Linked Product Data

Rita Peres Cruz Curado

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Miguel Filipe Leitão Pardal
Prof. José Manuel da Costa Alves Marques

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Miguel Filipe Leitão Pardal
Member of the Committee: Prof. Bruno Emanuel da Graça Martins

May 2016

To my grandfather who always supported me throughout this journey, even though he could not witness its outcome

Acknowledgments

I would like to thank my advisor Miguel Pardal, for all the support and for always being available to help me clear my thoughts, motivate me and also help me review my work. I would also like to thank my family and friends who never allowed me to give up whenever the obstacles seemed insuperable. And last but not least, I would like to thank all the technical and common users who helped me evaluate this work.

Abstract

Over the years, the Web has become a place accessible to everyone and that fact was responsible for the increasing amount of data that is available today. The AAA slogan which means that “Anyone can say Anything about Any topic” describes the open nature of the Web. In this so-called “Web of Data” the task of search is becoming harder. When someone performs a search in an engine, like Google, several web pages are returned, that are related to the search by the words given by the user, but most of those web pages are documents without a well defined data schema. This makes it difficult to reference information and to compare it accurately when it comes from different sources. The aim of this work was to develop an identity mapping engine for informational entities based on Semantic Web Technologies that can find similarities between data that is kept in different data sources and presented in different web sites, and decide which of these sources are referencing the same real world objects. With that information it is possible to create links between the different data sources and offer richer information to the users, by merging disjoint information. An application for mapping and merging product data in the Web was developed and evaluated, in different scenarios.

Keywords: Linked data, identity mapping, Semantic Web, RDF, SPARQL.

Resumo

Ao longo dos anos a Web tornou-se um lugar acessível para todos e esse facto foi responsável pela quantidade crescente de dados disponíveis hoje. O slogan AAA - “Anyone can say Anything about Any topic”, que significa que qualquer um pode dizer qualquer coisa sobre qualquer assunto descreve bem a natureza aberta da Web. Na tão conhecida “Web of data” a tarefa de pesquisa tem-se tornado cada vez mais difícil. Quando alguém realiza uma pesquisa num motor de busca, como o Google, várias páginas da web são retornadas, as quais estão relacionadas com a mesma pelas palavras dadas pelo utilizador. Contudo, a maioria das páginas retornadas pela web não possuem uma estrutura de dados bem definida, o que torna difícil a referenciação da informação e sua comparação quando provêm de diferentes fontes. O objetivo deste trabalho foi assim desenvolver um motor de mapeamento de identidade entre entidades informacionais, com base em tecnologias de Web Semântica, capazes de encontrar semelhanças entre os dados que são mantidos em diferentes fontes de dados e apresentados em diferentes páginas, e decidir quais dessas fontes fazem referência ao mesmo objeto do mundo real. Com essa informação, será possível criar vínculos entre essas diferentes fontes de dados e oferecer informações mais ricas aos utilizadores, através da fusão de informação disjunta. Uma aplicação de mapeamento e fusão de dados de produtos na Web foi desenvolvida e avaliada em diferentes cenários.

Palavras-chave: Associação de dados, mapeamento de identidade, Semantic Web, RDF, SPARQL.

Contents

List of Tables

List of Figures

Acronyms

AAA “Anyone can say Anything about Any topic”

CNPEM Código Nacional para a Prescrição Eletrónica de Medicamentos

CSV Comma Separated Values

DB Data Base

FI Folheto Informativo

HTTP Hypertext Transfer Protocol

INFARMED Autoridade Nacional do Medicamento e Produtos de Saúde, I. P.

JSON JavaScript Object Notation

LinkedMDB Linked Movie Data Base

LOD Linked Open Data

OWL Ontology Web Language

RCM Resumo das Características do Medicamento

RDF Resource Description Framework

RDFS Resource Description Framework Schema

SPARQL SPARQL Protocol And RDF Query Language

SQL Structured Query Language

SWE Semantic Web Engine

TSV Tab Separated Values

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

XML Extensible Markup Language

Chapter 1

Introduction

In the old days, when someone wanted to know something about a certain topic s/he had to search for information in an encyclopedia or go to a library and search for it in specific books. Nowadays the information is electronic and spread all over the world, accessible everywhere and kept in several formats which is collectively known as the Web. However, despite all of its advantages, the Web has become a confusing place because there are many sources of information about the same real objects but with inconsistent values for some properties. Therefore, it is on the user's hands to perform the search, decide which sources are reliable and understand which of those are referencing the same real object. This burden should be relieved.

1.1 Motivation

Search engines crawl the Web starting from some specific Uniform Resource Locator (URL) trying to find, in the content of those references, the list of words given by the user. Then they return a list of web pages, indexed by ID, where the number of occurrences of the given words is shown to be higher. After that, these engines give back to the user a list of web pages sorted by word occurrence ranking [1]. Figure 1.1 shows the work flow of this approach.

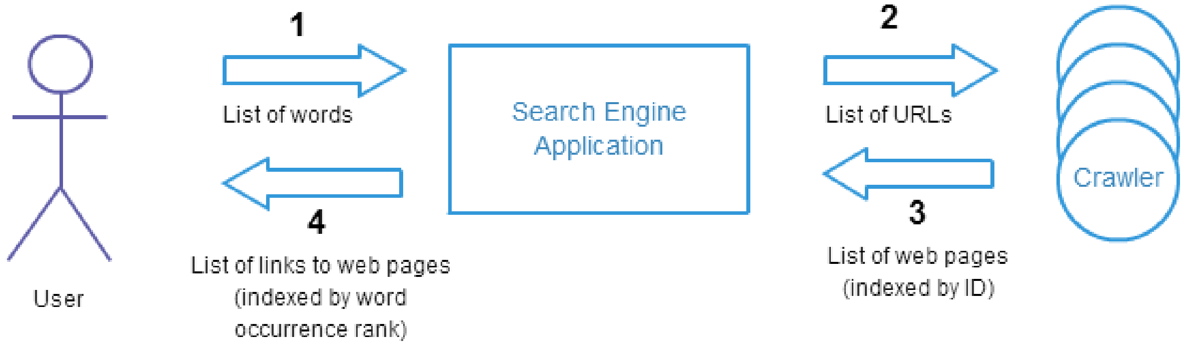

1
List of words
4
List of links to web pages (indexed by word occurrence rank)
User
Search Engine Application
2
List of URLs
3
List of web pages (indexed by ID)
Crawler

Figure 1.1: Current Search Engines Architecture

The former approach leads the user to collect all the necessary information about a topic, by her/himself. Each web page returned by the engine will have information also contained in other web pages, but that page can contain unique information. Therefore, the user is required to visit all the sites returned (or a reasonable number of them) to collect the information s/he is searching for. Moreover, the user is responsible for the filtering of information, i.e., he has to decide which are the web pages that reference the intended product.

Let us see a specific example of this: When a user searches on the Web for a specific smarthphone, s/he will get the manufacturer web page, a web page with reviews about the phone's features, a web page with comparisons between the specified camera and others, some retailers' web pages with prices and retailer's web pages with the phone's accessories. Therefore, if the user wants to collect official information but also reviews, prices and comparisons, s/he will have to visit all the web pages returned by the search. Also, s/he will have to filter the information retrieved to only visit the web pages that are about the smartphone and exclude the ones that, despite of the similar name of the product, do not refer to the phone itself like the accessories page. Obtaining complete and unified information about a topic could be very useful to the users. In the example, this consolidated information would help her/him make a better purchase decision.

1.2 Challenges

Search engines are optimized for quick answers but not for providing complete and unified information about a topic. To achieve this it is necessary to merge data, which, at first, might not seem a very difficult task, however the sources of information differ in their structure, therefore it is essential to standardize the data before mapping and merging it. Beyond that, inside each data source there are duplicated information which needs to be initially filtered, to not hamper the identity, mapping and merging processes.

1.2.1 Identity Problem

Data is kept in different databases created by different people with different points of view and different interpretations about things. This is why, to describe the same smartphone's model, certain databases will have the attribute “Name” and others will have the attribute “Model”, although both attributes give the same information. The same applies to the identification of products where, according to a specific database, the product has a specific identifier that is unique inside that scope. However, many databases could describe the exactly same product with different identifiers, which might mislead us to think that they are representing different objects. This issue is known as the identity problem.

In the Semantic Web entities are differentiated through its Uniform Resource Identifier (URI) to ensure their uniqueness, therefore Figure 1.2 illustrates the problem of ”How can we decide if two objects with completely different URIs are actually the same?”

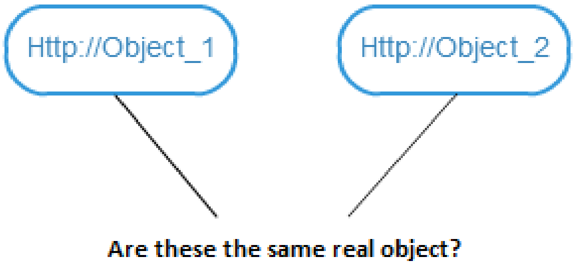


Figure 1.2: Products with different URIs referring to the same real object

Since entities have a unique identifier, given by an URI, that is different according to the source they belong to, there had to be a way to find if two entities were referring the same real object based on their attributes.

1.2.2 Mapping Problem

Taking into account the impossibility of mapping entities based on its URI, another process would have to be created for us to conclude similarity based on the entities' properties. The Ontology Integration Process [2] is one process made for the purpose of merging similar information on the Web. In this process there appear two fundamental concepts: Ontology Mapping and Ontology Merging.

Ontology Mapping aims to build a new ontology by finding common concepts between different ontologies whereas Ontology Merging goal is to build a new ontology by merging several ontologies into a unique one, thus creating a generalized ontology about a certain topic. This process is then responsible for, firstly, mapping similar ontologies and then merging the information contained in each one of them. However, in our work we decided not to use ontologies, so this process had to be implemented in the scope of entities (objects references). Moreover, it had to be responsible for identifying similar entities and merging their information, thus creating a new representation of a real product, that was a mixture of several representations (entities).

1.2.3 Duplicated Data Problem

While identifying which different entities refer to the same real objects, we became aware that in the scope of products, duplicated data exists. To better explain this issue we will give an example.

Imagine that a user is searching for a specific smartphone, for example the iPhone 6. In fact there will be many data sources with information about this product but in each one of them we could find variations of it, i.e., the iPhone 6 16GB, the iPhone 6 32GB, etc. Obviously, from the perspective of bar code identifiers these products are not the same and consequently they have to be described with different codes, but from a user search perspective the only thing the user wants to know are the product characteristics and probably the variations it has. As a matter of fact, both iPhones have the exactly same properties except their storage capacity and the associated price. Therefore these two entries, from one of the available data sources, could be seen, or not, as the same real object depending on the user's perspective.

1.3 Solution Overview

First, the information is filtered based on rules that specify which properties from a data source need to have the same values, in order to be able to infer the notion of duplicated information. After that, we have to identify entries from separate data sources that refer to the same real world object and merge their information. This implies the development of a process identical to the Ontology Integration Process, responsible for evaluating which entities (as RDF models) are equivalent based on its properties and create a single entity (single model) that refers to a specific product, concatenating all the information extracted from the mapped entities. Figure 1.3 shows the overall goal of this work.

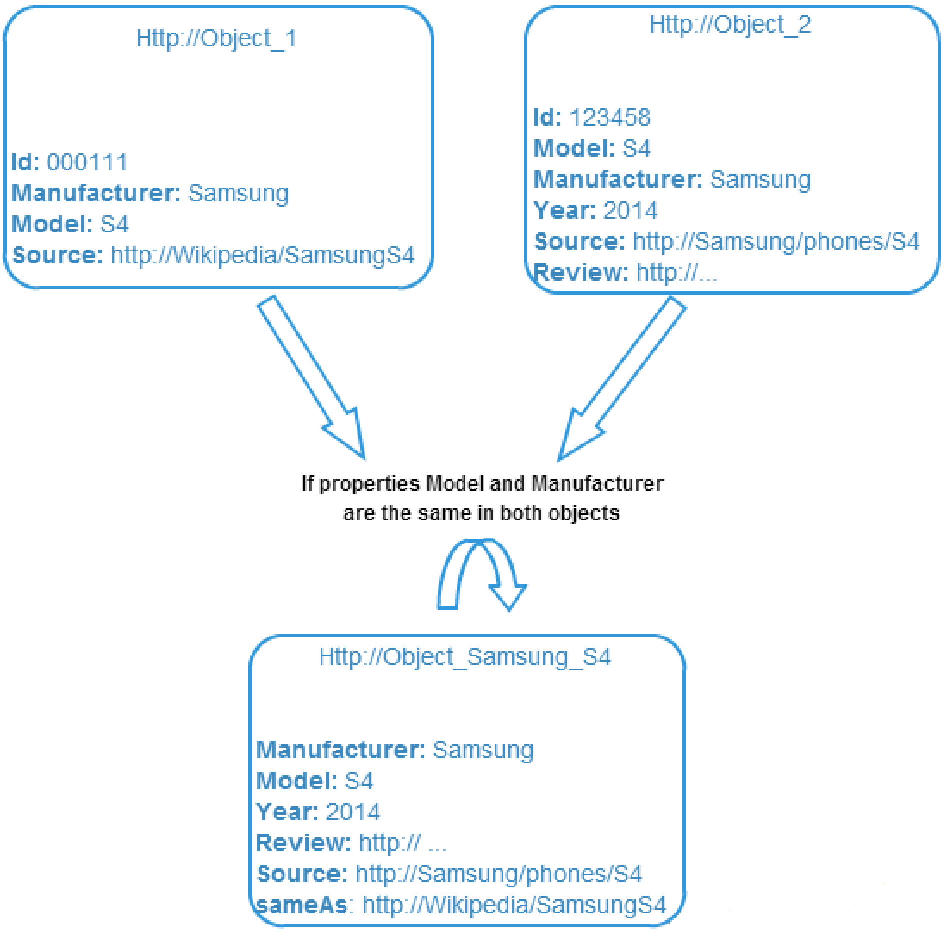


Figure 1.3: Linkage of information about the same product from different sources

1.4 Use Case Scenarios

1.4.1 Pharmaceutical Products

Medicines are well known products in all developed societies in the world, i.e., when someone refers to an active substance, it is clear to people which object is being referenced, like Ibuprofen. However, each country has its own drug regulator entity that defines which medicines can be commercialized in that same country and under which brands. Consequently, sometimes it is necessary to merge information between data available in each country and data available on the Web of Data to enrich information. An example of this kind of data available on the Web is the Drug Bank, a dataset that is part of the Linked Open Data (LOD) project and that contains loads of specific information about drugs [3].

1.4.2 Cinematographic Works

After the first scenario was fully developed we wanted to prove that the developed application was flexible enough to be used in other domains. Besides that, we wanted to prove that this tool could deal with information from distinct data sources, i.e., that the base of the information was not the same entity (as it happened in the previous case). To address the flexibility domain issue, we decided to take advantage of structured data available on the web, via the Linked Open Data community (Section 2.5 describes this concept in more detail).

After a research for available repositories, we decided to cross information between two well known repositories: “LinkedMDB” and “DBpedia”. The Linked Movie Data Base (LinkedMDB) project provides a high quality source of RDF data about movies [4], while the DBpedia focuses on the task of converting Wikipedia content into structured knowledge, such that Semantic Web techniques can be employed against it (asking sophisticated queries to Wikipedia, linking it to other datasets on the Web, or creating new applications) [5].

Overall, what we would like to offer to the user is an unified answer to all the possible questions s/he can has about a movie.

1.5 Contributions

This work had the aim of gathering information about products over the Web and existent repositories in the LOD community, filtering that information to remove duplicate entries, standardize it as RDF models and finally mapping and merging it to be presented to the user. Therefore, the main goal of this work was to solve the identity problem in the Semantic Web scope, i.e., identify which information was referring to the same product even though the identifiers do not coincide with each other. For that reason we developed a system, available in a public repository [1], where we were able to link information from different sources, join it into one model and offer it to the user.

[1] https://github.com/inesc-id/AdvancedSearchApp

The system proved to be useful and friendly in a end-user perspective, and domain independent. Besides that, for a know set of information, the system has shown the ability to find and manage duplicate information with high levels of precision, depending on the filtering rules applied, and also the ability to merge information from different data sources.

Therefore as a second goal we hoped that the reduction of the users workload could be seen as an advantage, despite of not being able to return the information as fast as the actual search engines. A goal that we have achieved, taking into account the feedback given to us by users who have tested the system and shown in Chapter 5.

1.5.1 Document Structure

The remainder of the paper is organized as follows. Section 2 explains with more detail the concepts of Semantic Web, Linked Data and Ontologies; In Section 3 the related work is presented; The architecture design and used technologies are described in Section 4; The evaluation methods and results are discussed in Section 5 and Section 6 concludes the dissertation.

Chapter 2

Background

Many concepts and languages have been created with the aim of structuring data on the Web: Semantic Web, Linked Data, RDF, SPARQL or OWL.

In this chapter the Semantic Web technologies and the most relevant concepts for this work will be presented in more detail below.

