies on 3D modeling.

The ability to assess the quality of a semantic warehouse, using methods like those presented in this paper and those in the literature, is very important also for dataset and endpoint selection, as well as judging whether the warehouse can be used in e-Science. 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 maintenance and preservation.

Acknowledgement

This work was partially supported by the projects: iMarine (FP7 Research Infrastructures, 2011-2014), BlueBRIDGE (H2020 Research Infrastructures, 2015-2018) and Life-Watch Greece (National Strategic Reference Framework, 2012-2015).

References


A Competency Queries

Figure 20 displays the textual description for some indicative competency queries as they were supplied by the communities.

<table>
<thead>
<tr>
<th>Query</th>
<th>For a scientific name of a species (e.g. Thunnus Albacares or Poromitra Crassiceps), find/give me</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q₁</td>
<td>the biological environments (e.g. ecosystems) in which the species has been introduced and more general descriptive information of it (such as the country)</td>
</tr>
<tr>
<td>Q₂</td>
<td>its common names and their complementary info (e.g. languages and countries where they are used)</td>
</tr>
<tr>
<td>Q₃</td>
<td>the water areas and their FAO codes in which the species is native</td>
</tr>
<tr>
<td>Q₄</td>
<td>the countries in which the species lives</td>
</tr>
<tr>
<td>Q₅</td>
<td>the water areas and the FAO portioning code associated with a country</td>
</tr>
<tr>
<td>Q₆</td>
<td>the presentation w.r.t Country, Ecosystem, Water Area and Exclusive Economical Zone (of the water area)</td>
</tr>
<tr>
<td>Q₇</td>
<td>the projection w.r.t. Ecosystem and Competitor, providing for each competitor the identification information (e.g. several codes provided by different organizations)</td>
</tr>
<tr>
<td>Q₈</td>
<td>a map w.r.t. Country and Predator, providing for each predator both the identification information and the biological classification</td>
</tr>
<tr>
<td>Q₉</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 20: Some indicative competency queries.

B Inspecting the Evolution of a Sequence of Versions (Real Datasets)

Figure 21 shows the values of several metrics for each source for a sequence of versions (regarding the real datasets). Specifically, the 1st column shows the Jaccard distance between the versions, the 2nd column shows the normalized value of the triples in each version, the 3rd column shows the normalized average degree in each version, and the 4th column shows the value in each version. Regarding the Jaccard distance, one can observe that the source WoRMS completely changed from version to version. On the other hand, DBpedia had the smallest change among all sources. Concerning the value of the normalized triples, the addition of Fishbase and the increase in the number of triples of WoRMS had a huge impact in the normalized triple value of the other sources. Moreover, DBpedia, has the biggest average degree in all versions while Fishbase and WoRMS have a consecutive increase in their values from version to version.

C Inspecting the Evolution of a Sequence of Versions (Synthetic Datasets)

Figure 22 shows the values of several metrics for each source (regarding the synthetic datasets). Specifically, the 1st column shows the Jaccard distance between the versions, the 2nd column shows the normalized value of the triples in each version, the 3rd column shows the normalized average degree in each version, and the 4th column shows the value in each version. Firstly, one can easily see the change in the Jaccard distance of DBpedia from version 6 to version 7. In this case, the change of the policy of DBpedia’s URI had a negative consequence for the value of DBpedia. By observing this figure, one can realize that the aforementioned problem possibly occurred because of this sudden change in
Figure 21: Measurements per source for the real datasets.
Figure 22: Measurements per source for the synthetic datasets.

DBpedia’s Jaccard distance. As regards the values of the normalized triples, FLOD had the biggest decrease because of the insertion or the increase of other sources. The most remarkable figures concerns the value of each source. We can observe the variations in each source’s value due to the pathological case that we saw previously.