2.1 The Semantic Web

One of the basic slogans of the Web is that “Anyone can say Anything about Any topic” (AAA) and while it brought a richness of information, it also brought some problems in the scope of the information' interpretation. In other words, two crucial questions emerged:

1. How to know if different data sources are representing the same real object?
2. What value should be shown to the user if different values exist for the same property?

The Semantic Web [6] [7] came to answer these questions once it refers to World Wide Web Consortium (W3C) vision of the Web of linked data and aims to convert the current web, with unstructured documents, into a web of data to provide an environment where applications can query the data and draw inferences. Therefore, Semantic Web supports a distributed web at the data level rather than at the presentation level, like in the web as we know (HTML, CSS, etc.) and uses a standard known as Resource Description Framework (RDF), intended to be a simple and universal manner of representing anything [8]. Several companies, like NASA, Facebook, Google and CIA, are using this new approach to structure information on the Web[1].

In the history of artificial intelligence it is shown that knowledge is critical for intelligent systems and in many cases, better knowledge can be more important for solving a task than better algorithms. To have truly intelligent systems, knowledge needs to be captured, processed, reused, and communicated.

To identify the web resources uniquely, the Semantic Web uses URIs to provide a global identification for a resource that is common across the web. Most of the times, the resources are identified with URLs

[1] http://www.webnodes.com/who-uses-semantic-tech-today

(Uniform Resource Locator) that are a special case of a URI - besides identifying the resource, they also identify the resource's location. Therefore, instead of having pointers from a web page to another, there are pointers from one data item to another that consist in global references given by URIs. Another aspect of this concept is that every pair of entities (RDF subjects) that present the same URI are seen as the same object once RDF model is able to merge objects automatically whenever they have this characteristic in common (equal subject values). For that reason, every time two datasets want to reference the same real object, they could agree on establishing the same URI for it.

On the Semantic Web, the concepts and relationships (terms) used to describe and represent an area of concern, are defined as vocabularies. Vocabularies are used to classify the terms that may be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. Consequently, vocabularies have the role of helping data integration when, for example, ambiguities may exist on the terms used in different datasets, or when a bit of extra knowledge may lead to the discovery of new relationships.

Vocabularies can be also referred as ”ontologies”, where ontology is seen as an explicit specification of conceptualization. They are able to capture the structure of the domain, i.e. conceptualization, which includes the model of the domain with possible restrictions. For example, ontologies are applied in the field of health care by medical professionals to represent knowledge about symptoms, diseases, and treatments. Pharmaceutical companies, in the other hand, use ontologies to represent information about drugs, dosages, and allergies. Combining this knowledge from the medical and pharmaceutical communities with patient data enables a whole range of intelligent applications such as: decision support tools that search for possible treatments, or systems that monitor drug efficacy and possible side effects.

Many ontologies were created throughout the years, according to the ”semantiweb.org”[2], and some of them are shown in Table 2.1.

Ontology Name	Ontology Language	Revised Date
GoodRelations	OWL DL	2011
Music Ontology	OWL DL	2010
Dublin Core	RDF	2006
FOAF	OWL DL	2005
BIO	RDF	2004
VCard RDF	RDF	2001

Table 2.1: Existing Ontologies on the Semantic Web

2.2 Resource Description Framework

As previously said, the Resource Description Framework [8] is a standard model for data on the Web that facilitates data merging between schemas that can differ from one another. It is both simple and universal, and it can represent data as triples (subject-predicate-object) which can be uniquely identified

[2] http://semanticweb.org/wiki/Ontology.html

by URIs, extending the linking structure of the Web. For processing, RDF can be represented as a graph thereby having the capability of linking its nodes with relative ease.

An RDF Model is represented as a set of statements, which can be shown as a directed graph, where each statement is composed by three parts:

1. Subject - Resource from which the arc leaves;
2. Predicate - Property that labels the arc;
3. Object - Resource or literal pointed by the arc.

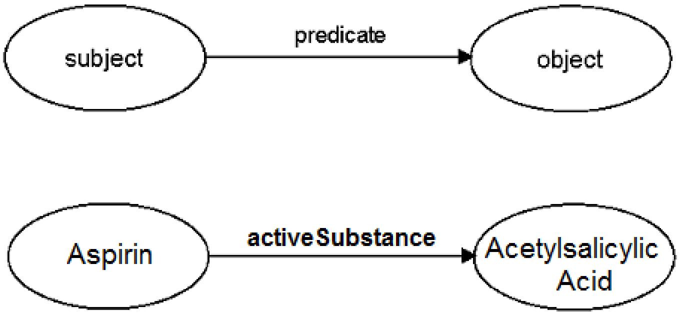


Figure 2.1: RDF graph representation and pratical example

Each subject can assume an URI or an unnamed node, known as blank node, as value. This way, a subject is always seen as a RDF Resource, which can has Properties that represent a relation between two components (subject and object). In its turn, an Object can be either a Resource or a Literal, being this last an absolute value (integer, string, boolean, etc.) [9].

Throughout the years many companies started to see the use of Semantic Web technology, such as RDF, as a benefit. Two examples are BBC[3] and Volkswagen[4]. Many are the reasons that lead companies to have this interest in the RDF technology, therefore we will present some of them [10].

Basic instance data is simply represented as attribute-value pars where the subject represents the instance itself, the predicate is the attribute and the object is the value. Such instances are known as the ABox.

RDF triples can be applied to all types of data: structured (standard Data Base (DB)), semi-structured (html) and unstructured (text). By defining new types and predicates, more expressive RDF vocabularies could be created which make this technology a language for data federation and interoperability across dispare datasets. Therefore, it is easy to aggregate new datasets or new information due to the no need of format or serialization.

To enrich basic vocabularies Resource Description Framework Schema (RDFS) was introduced and it was responsible for bringing new concepts as ”class” and ”subClass” for subjects, and ”domain” and ”range” for properties. The next layer was Ontology Web Language (OWL) which brought expressiveness to describe the relationships between entities - known as TBox.

[3] https://www.w3.org/2001/sw/sweo/public/UseCases/BBC/
[4] https://www.w3.org/2001/sw/sweo/public/UseCases/Volkswagen/

In the conventional relational databases, the information is represented as a table of rows and columns with keys to other tables, but as soon as that representation changes, those connection can be lost. RDF does not show that kind of limitations since it represents both instance data and the schema that describe them. Moreover, as new data and new relationships are discovered or created, they can be added to the existing model without any change or update to the prior schema. This adaptability characteristic made RDF to be seen as a data-driven design.

Besides all these advantages, RDF could also be seen as a graph database which from a computational perspective could mean scalability. Having this type of design could also offer new search paradigms.

To conclude, RDF is a framework for modeling all forms of data, describing that data as vocabularies, and interoperating it through shared conceptualizations and schemas.

2.3 SPARQL Protocol And RDF Query Language

For users to be able to query information in RDF format, W3C consortium developed the SPARQL Protocol And RDF Query Language (SPARQL) that is able to query and manipulate this kind of data stores, called triple stores, where the information is stored in a RDF format [11]. SPARQL has the particularity of being able to query a dataset without knowing anything in advanced about the contained data [10].

SPARQL consists of two parts: query language and protocol. The query part's purposed straightforward: SQL is used to query relational data, XQuery is used to query XML data and SPARQL is used to query RDF data. Despite the similarity with Structured Query Language (SQL) and being able to access relational data as well, SPARQL differs in that it was designed to operate over disconnected sources over a network in addition to a local database. In particular, the SPARQL protocol allows transmitting SPARQL queries and results between a client and a SPARQL engine via Hypertext Transfer Protocol (HTTP), which can be used to query live and public SPARQL endpoints.

SPARQL was designed to be easily learned by SQL users. A SPARQL SELECT query is composed by two parts: a set of question words, and a question pattern. The keyword WHERE indicates the selection pattern, written in braces, which is a pattern that is matched against the data graph. The action of the query engine is to find all matches for that pattern in the data, and to return all the values that the question word matched.

In order to exchange the results in machine-readable form, SPARQL supports four common exchange formats, namely the Extensible Markup Language (XML), the JavaScript Object Notation (JSON), Comma Separated Values (CSV) and Tab Separated Values (TSV).

Next, in this section, and in Table 2.2 some differences between SQL and SPARQL are presented.

SPARQL reuses some key words familiar to SQL users: SELECT, FROM, WHERE, UNION, GROUP BY, HAVING and most aggregate function names.

SQL uses the token NULL to indicate that data is not available or not applicable and this approach

SQL	SPARQL

SELECT <attribute list>
FROM <tables list >
WHERE <test expression>
	SELECT <variables list>
	WHERE { <graph pattern>}

SELECT EmployeeID, HireDate, City
FROM Employees
WHERE City='London'
	SELECT ?employeeID ?hireDate ?city
	WHERE {
	?s <http://company/empID>?employeeID
	?s <http://company/hireDate>?hireDate
	?s <http://company/city>?city
	}

Table 2.2: Difference between SQL and SPARQL queries

has some implications: SELECTs match table rows even if the selected attributes are NULL, however, JOIN rules will typically eliminate rows if there are no rows with corresponding values. To solve this problem OUTER JOIN operator could be used - it performs a regular (”inner”) join without eliminating solutions if the join constraints are not met. In comparison, missing data is simply not expressed in RDF. Also and consequentially, SPARQL graph patterns will not bind if there are missing attributes. Therefore SPARQL uses the key word OPTIONAL instead of LEFT OUTER JOIN, but with a similar effect.

Probably the main feature of SPARQL which will impress SQL users is the ability to federate queries across different repositories, since SQL has no standard system for query federation. RDF provides the integration of large databases as a trivial task, but instead of retrieving the data and merging it locally, SPARQL queries could be written to delegate portions of the query to remote query services.

The “P” in SPARQL stands for “protocol.” Since SPARQL was designed as a query language for the Web, it includes a protocol for publishing the results of a query to the web, which can deal with binary results (ASK queries - Yes/No), tabular results (SELECT queries), and, of course, triples (CONSTRUCT queries). This means that the output of a SPARQL query could be used on the Web as input for another query.

A server for the SPARQL protocol is called a SPARQL Endpoint — a service that accepts SPARQL queries, and returns results, according to the details of the protocol. That is why the reserved word SERVICE is used to dispatch a subquery to a specific endpoint.

Many SPARQL Endpoints are available today, providing information about a variety of subjects. Also, many of them are linked with other endpoints which has created the concepts of Linked Data and Linked Open Data, both described in the following sections.

2.4 Linked Data

The Linked Data concept is the activity that creates links between data, from different sources, which was not previously related, using the Web [12]. This is achieved by defining some specific attributes in the description of the objects which identifies that one resource is similar to another previously defined in a different dataset, on the Web.

The data is described in RDF with the purpose of be all in the same format and facilitate their linkage, once RDF make typed statements that link things in the world [12]. Moreover, with everything in the same format it is possible to make conversion and on-the-fly access to existing databases.

Linked Data is based on four main principles [13]:

1. Use URIs to denote things;
2. Use HTTP URIs so that these things can be referred to and looked up by people and user agents;
3. Provide useful information about the thing when its URI is dereferenced, using standards such as RDF and SPARQL;
4. Include links to other related things (using their URIs) when publishing data on the Web.

The AAA slogan along with the Semantic Web and Linked Data were able to create an increasingly referenced concept known by the Web of Data.

The Web of Data [14] is the concretion of the structuring and connection of data on the Web, hence it is build upon two main principles: Use the RDF data model to publish structured data on the Web and create links between data that belong to different data sources.

The Web of Data is thus achieved by the joint effort of Semantic Web and Linked Data, since the Semantic Web aims to structure the data, with the help of RDF models, and Linked Data aims to create links between data within different data sources.

Linked Data became an important issue since the Web began to be accessible for many people and along with it came the AAA slogan, meaning that ”Anyone can say Anything about Any topic”. Therefore, there are many databases that reference the same objects (products) in different ways that can be linked. One example of this reality is the Linked Open Data Project that will be described in further detail in the next section.

2.5 Linked Open Data Project

The Linked Open Data (LOD)[5] [6] is a community effort founded in January 2007 and supported by the W3C Semantic Web Education and Outreach Group.

Initially participants were researchers and developers in universities and small companies but with the growth of the project large organizations like BBC, Thomson Reuters and the Library of Congress, have become involved

In the beginning, LOD was a project with the aim of transforming unstructured data in the Web into structured data. This was achieved by publishing the data in RDF format and by using URIs and HTTP URIs to denote and reference objects, respectively. Nowadays, LOD is open to everyone, thus people who want to join simply have to publish data according to the Link Data principles and link it to data within some existing datasets. Because of its open nature, the LOD concept has been growing significantly

[5] http://linkeddata.org/

[6] https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

every year and that growth is illustrated in Figure 2.2 that compares the 2010 (Figure 4.8(a)) and 2011 (Figure 4.8(b)) visions of the LOD cloud.

In these figures we can see that the information is divided into seven main groups which refer:

- Media - television stations, radio stations, newspapers, etc.;
- Geographic - describe places geographically;
- Publications - organizations where people can publish their papers, journals, books, etc.;
- User-generated content - references blogs, forums, posts, chats, etc.;
- Government - services hold by the government like public schools and public transports, etc.;
- Cross-domain - datasets that cross information from several others. Ex.: DBPedia has structured information from Wikipedia;
- Life sciences - all information that has to do with science of life like genes, medicines, diseases, etc.

Every arrow shown in the figures represent a link between two different datasets which means that a described object in a certain dataset refer to the same real object as the one described in another dataset.

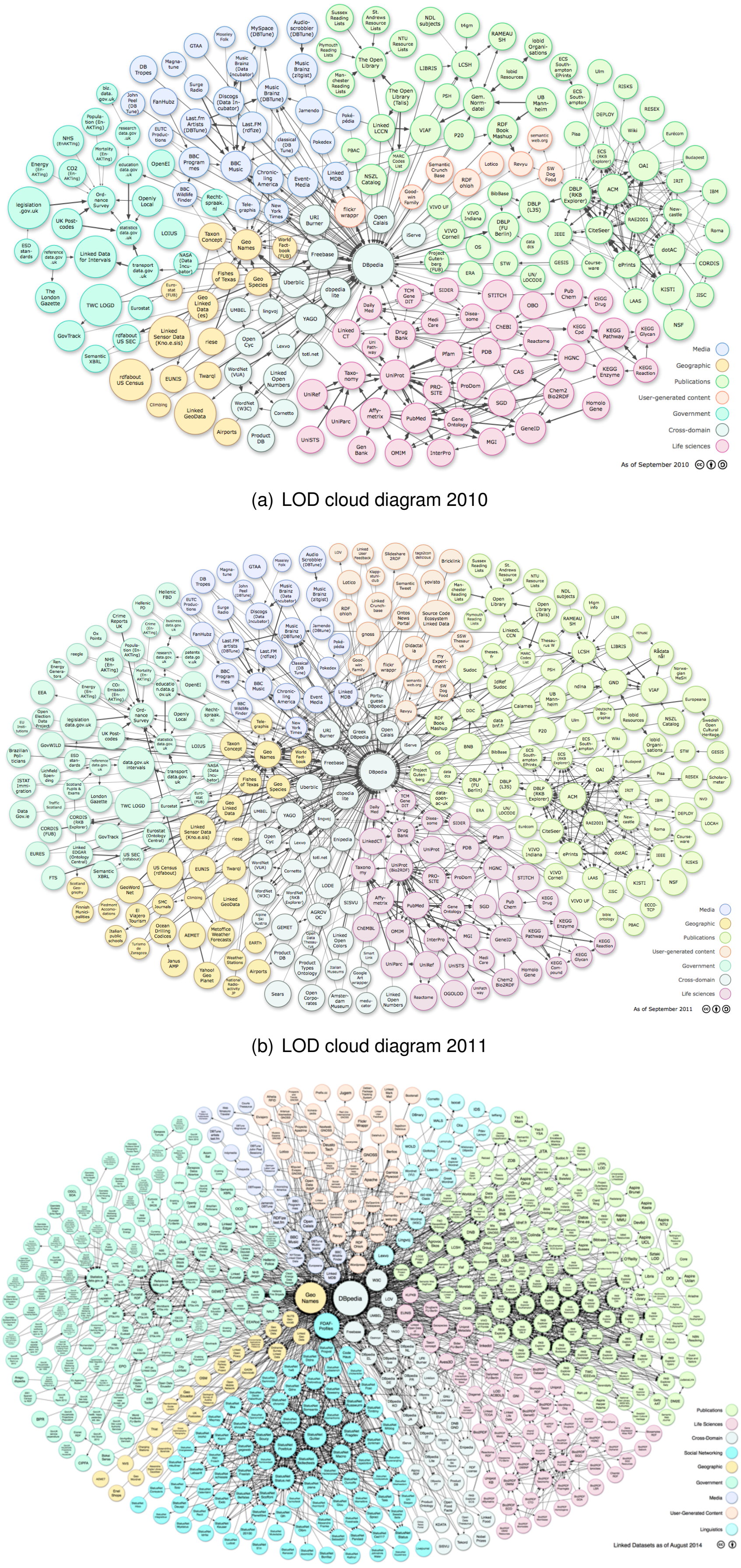
The growth of this project is visible in terms of number of links between data sources, for example, the number of links with DBpedia have increased, as well as the ones with GeoNames and DBLP. Besides that, data about cross-domain and user-generated content groups also increased in quantity.

In 2014 the LOD community has grown a lot which resulted in the appearance of two more groups named Social Networking and Linguistics, respectively. Figure 2.2(c) shows the most recent Linked Open Data cloud.

These images illustrates that data is being created in large quantities and without a centralized coordination. Therefore, in this context, it can be useful to merge information from different sources from an individual perspective, to provide it in adequate context to a specific user.

2.6 Summary

In this chapter we addressed the Semantic Web technologies such as RDF, SPARQL, Linked Data and LOD in further detail.



(a) LOD cloud diagram 2010

(b) LOD cloud diagram 2011

(c) LOD cloud diagram 2014

Figure 2.2: Cloud Diagram of Linked Open Data Project

Chapter 3

Related Work

With the growth of data in the Web, the problem of verifying if two entities from different sources are referring the same real world object has been recognized as an important issue by several other researchers. In fact, unique identifiers are absent in most of the existing data over the Web and therefore the need of identifying real world objects became a area of study.

We started by looking at a manual approach, where humans take a central role in the mapping by introducing their knowledge into the system. Then we explored the deterministic approaches, which take advantage of a specific ontology property to define which products are actually the same. After that, we present heuristic approaches known by defining which properties from different data sources are "good" to decide "similarity", and finally we addressed probabilistic approaches which are based on the creation of similarity probabilities for each pair of products.

3.1 Manual Approach

The first example of introducing human users in the system to obtain knowledge is Muse [15], a mapping design wizard that uses data examples to assist designers in understanding and refining a schema mapping towards the desired specification. To do so, Muse uses examples to differentiate between alternative mapping specifications and infer the desired mapping semantics based on the designer's actions, then automatically constructs a small number of, yes-or-no questions using small examples. The designer's answers to these questions will allow the system to infer the desired semantic grouping.

Another example is Clio [16] which uses reasoning about queries to create, manage and rank alternative mappings. However, the final choice of mappings must necessarily be made by a user who understands the semantics of the target application. By examining and manipulating carefully chosen data examples, the user decides what source data should be combined and transformed, and where in the target this data should be placed.

Belhajjame et al. [17] developed a system where users comment on results to queries evaluated using mappings previously made and applied to the data. In their simplest form, the schema matchings derived are binary relationships, each of which connects an element of a schema, e.g., a relational table in a source schema, to an element that is predicted to be semantically equivalent in another schema. The feedback required from users provides information about the usefulness of the results obtained by evaluating queries that use the generated mappings, i.e., given a certain tuple it was expected or not in the answer, or it was expected but was not retrieved.

In Cao et al. [18] the authors propose and formalize four kinds of feedback (SHOULD, SHOULD-NOT, LIKE, and DISLIKE) to filter and reorder results to a path query. Initially, the user poses a query and then the path results that match that initial query are retrieved. The user marks then parts of the presented results as preferred or not preferred and may also highlight some other parts as required or to be eliminated. This way, users give feedback for ordering paths based on preference criteria and for eliminating irrelevant paths from consideration.

In the work of McCann et al. [19] the key idea is to enlist the multitude of community members (i.e., users) to help the builders (small set of volunteers) match schemas. During the matching process the system asks users simple questions, then learn from the answers to improve matching accuracy, thereby reducing the builders' matching workload. Builders examine a list of candidate matches, ordered by ranking, to find the correct match, while users can answer ”yes”, ”no”, ”don't know”, or ”postpone” according to their beliefs. To deal with unpredictability the system classifies users as trusted or untrusted, based on their answers to a set of evaluation questions (with known answers) and ignores untrusted ones.

Doan et al. [20] create Cimple, a project that aims to develop a software platform that can be rapidly deployed and customized to manage data-rich online communities. To its functionality, a community expert has to provide a set of relevant data sources, domain knowledge about entities and relationships of interest, and possibly hints on extracting relevant mentions from the listed data sources, so the system can crawl the web looking for entities' relations. After transforming the raw data into a semantic entity-relation (ER) data graph a mass collaboration is employed to evolve and maintain the inferred graph while learning from users' interactions.

However, and according to [21] there are two main problems with the approaches based on user feedback: feedback inconsistency and validity.

Inconsistency - different users may have different requirements. Moreover, the requirements of the same user may change over time. The basic idea is that if two users have different requirements, then this difference may give rise to inconsistency between the feedback instances they supply. Similarly, if the requirements of a given user changes over time, then this change may give rise to inconsistency between feedback instances supplied at different points in time.

Validity – the user feedback is classified into valid, invalid or unknown, depending on the conflicts that arise trough the process or, in the last case, due to the absence of the entities inside the data sources.

Table 3.1 summarizes the manual approaches described before in terms of objects on which feedback is given by users.

Many other approaches to solve the mappings and data heterogeneity problems were used.

Work	Objects on which feedback is given
Muse[15]	An instance of a given schema and the instance obtained by transforming the source schema into target
Clio[16]	A view of the mapping result
Feedback-based annotation, selection and refinement[17]	A result tuple or an attribute and respective value
Feedback-driven result ranking and query refinement[18]	A candidate query or a pair of candidate queries
Matching schemas in online communities[19]	A list of candidate matches or a view of the matches results
Cimple [20]	A semantic entity-relation data graph create before applying mappings

Table 3.1: Manual Mapping Approaches

In the next sections some of those approaches, used to map resources in some given scope and decide if there is any pair of resources that can be merged into a single one, will be presented.

3.2 Deterministic Approach

In deterministic approaches the outcome is precisely determined through known relationships among different sources, without any room for random variation. In such models, a given input will always produce the same output. In this particular case, we have the certainty that two entities are the same when, for example in an ontology, we have the attribute “owl:sameAs” referencing a object in other dataset.

One of the done researches belongs to Raimond et al. [22] that developed an automatic interlinking for music databases. For the purpose, they implemented two different approaches: the first one was able to link data from two overlapping web datasets and the second was able to link data from the end-user with data from a public dataset. In both of them they manage to find similarities in pairs of resources, from the different sources, by comparing the literals attached to them. Whenever they find a similarity value that was higher than all the other values, they derived statements based on ontologies like “owl:sameAs”. Therefore, they were able to traverse different datasets just with the created links.

Despite the obtained results, their approach does not work for large datasets because that would mean thousands of comparisons. Besides that, for datasets without meaningful similarities or based on different ontologies, the algorithm will not be able to link the resources correctly.

The Ramezani et al. [23] work has led to another approach [24] that aimed to find association rules in Linked Data. For this approach, the authors collected the desired data by dealing with ontologies and traverse the datasets through predicates like “owl:sameAs”. After that, they made decisions to solve the duplicated data problem and placed the data in a central database with a unified ontology. Finally they called the SWApriori [23] algorithm and were able to create the association rules. This approach is able to recognize when two resources are the same by traversing datasets based on information stored in the ontology, placing the data together without any conflict and using the same data structure.

3.3 Heuristic Approach

Heuristic approaches aim to find an identity mapping which is not guaranteed to be optimal, but good enough for a given set of goals.

Heuristic methods are used to speed up the process of finding a satisfactory solution by facilitating the cognitive load of making a decision. For that purpose there are made rules that permit us to verify if two objects are referencing the same real world product, for example, if two given properties have the same values in both objects it can be assumed that they are referencing the same real product.

Some works based on this approach are described below.

Isele et al. [25] developed an algorithm called GenLink that learns linkage rules from a set of reference links using genetic programming. Therefore they get a list of pairs of properties with similar values that will be analyzed by tokenizing and lowercasing those values and creating a new property pair (p1, p2, measure), where measure is the distance measure according to how similar two tokens are, given a specific threshold. After that, a linkage rule is built consisting of a random aggregation and up to two comparisons which have selected random pairs, made in the previous step. The Matthews Correlation Coefficient (MCC) is then used to associate a value to each linkage rule and decide if it represents a true positive, true negative, false positive or false negative. Finally they use a aggregation crossover function that permits concatenation of rules to define objects structure.

Friedrich et al. [26] present a totally different approach for this problem. Instead of regular expressions or ontology classification, they analyze the HTML headers of the Web pages and the anchors that point to it. As it is known some identifiable HTML tags contain names of entities mixed with irrelevant information and to solve that problem they applied classification filters to extract the identifier of the instance. Therefore, tags (title tags and anchor tags) contain useful information about the product name they are referring to as strings, and for that reason they are considered as observation strings of a Web page. Then, they split the strings by tokens and, each occurrence of a token in an observation string is considered a positive token observation and the number of occurrences, associated to that token, is incremented by one. They use HAC (Hierarchical Agglomerative Classification) as a classification algorithm in order to group objects by proximity, once the goal is to put in different clusters specific Web information like “Specifications & Features” and specific product information, like “18mmto-55mm lens (silver)”. Once they achieve this goal, tokens with few observation are removed from each cluster and names are given based on unifying tokens of the observation strings. In the end they have named groups with corresponding product describing Web pages.

The SWApriori [23] algorithm mine association rules from Semantic Web data but the first steps of it could be used for this problem of linking data from different sources. This approach is divided in 5 steps but the first two are particular interesting. In the first step data is collected from different sources and maintained in memory as triple format (RDF), after that the algorithm generate 2-large itemsets based on the objects frequency. With that information, the algorithm is able to create larger item sets and find association rules for each itemset. For the linking data problem, higher object frequencies could mean that different resources are representing the same object.

3.4 Probabilistic Approach

Probabilistic approaches enable variation and uncertainty to be quantified, mainly by using distributions instead of fixed values. A distribution describes the range of possible values for some property, and shows which values within the range are most likely. In the scope of link products, this approach is used to define classes of products and determine how probable a product belong to a certain class, according to the similarity between its properties and the ones defined for the class.

Kopcke et al. [27] use this approach to determine if two products' references are identifying the same real world product, thus they do the matching of product entities based on machine learning and probabilistic classifiers.

First they construct a base of knowledge that is divided into several categories (classes) based on the products' properties (eg. title, price, description). Whenever it is needed to insert a new product information to the knowledge base, Naive Bayes Classifier was used to calculate the probability of the new product p belong to a certain category c P(cjp). This probability was achieved by testing how the properties' values of p were similar to the ones defined for each category.

Finally, when all products are categorized, they use a heuristic approach to decide if two products refer to the same real object. To this end, for each category and for each pair of objects they verify if the manufacturer is the same and if it is so, the objects are merged.

3.5 Summary

The Table 3.2 summarizes the approaches above described.

Our approach in this work presents a particular way of creating mappings between two dissimilar datasets. As referred before we firstly provide our system to expert users (Data Curators), which have the capacity to extract information from a schema file and suggest mappings that could make sense to other users, and then we make the system available to common users which are able to choose the mappings that suits them best. This way our work does not present any of the problems described in [21] since the users choose the mappings whenever they turn on the system, they are free to choose their relevant requirements at each moment. Besides that, the validity of the mappings does not apply in this case due to the dynamic search on the different data sources at each run of the system and due to the user's choices not being saved for later researches in the system.

Overall, our work combines both manual and deterministic approaches since it is based on the users knowledge to create mapping rules, but also because it uses the property "owl:sameAs" to map products, and in case of existence of that property before the mapping, it is used to determine which

Approach	**Level of certainty**	**Description**
Manual	High	Users are responsible for confirm the accuracy of existing mapping rules based on its returned results.
Deterministic	High	Some properties provide the mapping result. In this case, the ontology property "*owl:sameAs*"
Heuristic	Medium-high	Rules are created to define the results. In this case it is decided which properties from different RDF models have to be compared to declare that the information is about the same real product.
Probabilistic	Medium-low	The rules created are based on the probability of similarity between RDF models. Classes are made to cluster similar products to be merged afterwards.

Table 3.2: Different Mapping Approaches

different products are actually the same real world objects with certainty. However, going through those other presented approaches is something that is in our future plans.

Chapter 4

Solution

As a useful application of an identity mapping engine, we developed a linked data search application.

Although actual search engines are optimized for the quick answer, obtaining a complete and unified information about a topic could be very useful to the users. The proposed solution will not be able to give a quick answer but will produce a complete and unified result. Therefore, this search approach aims to reduce the user workload by identifying different data sources with information about the same topic of interest, linking that information and retrieving it in a single interface step without the need to visit more than one site.

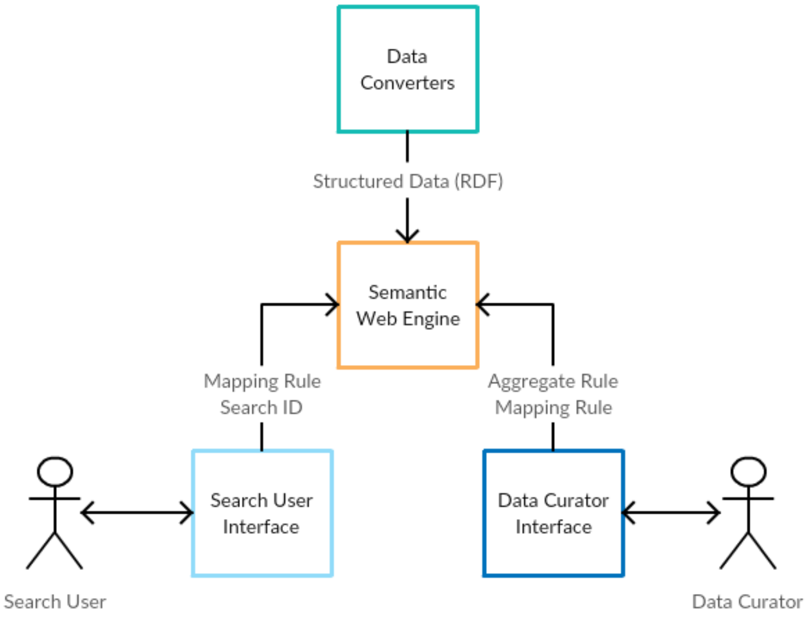


Figure 4.1: Solution Overview

4.1 Architecture

Figure 4.1 shows an overview of the implemented solution's architecture. This solution is divided into four distinct modules:

- Data Converters (green box);
- Semantic Web Engine (orange box);
- Search User Interface (light blue box);
- Data Curator Interface (dark blue box).

Throughout this section each module will be explained in further detail.

The **Data Converters** (green box) in this architecture is responsible for extracting the information and give it a standard structure to allow its manipulation. For that purpose, it is composed by as many Converter Processes as Data Sources. That is because each Data Source has its own data representation format. i.e., the properties used to describe the products are not the same for every source. Therefore, it has to exist one Converter Process per Data Source, which has the knowledge about the used properties, and consequently, the ability of extracting the data and convert it to RDF format. When all the data is standardized, this engine sends the RDF triples to the Semantic Web Engine. Figure 4.2 illustrates this part of the solution architecture.

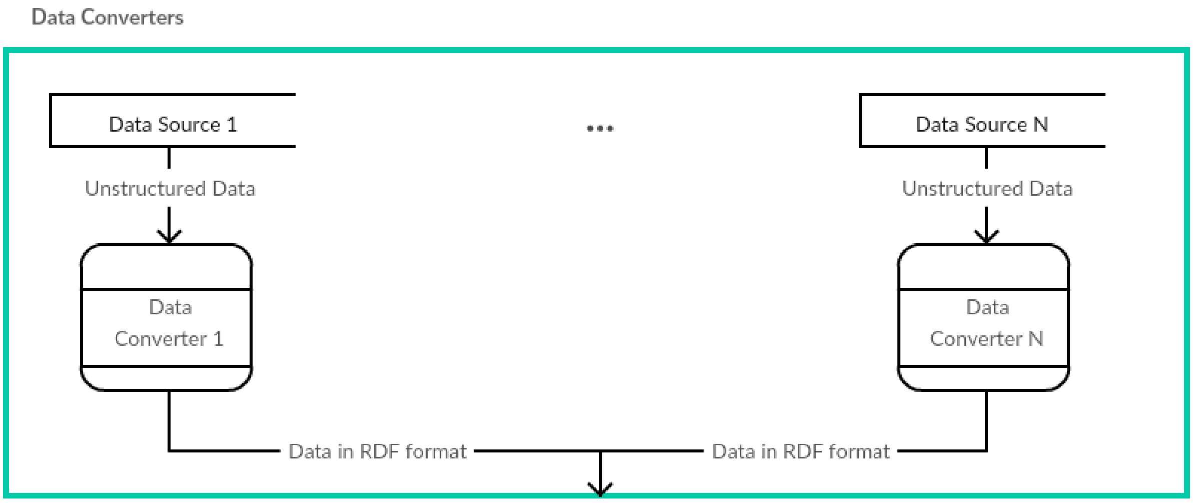


Figure 4.2: Data Converter Engine

The **Semantic Web Engine** (orange box), shown in detail in Figure 4.3, is composed by three different repositories: a RDF triple store, a repository to keep aggregation rules and another one to keep Mapping rules, and by three engines: SPARQL, Aggregation and Mapping.

This module in particular could receive a considerable range of inputs, which are:

- List of properties to create new aggregation or mapping rules;
- A particular aggregation or mapping rule to be applied to the repository;

- A product/object identifier to perform a search in the RDF dataset.

Besides that, it also receives RDF triples created by the Data Converter and stores them in the RDF Repository.

Concerning the three existing engines, each one plays a different role. The Aggregation Engine is responsible for creating new aggregation rules and store them in the Aggregation repository. Also, it is responsible for getting all the stored information about a certain rule, chosen by the user, and creating a query to be then applied to the RDF repository by the SPARQL Engine. The Mapping Engine is very similar to the previous one, the only difference is that it creates and manipulates mapping rules rather than aggregation ones. Finally, the SPARQL Engine is responsible for make all the queries to the Repository and return the respective results.

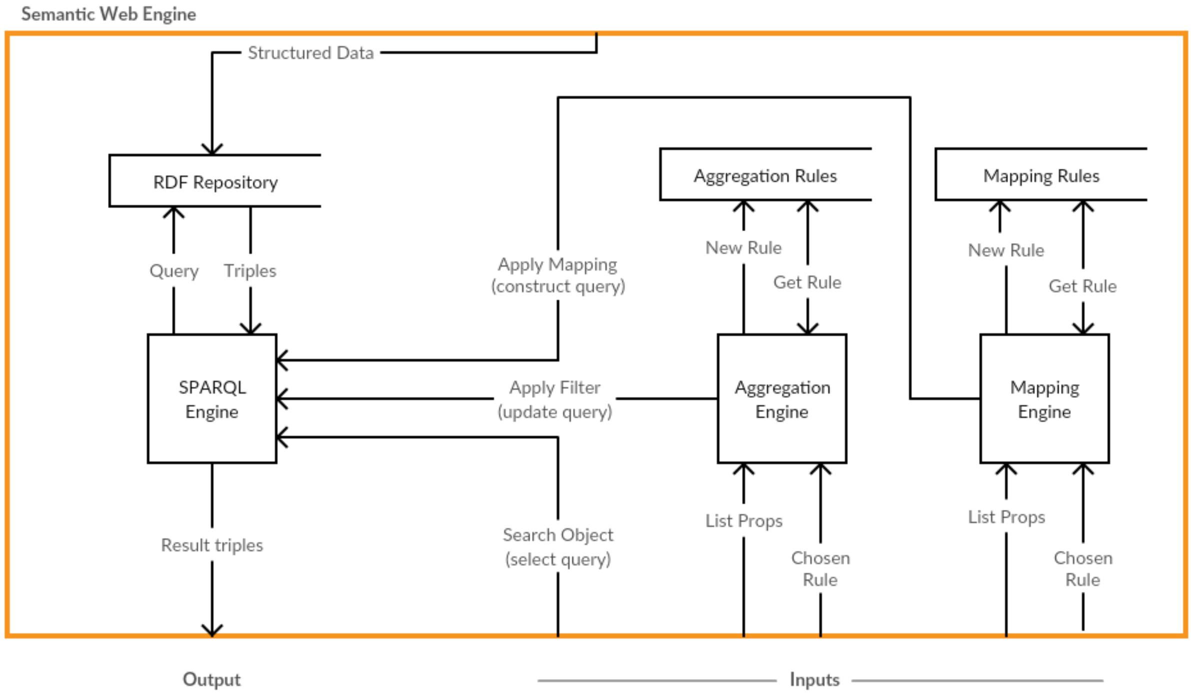


Figure 4.3: Semantic Web Engine

The **Search User Interface** (light blue box) represents the interaction between the system and the user who is performing a search. The Search User is responsible for choosing a mapping rule, so the information can be merged, and for search on the system by giving as input an identifier of the product s/he wants to search for. After receiving the inputs given by the user, the User Interface is responsible for communicate with the Semantic Web Engine and return to the user the results of the requested information. All these interactions are shown in Figure 4.4.

The **Data Curator Interface** (dark blue box), with higher detail in Figure 4.5, is the novelty brought by this project. This engine is composed by a Data Curator and a system for her/him to create aggregation and mapping rules. Therefore, the Data Curator is responsible for two main tasks:

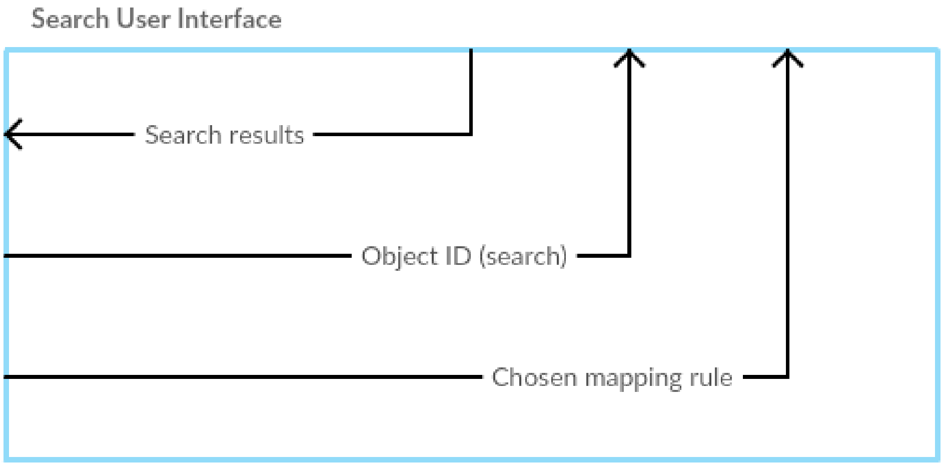


Figure 4.4: Search User Engine

1. Give a list of properties that will be used to create a new aggregation rule and consequently, will remove duplicated entities;
2. Give a list of properties used to create a new mapping property which will define if two entities from different sources are referring the same real world object;

Each list of properties are going to be given as an input for both Aggregation and Mapping engines.

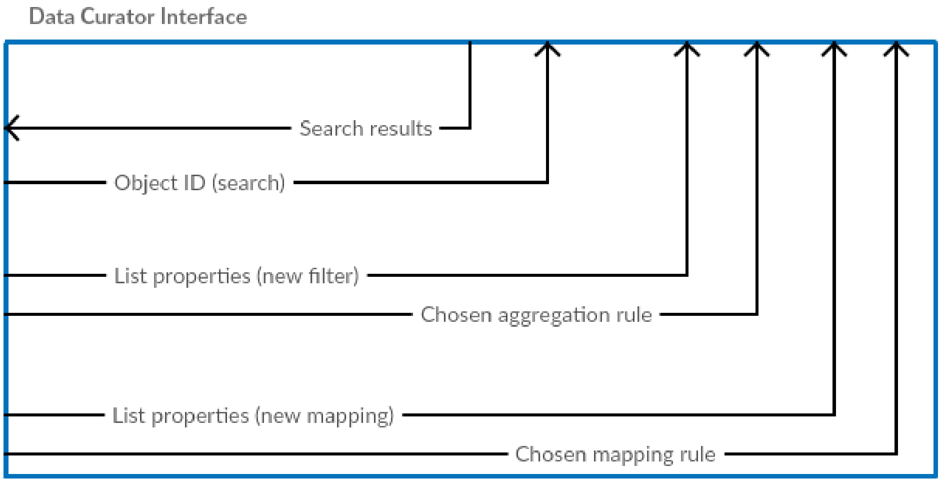


Figure 4.5: Data Curator Engine

Overview of Solution Processing

In the presented solution, the gathering of information is done before before any of the users interact with the system. That work was made a priori by choosing two sources of information and converting them into a RDF format.

There is an interface for each type of user, and the data curator should primarily be the first to run the application. This is because he is the responsible for creating both the filtering and the mapping rules, i.e., it is required a expert user to make the decision of which properties tells us that an entity has duplicates and which entities represent the same real world object.

The Data Curator, despite being an expert user and being responsible for the creation of such rules, he has also the possibility of querying the repository to see how the data changed and infer whether the created rules are working properly.

After the Data Curator step, the Search User can run the application and will be informed about how the filtering of the data was made by the Curator, since he is the only one with the ability of decide which two entities represent a integral copy of information. From that point on is the Search User who is going to make the decisions. This second user can now choose which mapping rule is going to be applied to the data and perform searches in the updated repository.

Relative to the Semantic Web Engine, depending on the input, a specific sub-engine (SPARQL, Aggregation or Mapping) will be called. In the case of creation or appliance of an aggregation rule, it will be the Aggregation Engine to deal with the input. Similarly, if the input is a list of mapping properties or a mapping rule, the Mapping Engine will be called. Whenever a query has to be made to the repository, the SPARQL Engine is activated, even when the input comes from inside the Semantic Web Engine. This engine is then responsible for returning the query's results back to the entity that has requested them.

Inputs from the users are passed to higher layers (lower layers of abstraction) until they arrive to the engine capable of treating them. In the end, users are presented with a complete information, listed in a friendly and easy understandable format (tables).

4.2 Technologies

A few years ago the World Wide Web was a Web of linked documents, full of content intended to be displayed for humans. The information contained in those documents (web pages, videos, images, etc.), was opaque to computers and so hard to be treated in any automated way.

The Semantic Web was introduced to the world in a 2001 article in Scientific American [28] where a new vision of the web appeared, i.e., software agents explore the Web, discovering and consuming structured content and semantic relationships that allow them to automate many aspects of our lives.

The reason why we chose the Semantic Web as technology to this work was the fact that it is recent and a reality. In fact there are hundreds of data sets across the Web linked together at the data element level by millions of relationships. Consequently, we chose the Resource Description Framework (RDF) as the technology to describe the data in the system, according to the W3C consortium practices and due to its main advantage of combining metadata with data in the same format.

Comparing the semantic web approach with the dominant approach: the relational model.

As it is known relational data is stored in tables, with particular columns, from which the data meaning derives from. Details about each table and its columns (name, definition, acceptable values, etc.), are kept and stored separately from the contained data - known as catalog - which difficults the access to it. In sum, this means that software that accesses relational data needs to have the meaning of the data hard-coded into it in some way to be able to draw inferences. In contrast, the meaning of RDF data is part of the data itself, in other words, wherever data is its details (i.e. metadata) are always immediately available. There is no need of explicit knowledge of the meaning of the data, nor the need for completely separate mechanisms to interrogate its meaning, i.e. RDF data is self-describing.

As before, we continue to choose technologies recommended for the W3C and so, for querying the data kept in RDF format we chose the SPARQL Query Language. The main reasons for this option were based on the advantages brought by this language, previously described in Section 2.3.

Since we had the goal of linking information from disparate datasets, SPARQL proved to be the most obvious way to achieve it, since it can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware [11]. Moreover, SPARQL has the capability of querying required and optional graph patterns along with their conjunctions and disjunctions, also supports value testing and constraining queries and could create new RDF graphs based on the existing ones. Another advantage is that since the results of SPARQL queries can be results sets or RDF graphs, it is possible to create a chain of queries, where the output of a query is the input of the following query.

For our purpose of “Linked Data” (to link the information together), we chose the OWL language for having the purpose of publishing and sharing ontologies on the World Wide Web and for being developed as a vocabulary extension of RDF [29]. OWL is a language to extend the expressibility that RDF provides, once it was projected to be used over the World Wide Web and all its elements (classes, properties and individuals) are defined as RDF resources and identified by URIs. Besides that, this language has also the advantage to be appropriate for computers to process it and extract knowledge more easily.

The Linked Data concept will be used to link the data inside the application. Since the data are derived from different sources this approach will be used to describe which RDF subjects are representing the same real world object. Therefore, we are only using the OWL property ”owl:sameAs” to later be able to infer similarity relationships.

For the programming development the Apache Jena Framework[1] was the chosen one because its a free and open source framework designed for the development of Semantic Web and Linked Data applications.

For storing RDF data there are two available triple stores Virtuoso Quad Store and Jena TDB. Virtuoso Quad Store is scalable, high-performance and open-source triple store, however it requires separate installation of Virtuoso Universal Server and additional libraries to work with Jena RDF API. On the other hand, Jena TDB is native file based triple store of Jena, which is highly scalable and requires no extra tool other than Jena Framework.

[1] https://jena.apache.org

The Apache Jena is written in Java and was created by the Apache community. Moreover, this framework complies with all recommendations and technologies from W3C and offers:

- Java programming API;
- Parsers for RDF in multiple formats: XML, Turtle and N-triples;
- Rule-based inference engine for RDF Schema (RDFS) and OWL connections;
- SPARQL query language (complete implementation);
- SPARQL endpoint (named Fuseki);
- In-memory, SQL, or native tuple storage.

In order to crawl web pages for the extraction of information for further RDF formatting HtmlUnit[2] was used, a well known web browser tool, written in Java, with a large community to help solve some issues and with a easy way to include in a project (simply add some specific libraries to the class-path). HtmlUnit [3] models HTML documents and provides an API that allows the programmer to invoke pages, fill out forms, click links, etc. just like it is done in a common browser. It has JavaScript support and is able to work with quite complex AJAX libraries, simulating Chrome, Firefox or Internet Explorer depending on the configuration used. Typically it is used for testing purposes or to retrieve information from web sites. Summing up, this tool allows the manipulation of websites from other Java code and emulates part of browser behavior. Besides that, it allows users to navigate through hypertext over HTTP and obtain web pages that include HTML, Ajax, JavaScript and cookies.

Finally, with the purpose of structuring the project we decided to use the Apache Maven[4] tool - a popular open source build tool for enterprise Java projects, designed to facilitate the build process. Maven uses a declarative approach, where the project structure and contents are described, rather then the task-based approach used in Ant or in traditional “make files”. Moreover, Maven is a project management tool which includes a project object model, a set of standards, a project lifecycle, a dependency management system, and logic for executing plugin goals at defined phases in a lifecycle [30]. This way, one of the main advantages in our case is the dependency management with all the technologies we are using due to the creation of project dependencies, extracted from the Maven repository at the building time.

4.3 Implementation

For the aim of linking product data a Java application was created to gather information from the Web, map it in a similarity concept view and search for specific data inside the created repository. Therefore, the developed application is divided into four main steps:

[2] http://htmlunit.sourceforge.net/

[3] according to Mike Bowler (its creator) is a ”GUI-Less browser for Java programs”

[4] https://maven.apache.org/

1. Gathering and formatting of information;
2. Filter information to remove duplicate values;
3. Map entities based on similarity;
4. Perform searches.

4.3.1 Gather information

The first step is performed in the beginning of the application launch, more precisely upon the creation of the Semantic Web Engine (SWE). Therefore, in the end of the SWE initialization the RDF Repository, contained in it, is already filled with the information extracted for the different datasets and formatted in RDF. In this way, firstly a Data Converter is created for each source of information and two main things are made.

The first thing to do is to extract the schema from the data source and it is made by the function "***setSchemaModel(Model model)***". In the scenario of Pharmaceutical Products this function is responsible for crawling the web page where the information is kept, and convert the table structure to a RDF schema that will then be added to the model given as argument in the function. The crawl of the web page and extraction of the table structure information is done through the HtmlUnit tool, i.e., a navigation in the web page is simulated and information is extracted as we are going through the table columns.

A distinct Data Converter is needed for each source due to the different way that information is presented, i.e., each source has its own tables with its own properties and most of the times, even when the area of activity is the same, the characteristics of a product are represented by properties with different names. For example, in our case Autoridade Nacional do Medicamento e Produtos de Saúde, I. P. (INFARMED) present to the users two distinct webpages where the columns relative to a given type of information have different names: in one of them the user find the property *"Substância Activa"* while in the other the same information has the property name *"Nome Genérico"*. Therefore, in this work we simulate a system with previous knowledge about the manner as the information is presented to the user, in other words, the system knows a priori the existing columns of the displayed table.

The Listing 4.1 shows briefly how a Data Converter extracts the information contained in a webpage and translates it into RDF triples. The reserved words RDF.type, RDF.Property, RDFS.Class and RDFS.domain have the aim of describing the schema of the repository.

```
public void setSchemaModel(Model model) throws Exception{
    String baseURI = "http://www.infarmed.pt/";
    List<Resource> properties = new ArrayList<Resource>();

    Resource medicine = model.createResource(baseURI + "Medicine");
    medicine.addProperty(RDF.type, RDFS.Class);

    WebClient webClient = new WebClient();
    HtmlPage page = webClient.getPage(
                        "http://www.infarmed.pt/genericos/pesquisamg/pesquisaMG.php");
    HtmlTable table = (HtmlTable) page.getElementById("mainResult");

    properties.add(
        model.createResource(baseURI + table.getCellAt(0, 2).asText().replace(' ', '_')));
    properties.add(
        model.createResource(baseURI + table.getCellAt(0, 1).asText().replace(' ', '_')));
    properties.add(
        model.createResource(baseURI + table.getCellAt(0, 3).asText().replace(' ', '_')));
    properties.add(
        model.createResource(baseURI + table.getCellAt(0, 4).asText().replace(' ', '_')));
    properties.add(
        model.createResource(baseURI + table.getCellAt(0, 5).asText().replace(' ', '_')));

    for(Resource p: properties){
        p.addProperty(RDF.type, RDF.Property);
        p.addProperty(RDFS.domain, medicine.getURI());
    }
}
```

Listing 4.1: Extract Web page schema

After this first extraction of schema information, the real data is extracted and allocated to each property as shown in the Listing 4.2.

```
public void infarmedModel(Model infar, String substancy, String name, String type,
            String dosage, String numUnits, int prescriptionCode, float price,
            String generic, String rcm, List<String> rcmInfo,
            String fi, List<String> fiInfo){

    this.drugID++;
    String baseURI = "http://www.infarmed.pt/";

    infar.setNsPrefix("", baseURI);

    Resource r = infar.createResource(baseURI + name + "_" + drugID);
    r.addProperty(RDF.type, baseURI + "Medicine");

    r.addProperty(infar.getProperty(baseURI + "Nome_do_Medicamento"), name);
    r.addProperty(infar.getProperty(baseURI + "Substancia_Activa"), substancy);
    r.addProperty(infar.getProperty(baseURI + "Forma_Farmaceutica"), type);
    r.addProperty(infar.getProperty(baseURI + "Dosagem"), dosage);
    r.addProperty(infar.getProperty(baseURI + "Tamanho_da_Embalagem"), numUnits);
    r.addProperty(infar.getProperty(baseURI + "CNPEM"), Integer.toString(prescriptionCode));
    r.addProperty(infar.getProperty(baseURI + "Preco_(PVP)"), Float.toString(price));
    r.addProperty(infar.getProperty(baseURI + "Generico"), generic);
}
```

Listing 4.2: Extract Webpage data

Product 1	
ID	<http://med/med1>
Name	Aspirin
Dosage	100mg
Number of units	20

Product 2	
ID	<http://med/med2>
Name	Aspirin
Dosage	100mg
Number of units	60

Table 4.1: Example of duplicated data

4.3.2 Filter information to remove duplicate values

This step consists in a data filtering with the purpose of removing duplicated information. This issue became one of great importance because duplicated information represents a negative influence on the performance evaluation of mapping rules. To better explain this problem we give a concrete example:

Imagine that, in the pharmaceutical field, the repository A contains two products "Aspirin 500mg 30und" and "Aspirin 500mg 60unid" and that repository B has only "Aspirin" with information about the drug's side effects. In this case, the number of pills inside the drug box does not matter for the linkage of the products from A and B, therefore what would happen would be a total of two mappings instead of one which would mean a worse mapping in terms of performance. Following the logic, if we have a repository A with 5 entries and a repository B with 10 entries, at most we can pair 5 entities.

Under this assumption we realized that the filtering of repositories was essential before initiating the mapping process. The next challenge was to answer the question "Has the user enough knowledge to decide how the filtering should be done?", more than that "Has the user the ability to understand the information given by the schema of a repository?". At that time we concluded that a common user could not understand the information that was being presented to him, and consequently a more experienced user would have to came to help him. Therefore, we decided to bring the concept of "Data Curator" thus dividing the interaction between user and the solution into two parts: first a interface for the expert user to interact - the Data Curator - and second a interface for a common user - the end-user.

Based on the need of expert knowledge to understand the extracted information, we have decided that only the Data Curator had the ability to create aggregation rules (used for removing duplicates) and also mapping rules (used for defining object similarity).

Regarding the Aggregation Rules, these rules aim to find duplicate information in a given repository but because the duplicated information lies in the same data source, it is expected that two entities that represent the same real object are presented with different identifiers. Therefore, these rules are created based on a set of properties that together can say that two entities are a case of duplicated information.

As shown in Table 4.1 the two products have the same characteristics differing only in the property "Number of units". In this case, the aggregation rule should be centered on finding instances with the same values for properties "Name" and "Dosage".

In Listing 4.3 a more compact view of the function "***createAggregationRule(...)***", responsible for the creation of aggregation rules, is shown. This function represents the interaction between the **Data Cu-**

rator Engine and the **Aggregation Engine**, inside SWE, with the input “List of properties” (as criteria) and the further creation of a new rule inside the **Aggregation Rules** repository.

```java
public void createAggregationRule(String source, String ruleName, String criteria){

    String baseURI = "http://" + source;
    Resource newRule;

    newRule = dbFilters.createResource(baseURI + "/" + ruleName);
    newRule.addProperty(RDF.type, baseURI + "/AggregationRule");
    newRule.addProperty(dbFilters.getProperty(baseURI + "/criteria"), criteria);

    dbFilters.commit();
}
```

Listing 4.3: Create Aggregation Rule

After this, the Data Curator has the opportunity of listing all the aggregation rules already created and seeing the consequent results in the RDF repository by choosing one rule per data source (with the possibility of choosing none of the existing rules). Whenever a rule is chosen the **Data Curator Engine** sends that information to the **Aggregation Engine**. inside the SWE. In its turn, it will get the list of properties associated to that given rule, and finally apply the rule to the main repository, with the help of **SPARQL Engine**.

In the code, the list of properties request is done by the function “***showAggregationCriteria(String rule)***” as shown in Listing 4.4, after that a SPARQL query is constructed for applying the rule to the main repository.

```java
public String showAggregationCriteria(String rule){
    Query query;
    QueryExecution qe;
    ResultSet results;
    String queryString, result;
    ByteArrayOutputStream go = new ByteArrayOutputStream();

    queryString = "SELECT ?Aggregation_Criteria\n"
                + "WHERE {"
                + "<" + rule + "> ?p ?Aggregation_Criteria ."
                + " ?p a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>}";

    query = QueryFactory.create(queryString);
    qe = QueryExecutionFactory.create(query, dbFilters);
    results = qe.execSelect();
    ResultSetFormatter.out(go, results, query);

    result = go.toString();
    qe.close();

    return result;
}
```

Listing 4.4: Asking for Aggregation Rule properties list

In Listing 4.5 an example of an aggregation rule query is shown, based on the previous example given by the Table 4.1. The type of triples the query is trying to gather are the ones that belong to the class “Product” (given by the “a” expression) and have individually the same values for the properties “Name” and “Weight”. The last line of the query guarantees that the products “ID” are different.

```
SELECT ?s ?s1
WHERE {
        ?s a <http://med/Product> .
        ?s1 a <http://med/Product> .
        ?s <http://med/Name> ?Name.
        ?s1 <http://med/Name> ?Name.
        ?s <http://med/Dosage> ?Dosage.
        ?s1 <http://med/Dosage> ?Dosage.
        FILTER (?s != ?s1)
}
```

Listing 4.5: Aggregation Query

With the triples return from the above query, the **Aggregation Engine** is responsible for creating another query, responsible for eliminating the duplicates. We chose to maintain the subject relative to the “?s” and eliminate the subject “?s1”. The created query uses the reserved word “DELETE” but in fact it represents an update to the repository, through the following line of code “***UpdateAction.parseExecute(delete, dbInitialModel)***”, where the “dbInitialModel” is the main RDF repository and the “delete” argument is the query shown below.

```
DELETE
WHERE { <s1> ?p ?o }";
```

Listing 4.6: Delete Query for removing duplicates

4.3.3 Map entities based on similarity

The next step focuses on creating mapping rules and again the lack of knowledge by the general user led us to assign this task back to the Data Curator.

As we said throughout this paper, there are many data sources with information about the same topic of interest. Given that information belongs to different sources, that data is structured in different ways, with different identifiers and properties. This happens because there is no defined standard for every activity area and also because it is in the hands of each company to decide which technologies will be used to describe the data. Therefore, the process of linking data through the web is highly complex, which is known as the Identity Problem.

Being aware of this great obstacle, we decided to give the Data Curator the task of find similarities in terms of the properties being used for describing entities in each data source. For this process to be automatic we would have to introduce Artificial Intelligence, where a knowledge base of synonyms would be essential to conclude “Similarity”. However, and since it was not the aim of this thesis we introduced a Human Data Curator, with logical reasoning.

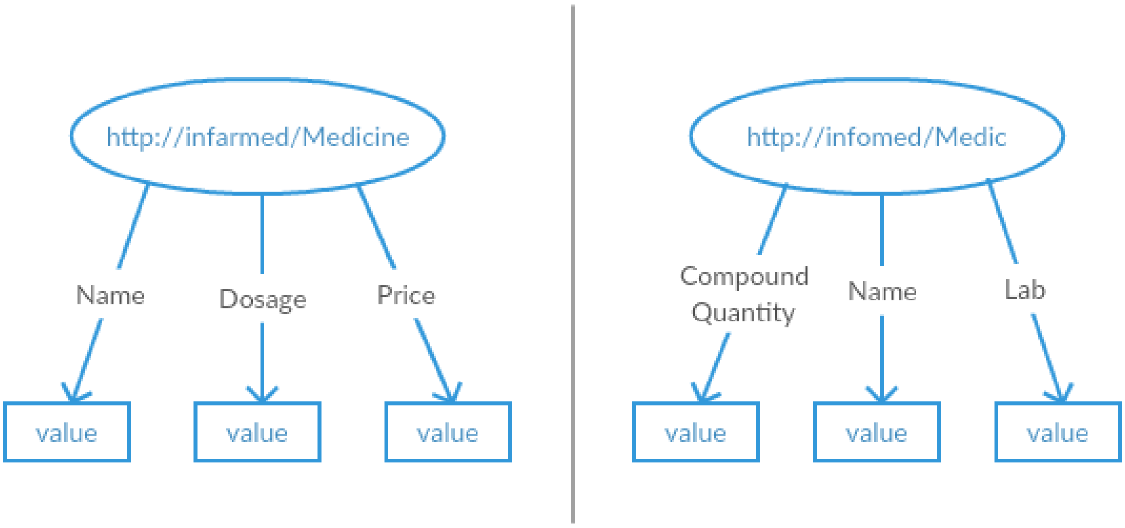


Figure 4.6: Example of two different data sources

The Figure 4.6 well illustrates the identity problem based on two different schema’s approaches which do not use unique identifiers to identify products. This way it is impossible to identify two references for the same real product based only on the URI and for that reason a mapping between each source’s properties has to be made.

In this case the only rule that could offer a sense of total similarity is by mapping “Name” with “Name” and “Dosage” with “Compound Quantity”. The remaining properties are added to the final object in order to give additional information that was previously unknown. All this is illustrated bellow in Figure 4.7.

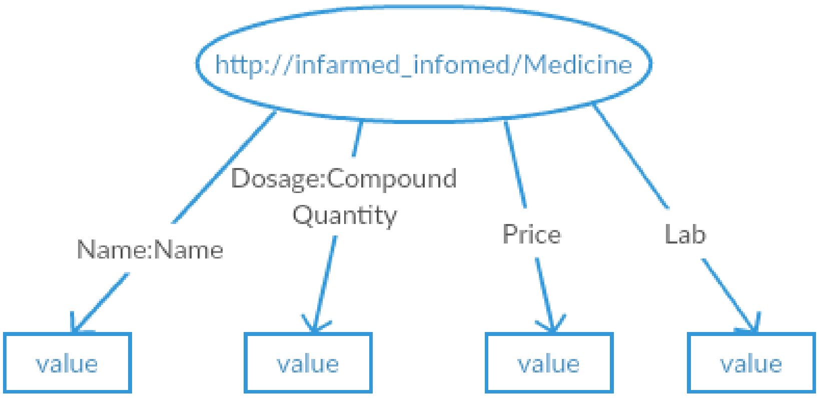


Figure 4.7: Result of applying the mapping rule

In terms of the developed software, the Data Curator is responsible for interact with the **Mapping Engine** by sending the list of mappings as input. That list is written according to a standard suggested by the application to the Data Curator, which describes the multiple mappings in this way: <**source1propertyX**>**-**<**source2propertyY**>**&**<**source1propertyZ**>**-**<**source2propertyW**> where X, Y, Z and W are the properties' names and the symbol "-" has the meaning "maps with". Just to clarify, this rule means that the property X from source 1 maps with the property Y from source 2 and also that the property Z from source 1 maps with the property W from source 2.

Similarly to the creation of aggregation rules, the creation of mapping rules is made by a particular engine, in this particular case, the **Mapping Engine** which has to save the new rule into the **Mapping Rules** repository for further use. The corresponding code block for this action is shown in the Listing 4.7. When applied, this rule is represented by a SPARQL CONSTRUCT query which is a very complex one. That is why it is needed the creation of all the query parts (schema, variables, where and optional) as we are going to show later.

```
public void createMappingRule(String source, String ruleName,
                                    Map<String, String> queryParts){
    String baseURI = source;
    Resource mainClass, newRule;
    List<Resource> properties = new ArrayList<Resource>();

    mainClass = dbMappings.createResource(baseURI + "/MappingRule");
    mainClass.addProperty(RDF.type, RDFS.Class);

    properties.add(dbMappings.createResource(baseURI + "/schema"));
    properties.add(dbMappings.createResource(baseURI + "/variables"));
    properties.add(dbMappings.createResource(baseURI + "/where"));
    properties.add(dbMappings.createResource(baseURI + "/optional"));

    for(Resource p: properties){
        p.addProperty(RDF.type, RDF.Property);
        p.addProperty(RDFS.domain, mainClass.getURI());
    }

    newRule = dbMappings.createResource(baseURI + "/" + ruleName);
    newRule.addProperty(RDF.type, mainClass.getURI());

    newRule.addProperty(dbMappings.getProperty(baseURI + "/schema"),
        queryParts.get("schema"));
    newRule.addProperty(dbMappings.getProperty(baseURI + "/variables"),
        queryParts.get("variables"));
    newRule.addProperty(dbMappings.getProperty(baseURI + "/where"),
        queryParts.get("where"));
    newRule.addProperty(dbMappings.getProperty(baseURI + "/optional"),
        queryParts.get("optional"));

    dbMappings.commit();
}
```

Listing 4.7: Create Mapping Rule

After the creation of all the desired mapping rules, the Data Curator has the possibility to view the results generated by applying a specific mapping rule to the repository. However and since the Data Curator's responsibility is not to choose the mapping rule to be applied in the system, a simulation is made in a way that the results are shown to the curator but without definitive implications to the system.

4.3.4 Perform searches

The fourth and final step is made by the general user, known as Search User once it is with that goal that s/he uses the application. Therefore, at the start, the Search User is informed about how the different sources of data were filtered and which mapping rules were created by the expert user. After that, an overview about the data sources to the general user is presented, i.e., which classes exist; which properties exist per class; and which values each property can assume.

Then, the Search User has to choose a mapping rule to be applied to the system, in order to obtain the information as completely as possible. The Search User is responsible for choosing the mapping rule because only s/he knows what makes sense given the search goals. If the property "Dosage" is irrelevant for the Search User because s/he only wants to know all the existing "Names" for a specific "Lab", that property will not be included on the mapping rule although it guarantees a better notion of similarity. Therefore, it is in the Search Users hands the responsibility of creating the connection between the **User Interface Engine** and the **Mapping Engine** inside the SWE by the choosing of mappings.

In the same way as choosing aggregation rules, choosing mapping rules means that the **Mapping Engine** asks for the characteristics of a specific mapping rule to the **Mapping Rules** repository, and then sends all the information to the **SPARQL Engine** that creates a SPARQL CONSTRUCT query to be applied to the RDF repository.

A CONSTRUCT query, as the name suggests, constructs a new RDF model based on the first given for the query. In our case, this query consists in four parts:

1. Schema - introduces new classes and properties;
2. Variables - defines all the properties for the new subject;
3. Where - defines the graph pattern to be found in the given model;
4. Optional - permits that some properties do not have a value (similar to the OUTER JOIN of SQL).

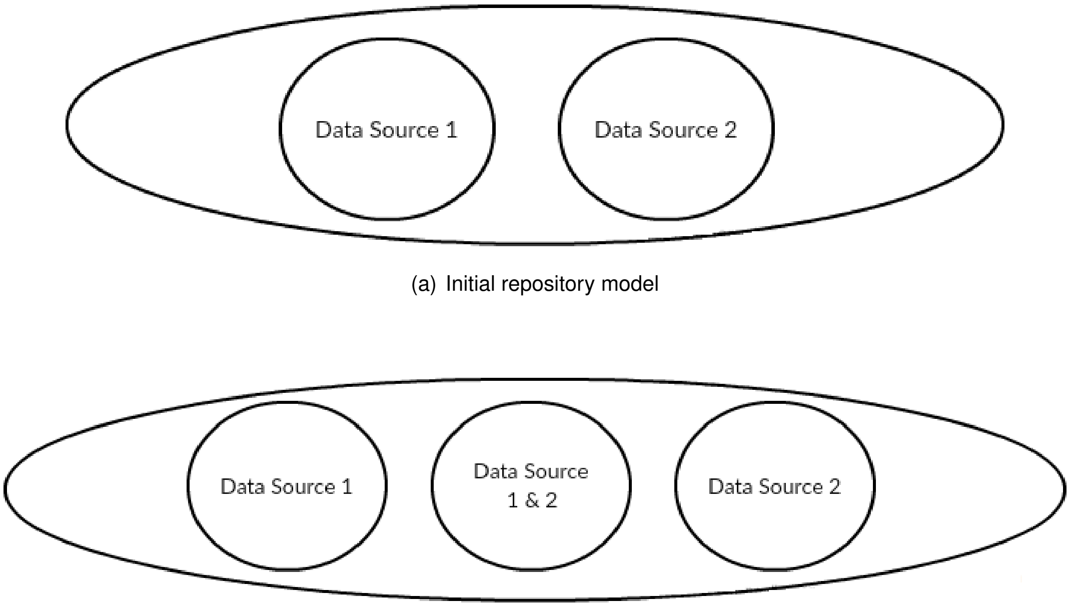
Since this type of query is very extensive in the real project, a pseudo-code version of it is shown in Listing 4.8. The four parts of the query are represented as follows: schema (lines 2 to 7), variables (lines 9 to 13), where (lines 16 to 17) and optional (lines 19 to 21).

```
1 CONSTRUCT {
        <ruleName> <RDF.type> <RDFS.Class> .
        <src1_src2/p1:p2> <RDF.type> <RDF.Property> .
        <src1_src2/p1:p2> <RDFS.domain> "ruleName" .
        <src1/p2> <RDFS.domain> "ruleName" .
        <src1/p3> <RDFS.domain> "ruleName" .
        <src2/p1> <RDFS.domain> "ruleName" .

        ?s1 <RDF.type> "ruleName" .
        ?s1 <src1_src2/p1:p2> ?o1 .
        ?s1 <src1/p2> ?o2 .
        ?s1 <src1/p3> ?o3 .
        ?s1 <src2/p1> ?o4 .
}
WHERE {
        ?s1 <src1/p1> ?o1 .
        ?s2 <src2/p2> ?o1 .

        OPTIONAL { ?s1 <src1/p2> ?o2 .}
        OPTIONAL { ?s1 <src1/p3> ?o3 .}
        OPTIONAL { ?s2 <src2/p1> ?o4 .}
}
```

Listing 4.8: Construct query to perform the appliance of a mapping rule in the system

When we first implemented this query we faced a significant challenge. Despite the using of the existing model as the base for the creation of a new model, the schema from the bases is not passed for the new model and this would have an impact throughout the whole system. I.e., whenever we list all the properties or classes in the model the result would be just the new classes and properties, once that information was created during the construct. That kind of result would not suit us due to the aim of presenting a complete information, so if we cannot access all properties we are not able to offer the whole available information to the user. Therefore, we first decided to add the property “subclassOf” to the new rule class with the vision of inheritance in mind. However, according to the conceptual model of this work, that vision did not mean what we wanted. In fact, a “subclassOf” tries to specify a generic concept, for example, the classes Drug and Nutritional Supplement are a specification of the class Pharmaceutical Product, therefore, Drug and Nutritional Supplement are sub-classes of Pharmaceutical Product. But what we are trying to do is to create a function that transforms a starting domain into an arrival domain. That being said, we solve the problem by extracting the schema of the starting model and added it to the final model.

Another detail we found about the CONSTRUCT query is that as this query is based on a given model to be able to create another one, at least one of the initial subjects as to be used. This means that whenever two entities, s1 and s2, are mapped, a new model is created but keeping all s1 initial subjects. However, our goal is to create a new model that is the intersection of the previous two or more data source models, where the entities inside it are new whole entities, i.e., there are new subjects (URIs) to identify new entities. With that purpose, we had to crawl the new model's list of statements, in order to



(a) Initial repository model

(b) Repository model after mapping

Figure 4.8: System's model evolution, before and after applying a mapping rule

update the subjects' URIs.

Figures 4.8(a) and 4.8(b) were placed in this paper to better describe the use of models.

The decision of separating the sources was due to the aiming of offer to the Search User the possibility of search for the information he needs, wherever he wants to. To that end, whenever a mapping rule is applied to the main repository, a new merged dataset is created and all the initial entities that mapped are removed from the other datasets, in order to avoid duplicates. This way, when the Search User performs a search s/he could decide in which dataset the query will be made, for example, according to the Figure 4.8(b) the user can choose one of the following datasets: Data Source 1, Data Source 2, Data Source 1 & 2 or the big dataset which encompasses all the others.

In fact, in our vision of the system, the goal was to present to the user all the information about a given topic as completely as possible, so, preferably the user would always search in the big dataset and the system would present all the mapped information plus the information that did not mapped but that references to the same search topic.

Since the aim of this work is to offer to general users a tool for performing searches with complete results, is essential to give to the user the ability to perform those searches. Therefore a SELECT query is created by the **SPARQL Engine** based on the search criteria given by the Search User during his interaction with the **Search User Interface**.

To give more detail, the **Search User Interface** presents to the user all the available datasets where the search could be performed. After choosing the dataset, it is time to choose the search criteria which is shown as a list of all the properties, that belong to the chosen dataset, and a input text box per property where the user can assign a specific value to it. After this interaction, the **Search User Interface** is responsible for sending the search criteria do the **SPARQL Engine** which, in its turn, will create a SELECT query to be applied to the RDF repository and, of course, to the chosen dataset. Listing 4.9 shows an example of a SELECT query.

```
SELECT *
WHERE {
        {       ?s a <chosenClass> . /* not applicable for the big dataset */
                ?s <www.infarmed_infomed.pt/Name:Name> ?Name .
                FILTER ( regex(str(?Name), 'Brufen'))
        }

        OPTIONAL { ?s <www.infarmed_infomed.pt/Dosage:CompoundQuantity> ?Dosage }
        OPTIONAL { ?s <www.infomed.pt/Lab> ?Lab }
        OPTIONAL { ?s <www.infarmed.com/Price> ?Price }
}
```

Listing 4.9: Select Query to perform a search

As can be seen, properties created by the mapping rule assume a new domain in their URI that is a composition of the linked data sources, while the remaining properties keep their original domain. This is made due to the LOD vision where existing schemas should be reused in the community.

Whenever a search is made to the whole set (containing all the other independent and intersected sets), the complex properties are separated to its original form, i.e., <http://infarmed_infomed/Dosage:CompoundQuantity> is decomposed into <http://infarmed/Dosage> and <http://infomed/CompoundQuantity>, so that the query is able to search all existing sets, each one with their own properties.

In summary, the process starts with the work of the **Data Converter Engine** that extracts and converts to RDF the information contained in different data sources. Then, the **Data Curator** interacts with the system through the **Data Curator Engine** to filter the data and create mapping rules that define similarity. Finally, the **Search User** chooses the mapping rule that is better suited to the search goals and performs searches through the interaction with the **Search User Engine**. All the three referred engines are connected via **Semantic Web Engine** at the “heart” of this system.

Chapter 5

Evaluation

This work was evaluated with two approaches: first, with an evaluation based on metrics and secondly, with another based on tests with real users. The second part of the evaluation was itself divided into two parts: an evaluation questionnaire with expert users and another with common users, in order to simulate both the Data Curator and the Search users.

5.1 Pharmaceutical Scenario

The “Pharmaceutical Products” scenario was chosen to be presented to the real users at the time of interacting with the system because the repository was populated by us. Therefore, we had the knowledge about the schema and also about which entities were referring to the same real world object beforehand. Given that, we were able to validate the engine results according to the number of false positives and false negatives returned.

The initial goal of this work was to cross information between that global dataset called Drug Bank and the Portuguese medicine regulator INFARMED[1], in order to present a more detailed information about the commercialized medicines in our country. However, the translation proved to be an obstacle difficult to be transposed due to its consequent time overload.

Under these circumstances, we found another data source that also belongs to INFARMED called Infomed, containing only medicines for human use and described in Portuguese. In this way, we not only avoided the time overload with translation but we also constructed a prototype of a system where the information came from different datasets and it is merged to provide richer knowledge to the users. We see this case as a prototype because basically the information comes from the same source, which is INFARMED. However, and strange as it may seem, different information could be obtained from each of the sources. Additionally one of them does not even have product’ identifiers.

[1] http://www.infarmed.pt

5.1.1 Filtering Data

As stated before, each database could have duplicate information and in this specific case of Infarmed we found some duplicate instances. Therefore, and not to compromise the mappings performance, we first decided to filter each data source based on their specific properties.

The data source named “Infarmed” is composed by 21 properties, 11 of which belong to the drug package leaflet assuming large blocks of text. From the 10 remaining properties we only use the first 5 to define duplicate information. The ten properties have the following names:

- Nome do Medicamento (Drug Name);
- Substância Ativa (Active Substance);
- Dosagem (Dosage);
- Forma Farmacêutica (Pharmaceutical Form);
- Génerico (Generic);
- Tamanho da Embalagem (Package Size);
- Código Nacional para a Prescrição Eletrónica de Medicamentos (CNPEM) (Prescription Code);
- Preço (Price);
- Folheto Informativo (FI) (Information Leaflet);
- Resumo das Características do Medicamento (RCM) (Drug Characteristics Summary).

Where **CNPEM** stands for prescription code, **Genérico** states if the medicine is, or not, a generic product and the last, FI and RCM, that are URLs that refer to the package leaflet. Throughout this process we assumed that the package size (**Tamanho da Embalagem**) and the **CNPEM** represented irrelevant information due to the fact that, for the same medicine, the number of pills contained in the pack could vary, as well as the price associated to it. Besides that, we found out that the code **CNPEM** was not a unique code, i.e. the same code can represent different drugs. That being said, we assumed that for a product to be considered a copy of another it must have the same values as the other for the first 5 properties listed before. That way, we learned how many duplicate products existed in the “Infarmed” database and had the ability to evaluate our own filtering rules as “Data Curators”.

For the “Infarmed” dataset (composed by 326 products, 74 of which are duplicates), we have evaluated 33 aggregation rules which differ from each other in terms of complexity (number of properties tested together) and variety (for the same number of properties tested together, there are compositions of different properties).

For filtering, with these 30 different rules we realized that for this particular data source, we only become to get close to the true number of duplicates when we test together at least 3 of the 5 properties. For a smaller number than that, none of the existing properties was sufficiently descriptive.

Based on our first match rule, we continue to test variations of it and concluded that the combination **Nome do Medicamento & Dosagem & Forma Farmacêutica** represents the best combination of properties to reach a correct result. Also, we have found that the property **Forma Farmacêutica** is more valuable than the property **Genérico** and that the property **Substância Áctiva** produces worse results than **Nome do Medicamento** when combined with others. In fact there is no rule without the property **Nome do Medicamento** that can reach the right number of duplicate instances.

We evaluated the obtained results in terms of number of duplicates, true positives, false positives, precision and recall. Where Precision (5.1) gives us the percentage of correct duplicates in the set of duplicates returned, and Recall (5.2) shows the percentage of correct duplicates in the set of duplicates.

$$Precision = \frac{Number\ of\ duplicates\ correctly\ detected}{Number\ of\ duplicates\ detected} \tag{5.1}$$

$$Recall = \frac{Number\ of\ duplicates\ correctly\ detected}{Number\ of\ existing\ duplicates} \tag{5.2}$$

Regarding the number of found duplicates and the number of false positives, these two metrics reveal a directly proportional relation, i.e. the greater the number of duplicated products found, the greater the number of false positives. Moreover, the greater the number of properties included in a rule, the greater the accuracy of that rule, which will mean a good aggregation rule. In terms of precision and recall we have concluded that as the number of duplicates increases, the precision decreases, while as the number of false positives comes close to 0, the recall comes close to 1, as expected. Table 5.1 shows some examples from where the conclusions were drawn.

Rule	**Detected duplicates**	**False positives**	**Precision**	**Recall**
N (Nome do Medicamento)	120	56	0.533	0.865
S (Substância Activa)	291	271	0.068	0.270
F (Forma Farmacêutica)	302	293	0.029	0.122
D (Dosagem)	272	245	0.099	0.365
T (Tamanho da Embalagem)	280	279	0.004	0.014
G (Genérico)	324	323	0.003	0.014
C (CNPEM)	196	192	0.020	0.054
P (Preço)	136	129	0.051	0.095
N & S	120	56	0.533	0.865
N & D	77	4	0.948	0.986
N & F	111	46	0.586	0.878
S & D	245	209	0.147	0.486
S & F	267	236	0.116	0.419
D & F	245	210	0.143	0.473
D & G	253	222	0.123	0.419
N & S & D	77	5	0.935	0.973
N & D & F	74	0	1	1
N & D & G	77	4	0.948	0.986
S & D & F	232	192	0.172	0.541
N & S & D & F	74	0	1	1
N & S & D & G	77	4	0.948	0.986
N & D & F & G	74	0	1	1
S & D & F & G	211	167	0.209	0.595
N & S & D & F & G	74	0	1	1

Table 5.1: Infarmed Filtering Rules

5.1.2 Filtering Data From a Second Data Source

The data source named “Infomed”[2] is composed by 442 products, 37 of which are duplicated, and by 5 properties that are all used for the filtering process as a way of finding duplicates. Those properties have the following names:

- Nome do Medicamento (Drug Name);
- Nome Genérico (Generic Name);
- Dosagem (Dosage);
- Genérico (Generic);
- Titular (Holder).

We assumed that for a product to be considered a copy of another it must have the same values as the other for all the 5 properties listed above. That way, as in the first source, we learned how many duplicate products existed in the database and again the ability to evaluate our own filtering rules.

We have evaluated 30 aggregation rules which differ from one another in terms of complexity and variety. With these rules we realized that for this data source the property **Nome do Medicamento** was the most relevant when grouped with others. Also, we realized that **Dosagem** is a good indicator when analyzing duplicates while **Genérico** does not help much. Furthermore and despite being a great indicator in this database, the **Dosagem** property works better when brought together with the property **Nome do Medicamento**.

Similar to Infarmed, the metrics used were the number of found duplicates, true positives and false positives, and the values of precision and recall. Once again the results have shown a proportional relation between the number of duplicates and number of false positives returned. Therefore, as the number of copies found increases, the number of false positives also increases.

In terms of precision and recall we also concluded that as the number of duplicates increases, the precision decreases. However, in this data source, it is possible to find high recall values even when false positives do exist, for that to happen it is only need to find all the correct duplicates even though the filtering rule may detect more duplicates than the expected ones.

In this dataset, it is possible to find a perfect match between the number of duplicate results and the true quantity of duplicates in the database with only the conjunction of two properties. In the Table 5.2 some of the 30 rules are presented in order to show how we get to the conclusion mentioned before.

5.1.3 Mapping Data From Two Data Sources

Similarly to the filtering part of this work, in order to evaluate the mappings it was necessary to have prior knowledge of how many matches there were between products from the two different sources, in this case: Infarmed and Infomed.

For that purpose, we first analyzed which properties each source had and decided if properties with the same name could represent a match, by looking to the values they assume. Secondly, for properties

[2] http://www.infarmed.pt/infomed/inicio.php

Rule	Detected duplicates	False positives	Precision	Recall
N (Nome do Medicamento)	202	184	0.089	0.486
NG (Nome Genérico)	381	370	0.029	0.297
D (Dosagem)	315	298	0.054	0.459
G (Genérico)	440	440	0	0
T (Titular)	341	328	0.038	0.351
N & NG	202	184	0.089	0.486
N & D	37	0	1	1
N & T	200	182	0.09	0.486
NG & D	248	219	0.117	0.784
NG & T	260	242	0.069	0.486
D & G	276	259	0.062	0.459
D & T	77	40	0.481	1
N & NG & D	37	0	1	1
N & D & G	37	0	1	1
N & D & T	37	0	1	1
N & G & T	194	176	0.093	0.486
NG & D & G	207	177	0.145	0.811
NG & D & T	63	26	0.587	1
N & NG & G & T	194	176	0.093	0.486
N & D & G & T	37	0	1	1
NG & D & G & T	63	26	0.587	1
N & NG & D & G	37	0	1	1
N & NG & D & G & T	37	0	1	1

Table 5.2: Infomed Filtering Rules

with different names we also had to base our decision on the values that those properties assume and see if there were similarities.

As said in Section 4.1 Architecture, we assume a contribution from a human "Data Curator" user, with cognitive thinking, who will create only mapping rules that could make sense. To better explain we will focus on a brief example.

Source 1	
Medicine_Name	**Dosage**
Aspirin	100 mg
Ben-U-Ron	500 mg
Brufen	400mg

Source 2	
Med_Name	**Generic**
Brufen	Yes
Aspirin	No
Ibuprofen 400	Yes

Table 5.3: Example of two data sources with different properties

In Table 5.3 presents two data sources with two properties each. By looking through those properties and the values that they assume, a human user can quickly associate the properties "Medicine_Name" and "Med_Name" even though they had different names as identifiers. Furthermore, the human user can also conclude with certainty that the remaining properties, "Dosage" and "Generic", do not match through the values presented in each one.

In this work, the human Data Curator realized that there were only 4 properties between the 2 sources (Infarmed and Infomed) that actually could be equivalent and therefore, 15 rules were created and taken into account for the system evaluation.

Given the properties from Infarmed (Nome_do_Medicamento (N1); Substância_Activa (S); Dosagem (D1); Genérico (G1)) and from Infomed (Nome_do_Medicamento (N2); Nome_Genérico (NG); Dosagem (D2); Genérico(G2)) we had assume the following 15 mapping rules, where the character "-" means "map":

- R1. N1-N2
- R2. S-NG
- R3. D1-D2
- R4. G1-G2
- R5. N1-N2 & S-NG
- R6. N1-N2 & D1-D2
- R7. N1-N2 & G1-G2
- R8. S-NG & D1-D2
- R9. S-NG & G1-G2
- R10. D1-D2 & G1-G2
- R11. N1-N2 & S-NG & D1-D2
- R12. N1-N2 & S-NG & G1-G2
- R13. S-NG & D1-D2 & G1-G2
- R14. N1-N2 & D1-D2 & G1-G2
- R15. N1-N2 & S-NG & D1-D2 & G1-G2

Each of these rules were implemented in the system with 252 products belonging to the first source and 405 belonging to the second source, 120 of which referring to the same real world object. The above rules were tested with the following metrics:

- Number of product matches;
- Precision;
- Recall;
- Distance to expected match number.

The first metric gives us the number of matches between products from different sources, the second metric is obtained through the equation (5.3) and gives the information of how many of the matches done are actually correct. The third metric gives us the rule coverage regarding the number of mappings within the expected set and is computed using the equation (5.4). Finally, the last metric tells us how many products could be matched at most: in our particular case we have 252 products from Infarmed and 405 from Infomed, which means that we can only match 252 products at most. Therefore, every match that we get beyond 252 are considered as excess. Thus, to calculate the distance between the number of matches obtained and the number of expected ones, we first compute the excess produced by the rule with the equation (5.5), where the "Maximum number of matches" represents the number of matches that can be made at most (252 in this particular case). Then, we compute how many possible matches there are in our system, which is given by the formula (5.6). Finally, we normalize the distance obtained earlier according to the equation (5.7) to show the values in a common scale.

$$Precision = \frac{Number\ of\ matches\ correctly\ detected}{Number\ of\ matches\ detected} \tag{5.3}$$

$$Recall = \frac{Number\ of\ matches\ correctly\ detected}{Number\ of\ existing\ matches} \tag{5.4}$$

$$Absolute\ Distance = Number\ of\ obtained\ matches - Maximum\ number\ of\ matches \tag{5.5}$$

$$Possible\ Matches = Number\ of\ Source_1\ products \times Number\ of\ Source_2\ products \tag{5.6}$$

$$Match\ distance = \frac{Absolute\ Distance}{Possible\ Matches} \tag{5.7}$$

Rule	Product matches	Precision	Recall	Distance
R1	287	0.418	1	3.43e–4
R2	2761	0.043	1	2.46e–2
R3	2542	0.047	1	2.24e–2
R4	50010	0.002	1	4.88e–1
R5	287	0.418	1	3.43e–4
R6	120	1	1	0
R7	280	0.429	1	2.74e–4
R8	854	0.141	1	5.90e–3
R9	1785	0.067	1	1.50e–2
R10	1592	0.075	1	1.31e–2
R11	120	1	1	0
R12	280	0.429	1	2.74e–4
R13	615	0.195	1	3.56e–3
R14	120	1	1	0
R15	120	1	1	0

Table 5.4: Used mapping rules and metrics evaluation

From the Table 5.4 we can conclude that when mapping “Nome_do_Medicamento” with “Nome_do_Medicamento”, each from one of the two sources, the obtained results are better than the ones achieved with mapping “Substância_Activa” with “Nome_Genérico”. This could be easily seen by compare the R6 and R8 rules, where R6 has a precision of 1 while R8 presents a precision of only 0,14. Similarly, mappings of properties named “Genérico” produce worse results, in terms of precision, than mappings of properties names “Dosagem” (ex. comparison between R8 and R9). That being said, we could then expect that the combination of R1 and R3 will produce better results than the others which is confirmed by the comparison of, once again, R6 and R8 but also by comparing R12 with R14.

In terms of precision, it is clear that having high number of matches does not mean that the mapping rule is better than the ones with lower numbers. The latter are the ones that should be considered closer to the expected matches. For example, comparing the rule R4 with the rule R15 it can be seen that the high number of product matches in R4 corresponds to a really low precision, while in R15 it is the opposite. This happens because precision uses the number of true positives to compute its value and, in this particular case, we know that only 120 matches are actually correct, so for a lower number of matches (below or equal to 120), the precision tends to be better.

The recall value shown to be always one because it takes into account the number of correct matches found and the number of existing correct matches. Since every rule found all the existing correct matches, even though some may found more matches than the expected ones, the recall does not give us useful information to draw conclusions, instead, the distance between the obtained results and the expected ones is more valuable.

Regarding the "Distance" metric, for rules that present number of matches below the maximum number of expected matches, zero is assumed as the distance value once they do not cross the defined boundary of maximum number of expected matches. To draw conclusion about those kind of rules, the precision values assumed are crucial.

Overall, we can conclude that the farther from the expected is the rule, the worse it is. Therefore, if we sort our rules by distance we will obtain the following list, ordered by ascending distance value: [R6, R11, R14, R15, R7, R12, R1, R5, R13, R8, R10, R9, R3, R2, R4], where R6 represents the better rule while R4 is seen as the worse one to be applied to the system.

Due to the precision computation, we had to know how many correct matches there were in the system, therefore we have calculated the distance between the number of matches obtained in each rule and the number of real expected matches to compare with the distances previously computed and we can conclude that for our example the distances between the matches obtained in each rule and both the maximum possible matches and correct matches, do not differ a lot. From the values presented in Table 5.5, this means that if we sort the rules, once again by ascending distance, we will obtain the same list as before: [R6, R11, R14, R15, R7, R12, R1, R5, R13, R8, R10, R9, R3, R2, R4].

Rule	Distance from ideal result
R6	0
R11	0
R14	0
R15	0
R7	1.57e–3
R12	1.57e–3
R1	1.64e–3
R5	1.64e–3
R13	4.85e–3
R8	7.19e–3
R10	1.44e–2
R9	1.63e–2
R3	2.37e–2
R2	2.59e–2
R4	4.89e–1

Table 5.5: Rule distance from the ideal expected matches

Regarding this study, the metrics "Number of matches returned" and "Distance from the expected match number" are directly proportional, i.e. whenever the number of match returned increases, the distance to the expected matches also increases, which means that the rule is farther from the truth. Thus, the main conclusion is that Curators should look for rules with the following characteristics:

- Number of matches close to the maximum expected number of matches;
- High precision values (close to 1);
- Distance values close to 0.

The Distance metric was conceived with the aim to help the curator to construct better mapping rules by letting her/him know the distance between the new mapping rule and the expected result, being that the lower the distance, the better the created rule.

5.2 Cinematographic Scenario

When a user wants to know all the information about a given movie, s/he has to search on the Web and the search engine will present a list of related information, such as sites about movies, fan communities, reviews about the movie and other items like books or posters. After that, the user has to find by her/himself which sources of information are referencing the intended product and decide in which information s/he is going to trust, according to its source. This search approach implies a significant workload to the user given that, most of the times, the sites returned by the search engines refer to products similar or somehow related to the intended one. For example, if a user searches for the movie “Frozen”, s/he will be faced with the film but also dolls, performances at Disney Land, fans Facebook pages, reviews, games and much more.

People fond of cinematography tend to search for movies’ ranking before afford the intended movie or to watch to it. Furthermore, they tend to search for movies with the same actors and the same producers that made other films that they have enjoyed. Therefore, these people spend a significant amount of time visiting the web pages that contain reviews and filtering the ones which have information about a given actor career and biography, instead of fans communities and others.

With the developed work we want to be able to present to the user only relevant information, in this case only in the scope of cinematography, where s/he can find more detailed information about the actors of a given movie gather in one unique place. The provided information is only about the product (movie) that was searched for. The mapping rule will be done by a specialized user and then chosen by a regular search user.

With this new scenario we were able to demonstrate that the system is domain independent, i.e. it is able to perform the same actions (filter, map and search) for another context just by adding some code lines.

Once we chose information available in repositories from the Linked Open Data community, we had to change the way the queries were made in order to perform them against SPARQL Endpoints instead of a local repository, which also shows its versatility since we did not contribute to the schema development.

5.3 User Validation

In order to infer the application utility and complexity, we decided to perform some tests with real users.

Those tests were made to twenty users, on a quiet and study environment to avoid distractions, in an academic context. The users were aged between 18 and 25, with few exceptions, and were at the majority males (60%) for the Data Curator tests and females (55%) for the End-users ones.

To perform these evaluation a smaller version of the original database was used so that the users were able to understand the results and not be overwhelmed by tables with hundreds of lines.

To each separate version, i.e. Data Curator or Search User, the test consisted of three tasks that were increasingly difficult. The objectives of this work and the basic concepts needed to interact with the interface were briefly explained to the users, as well as the application context.

In the end of each test (group of three tasks), the users were asked to answer a small satisfaction questionnaire which can be found in the Appendix.

5.3.1 Data Curator Users

In our work the Data Curator role is performed by a human that has the logical reasoning to decide which properties have to be identical to confirm that two RDF entities are referring the same real world product and can be merged. Therefore, we have asked twenty students of Computer Science Engineering to simulate this role in our application and to give us some feedback in terms of utility and complexity of understanding the system.

To that matter, the users were asked to perform three different task with increasingly difficulty. Those tasks are listed and briefly explained below.

5.3.1.1 Task 1. Pair Properties

This particular task was asked with the purpose of giving them a brief idea of how the mappings work.

In this task the users firstly had to explore the system (available sources; sources' properties; existing pharmaceutical products inside each source) and secondly they had to find which property from the Infomed data source could pair with the **Substância_Activa** property from the data source Infarmed.

In this exercise the users revealed a strong logical thinking, spending 10 seconds on average to find the right answer.

5.3.1.2 Task 2. Filter Data Sources

Before the users get to the point of mapping the data sources, they had to filter them. As it was explained before in this thesis, many times databases have duplicate information, for example, for the same smartphone model there are at least two different colors available (black and white) but if we think in terms of relevance probably the "color" feature is not relevant enough to determine that two products are actually different. Instead, there would be interesting to merged the information and present the user with only one entity that for the property “Color” assume both values combined “Black; White”.

That being said, the users had to perform three steps in this task:

- Verify the existing products in the Infomed data source;
- Create a new filtering rule named “ByName” for the source Infomed that aggregates products by the property **Nome_do_Medicamento**;
- After creating the rule, choose that same rule to filter the Infomed dataset and the rule “None” to filter the dataset Infarmed;
- Verify results.

With this task the users had to understand what filtering by the property **Nome do Medicamento** meant and which results were expected.

After understanding that, the users were able to recognize that the rule created by them was not a good rule since it assume that two products are the same by only verifying if they have the same name.

5.3.1.3 Task 3. Map Data Sources

In this task the users were asked to firstly filter each one of the data sources available, then to map those data sources by name and finally to verify the obtained results.

- Choose the rule named “ByNameSubsDoseFormaGen” to filter the data source “Infarmed” and the rule named “ByAll” to filter the “Infomed” data source;
- Create a new mapping rule which creates a match between the properties “Nome_do_Medicamento” and “Nome_do_Medicamento” from the sources “Infarmed” and “Infomed” respectively.
- Go to the next page and choose the mapping rule you have created in the previous subheading;
- Verify results.

With this exercise the users had to create a new mapping rule, with a specific syntax, that merged information of products from each one of the data sources that present the same value for the name properties. Therefore, what we asked was for them to map the two sources by name of medicine.

Although this was the most time-consuming task, since the users had to do three different things (filter, create mapping and choose that mapping), they were able to verify that the results were the ones they were expected and also that once again, this rule was not as good as it seemed at first sight.

Table 5.6 presents the time, on average ($\bar{\chi}$) and the standard deviation (σ), that the users take to perform each task.

Task	$\bar{\chi}$	σ
T1	0.10	0.07
T2	1.20	0.06
T3	1.42	0.07

Table 5.6: Average spent time by Curator users in the tasks (in minutes)

After performing each task, the users were asked to answer to a brief questionnaire which revealed that the majority of the interviewed had background knowledge in the area of databases and performs searches on the Web almost every day.

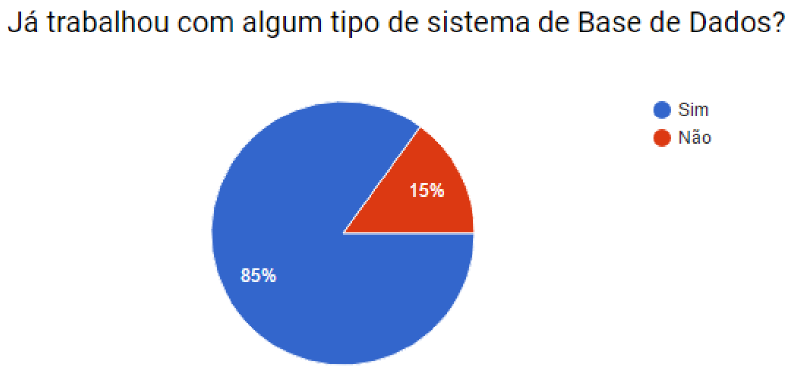


Figure 5.1: Knowledge in databases by the Curator users

Besides that, almost all of the inquired said that the information given in the application was sufficient to understand and interact with the system. As it was expected the users revealed that the last task was the most complex one of the three while the first one was the easiest.

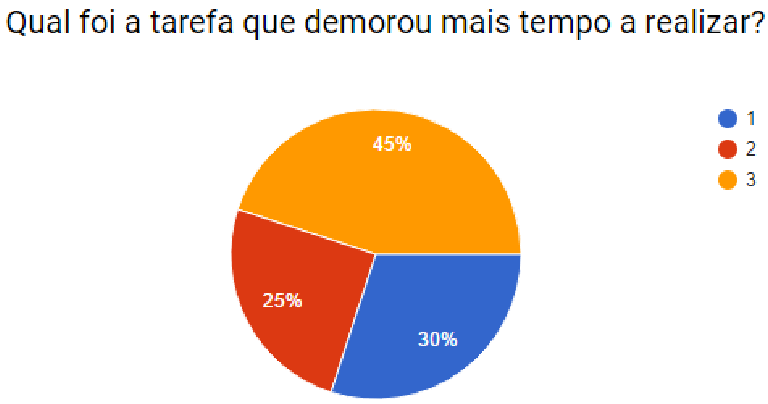


Figure 5.2: Curator users most time-consuming tasks

Overall, the users considered the application as being very useful for the daily life of a common user and that would like to have this system available in the contexts of: technological products, online stores and academic documents.

With respect to the interface, the users suggested some small changes to improve the usability and understanding of the system, such as: simplify the way properties are presented and give more information whenever there are drop-down buttons.

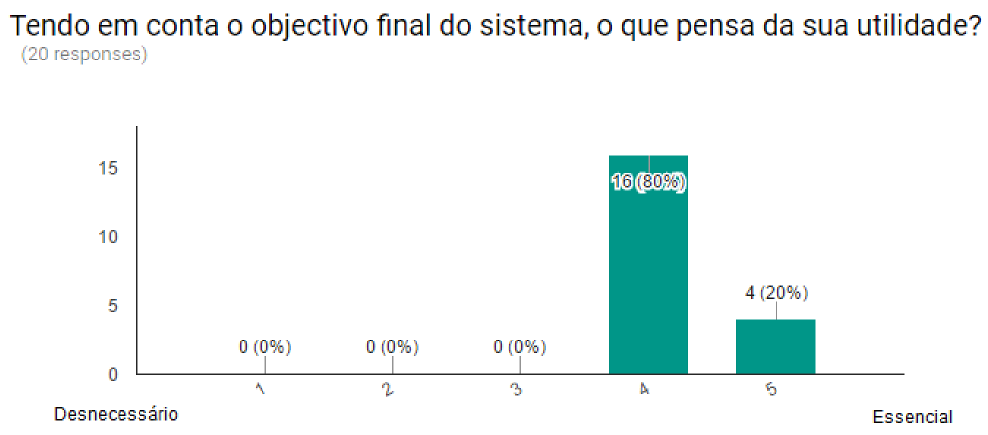


Figure 5.3: System utility in the Curators' perspective

5.3.2 End-users

After the solution had satisfied the functional requirements of the architecture described in Section 4.1, it was given to twenty end-users to allow them to compare browsing for product information using the traditional web search and our semantic-aware approach. Therefore, the end-users were given a computer and a list of three tasks to perform using the developed application.

Since these type of users could be common people, without background on databases or computer science, the tasks chosen for them were relatively easy. They only had to simulate a choice of an existing mapping rule that better suited their needs, and then they had to perform a search, via the application, by attributing a value to a specific property.

Each one of the "Search Users" tasks is presented below.

5.3.2.1 Task 1. Simple Search

For this tasks the user had to do the following steps:

- Explore the system (available sources; properties of each source; existing products in each source);
- Go through the system until arriving to the "Search Criteria" page and then make two consecutive searches:
  - Search on "Infomed" by "Nome_do_Medicamento" with the value "Betadine";
  - Seach on "Infarmed" by "Nome_do_Medicamento" with the value "Betadine".

This task has served to show to the users that if they search for Betadine in Infomed they will get too few information, while if they search for the same product in Infarmed they will get much more information. Besides that, it was a product that was common in both databases - that was made with the purpose of asking them if it would make sense to merge the two products and offer a more complete and unified information.

After this they were introduced to the second task.

5.3.2.2 Task 2. Map Disparate Data Sources

For this task the user had to do the following two steps:

- Choose the rule "ByNameSubsDoseGen" to map the databases of the system;
- Perform a search on the "All" model by the joint property "Nome_do_Medicamento:Nome_do_Medicamento" with the value "Betadine" and verify the obtained results.

With this task the users were able to see the problem found in the first task solved, i.e. in the end they could see two products Betadine, one from Infomed that mapped with another from Infarmed and also the other Betadines that still exist but that did not map with other products.

This second task revealed a further understanding about the problem stated in this thesis and a confirmation of the returning results.

5.3.2.3 Task 3. Complex Search

Finally the users got here and were asked to test the application to infer if it was capable of searching by more than one property at a time.

That way, the users were asked to do the following:

- Choose the rule "ByNameSubsDoseGen" as you did in the previous task;
- Search in the system on the "All" model by "Forma_Farmaceutica" with the value "Comprimido" and by the joint "Genérico:Genérico" property with the value "Sim".

With this task the users got to know the system better in terms of how the mapping is done and see different features of it.

In average, the users spend more time in the first task which means that once they are comfortable with the system they could be quicker and that the system is easy to learn and to interact with.

Below is a table with the average time spent (in minutes) by the users in each of these tasks, as well as the standard deviation:

Task	$\bar{\chi}$	σ
T1	1.02	0.04
T2	0.49	0.07
T3	0.50	0.05

Table 5.7: Average spent time by the Search users in the tasks (in minutes)

According to the questionnaires, most of the users said that they did not need more information to understand which fields they have to select to perform the tasks. Also, more than half of the inquired

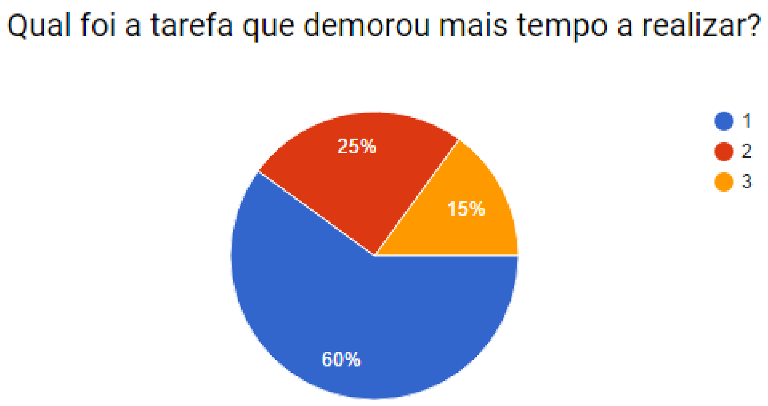


Figure 5.4: Search users most time-consuming tasks

stats that all the tasks were simple and almost all of the inquired answered that the system had great usefulness in their lives.

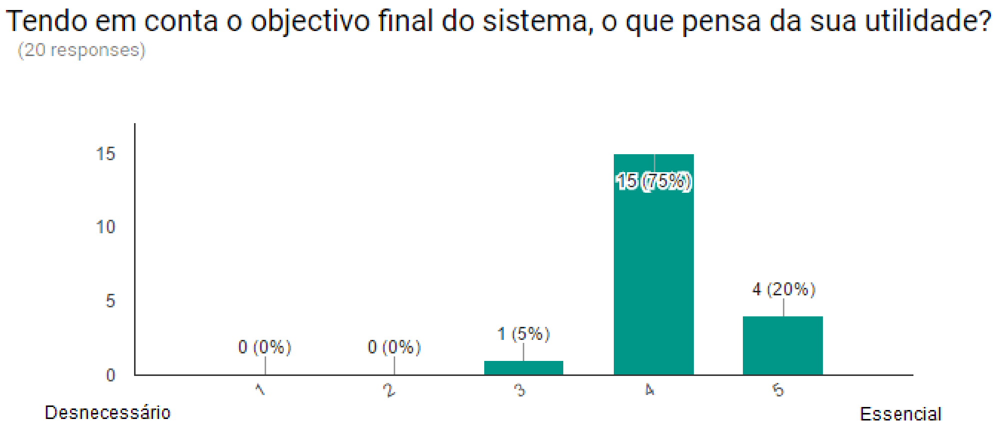


Figure 5.5: System utility in the Search users' perspective

Besides that, almost all of the users said that would like to have this application at all of the existing platforms (computer, tablet and smartphone) and the great majority in the context of technological products.

Some improvements to the interface were also suggested.

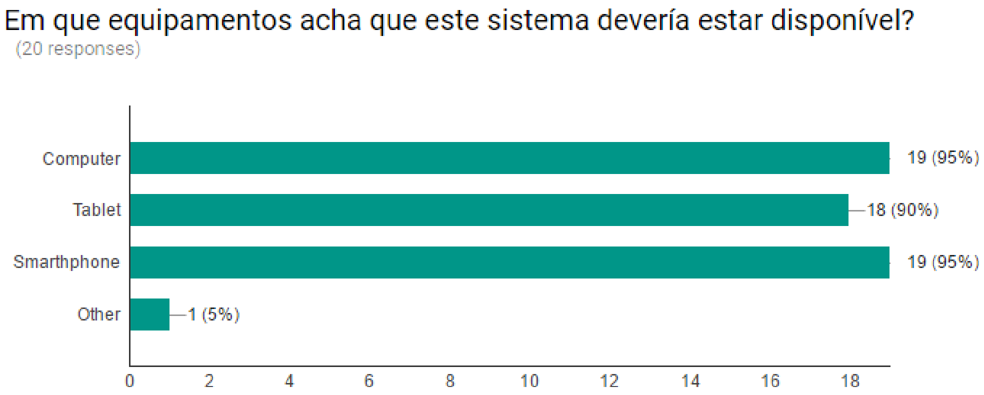


Figure 5.6: Platforms where the Search users would like to have the system available

Chapter 6

Conclusion

Nowadays, with the increasing amount of data available on the Web it is becoming imperative to develop new ways of searching. The existing search engines return an amount of possibilities that make the user's job a lot harder, because it is necessary to filter which of them are actually useful. Besides that, the user has to match and combine the information manually because it is unstructured and presents some redundancies, i.e. different web pages refer the same information which makes the user waste time.

The Linked Product Data solution came to address this problem since it provides tools to structure data about products. Therefore, the aim of this work was to facilitate the user's searches in a way that s/he will not has to be concerned about which sources describe the product that s/he was looking for. Instead, all the information needed was presented in a single place rather than in a considerable number of distinct web pages.

The work presented in this document had to deal with the problems of identity and redundant information since similar data is kept in distinct databases, which have different IDs for the same real products. In order to overcome these obstacles, we first suggested an examination of the available data sources by an expert user called Data Curator, with the knowledge to create rules that will aggregate duplicated data. After testing the created rules, the Data Curator also has to decide which of those rules are the most reliable to be applied to the system, by comparing the expected results (known “a priori”) and the results obtained after choosing the filtering rule. Second, we suggested that the expert user has to create mapping rules, i.e. s/he has to decide which properties of one data source map to other properties from another data source. This is again evaluated by the comparison of the expected results and the ones returned by applying the rule in the system.

After solving these obstacles we were able to offer common users an interface, on one specific context at a time, where they can search in a property-value perspective and obtain the results in table format where all the information appears unified - the one that was merged and the remaining information from each one of the data sources.

According to the users evaluation, we were able to verify that the users workload was reduced, that the designed interface is not too complex to be understood, although there are still some improvements to be taken into account, and that it is seen as a tool of great usefulness for people who perform searches daily.

6.1 Contributions

The major achievements of the present work was providing a way to address the Identity Problem, a way to deal with duplicate information inside a database and the use of metrics to evaluate the rules created to filter, a manner to map the information from different datasets and an application for users to test.

The first problem, brought by the new Web era where lots of redundant data is kept in disparate data sources with different data structures and different identifiers for properties and products, was solved by the introduction of a Data Curator, an expert user, with enough cognitive ability to compare the schemas of the available data sources and by the developed mapping tool.

To deal with duplicates it was also the Data Curator the one responsible for creating rules which will be used to aggregate information that assumes the same values for a set of properties. Similarly, to map different sources, the Curator user had to create rules that will make a correspondence between properties of different sources.

To evaluate the system, false positives and true positives were used as metrics. A correlation between the number of matchings returned by the appliance of a rule and the maximum possible matches between two sources was also used as metric.

The developed application allow users to search for pharmaceutical products (and cinematographic works) and get all the information from two different sources in an unified way. Besides being able to search in the mapped results, the users can also choose to search in each one of the available sets (source 1, source 2 and mapped source) or in all of them at the same time, this way they will never lose information even though it may not have been mapped.

6.2 Future Work

Although there were interesting results, there is still some work to be explored in this context. For example, the last contribution can be used to evaluate mapping rules created automatically by the computer. A semi-automated approach could be developed, where the Data Curator is responsible for the evaluation of the rules created automatically by the program, which means that he only will have to decide which rules best suits his necessities.

For an automated approach, the computer computation would be the responsible for creating the mapping rules based on all possible combinations of existing properties and then, those rules can be evaluated by the created metric to infer accuracy.

Finally, it would be interesting to add the Good Relations Ontology [31] or other ontologies, already used for e-commerce, to structure the data from the beginning of the whole process using existing classifications.

Bibliography

[1] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 56:3825–3833, 2012.

[2] H. S. Pinto, A. Gómez-Pérez, and J. P. Martins. Some issues on ontology integration. *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, 1999:1–12, 1999.

[3] D. S. Wishart, C. Knox, A. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali. DrugBank: A knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Research*, 36, 2008.

[4] O. Hassanzadeh and M. Consens. Linked movie data base. In *CEUR Workshop Proceedings*, volume 538, 2009.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a Web of open data. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4825 LNCS, pages 722–735, 2007. ISBN 3540762973.

[6] D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist*. 2011. ISBN 9780123859655.

[7] N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited, 2006.

[8] F. Manola, E. Miller, and B. McBride. RDF primer. *W3C recommendation*, 10(February 2004): 1–107, 2004.

[9] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation*, 10:1—-20, 2004.

[10] M. Bergman. Advantages and Myths of RDF, 2009.

[11] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. *W3C Recommendation*, 2009:1–106, 2008.

[12] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International journal on Semantic Web and Information Systems*, 2009.

[13] T. Berners-Lee. Linked Data. 2006.

[14] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the Web of data. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5823 LNCS, pages 650–665, 2009. ISBN 364204929X.

[15] B. Alexe, L. Chiticariu, R. J. Miller, and W. C. Tan. Muse: Mapping understanding and design by example. In *Proceedings - International Conference on Data Engineering*, pages 10–19, 2008. ISBN 9781424418374.

[16] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. *ACM SIGMOD Record*, 30:485–496, 2001.

[17] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. Fernandes, and C. Hedeler. Feedback-based annotation, selection and refinement of schema mappings for dataspaces. *EDBT '10 Proceedings of the 13th International Conference on Extending Database Technology*, pages 573–584, 2010.

[18] H. Cao, Y. Qi, K. S. Candan, and M. L. Sapino. Feedback-driven result ranking and query refinement for exploring semi-structured data collections. In *Proceedings of the 13th International Conference on Extending Database Technology - EDBT '10*, page 3, 2010. ISBN 9781605589459.

[19] R. McCann, W. Shen, and A. Doan. Matching schemas in online communities: A Web 2.0 approach. In *Proceedings - International Conference on Data Engineering*, pages 110–119, 2008. ISBN 9781424418374.

[20] A. Doan, R. Ramakrishnan, F. Chen, and P. DeRose. Community Information Management. *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases*, 29(1):64–72, 2006.

[21] K. Belhajjame, N. W. Paton, a. a. a. Fernandes, C. Hedeler, and S. M. Embury. User Feedback as a First Class Citizen in Information Integration Systems. *Conference on Innovative Data Systems Research (CIDR '11)*, pages 175–183, 2011.

[22] Y. Raimond, C. Sutton, and M. Sandler. Automatic interlinking of music datasets on the Semantic Web. *CEUR Workshop Proceedings*, 369, 2008.

[23] R. Ramezani. *SWApriori : A New Approach to Mining Association Rules from Semantic Web Data*. PhD thesis, Isfahan University of Technology, 2012.

[24] R. Ramezani, M. Saraee, and M. A. Nematbakhsh. Finding association rules in linked data, a centralization approach. *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, pages 1–6, may 2013.

[25] R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, jul 2012.

[26] G. Friedrich and K. Shchekotykhin. NameIt: Extraction of product names. *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 29–33, 2006.

[27] H. Köpcke, A. Thor, S. Thomas, and E. Rahm. Tailoring entity resolution for matching product offers. *Proceedings of the 15th International Conference on Extending Database Technology - EDBT '12*, page 545, 2012.

[28] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[29] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference, 2004.

[30] T. O'Brien, J. Casey, B. Fox, B. Snyder, J. V. Zyl, and E. Redmond. *Maven: The Definitive Guide*. 2008. ISBN 0596517335.

[31] M. Hepp. GoodRelations: An ontology for describing products and services offers on the web. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5268 LNAI, pages 329–346, 2008. ISBN 3540876952.

Appendix A

Questionnaires

In this section both the questionnaires for Search and Data Curator users are presented.

A.1 Search User Answers

curado.rita@gmail.com

Edit this form

20 responses

Publish analytics

Summary

Informação Pessoal

Idade

23

25

21

24

19

27

54

26

22

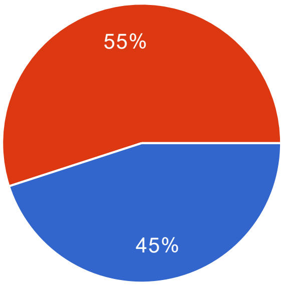
18

56

53

29

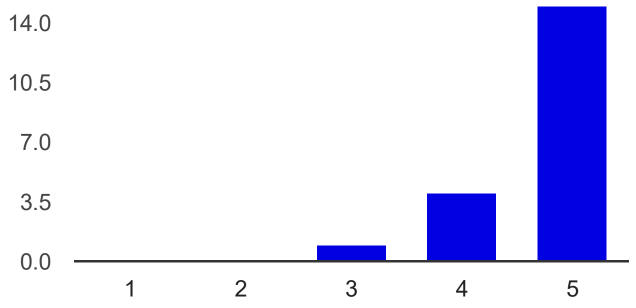
Sexo



Masculino	**9**	45%
Feminino	**11**	55%

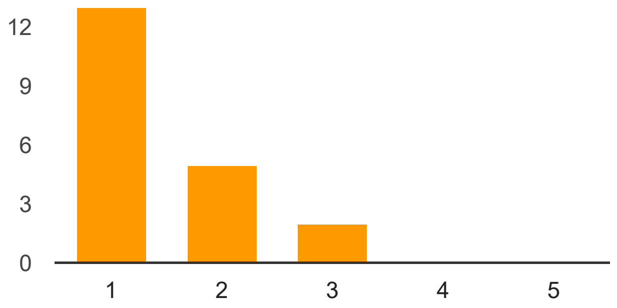
Informação Aplicacional

Com que regularidade faz pesquisas na Internet?



Raramente: 1	**0**	0%
2	**0**	0%
3	**1**	5%
4	**4**	20%
Diariamente: 5	**15**	75%

Com que frequência utiliza a opção de "Pesquisa Avançada" no seu motor de busca?



Nunca: 1	**13**	65%
2	**5**	25%
3	**2**	10%
4	**0**	0%
Sempre: 5	**0**	0%

Que opção de "Pesquisa Avançada" costuma utilizar?

Imagens - tamanho

pesquisa por iumagens

imagem/tamanho grande

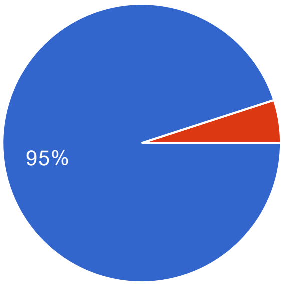
Imagens e Fundo Transparente

imagens tamanho; usage rights

tamanho de imagem

Avaliação da aplicação ASA

A forma como os dados lhe foram apresentados foi suficiente para a realização das tarefas?



Sim	**19**	95%
Não	**1**	5%

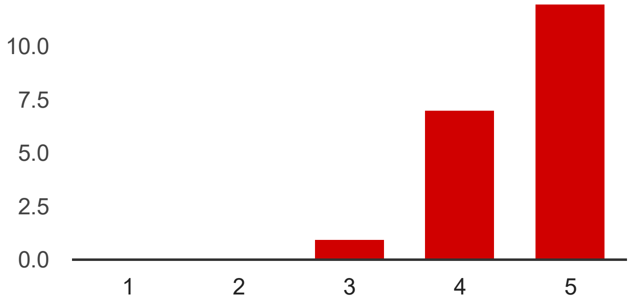
Se não, que informação adicional gostaria de ter tido?

Um indice para saber previamente onde está cada opção. Página informativa (a 2ª) pouco util. Explicação do que consistem as "mapping rules" e o "search criteria".

mais informação direcional

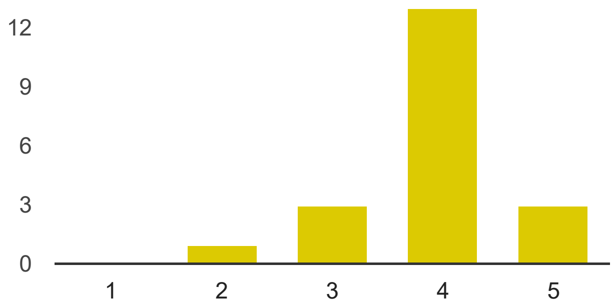
Na sua opinião as tarefas pedidas foram difíceis?

Tarefa 1 - Pesquisa Simples



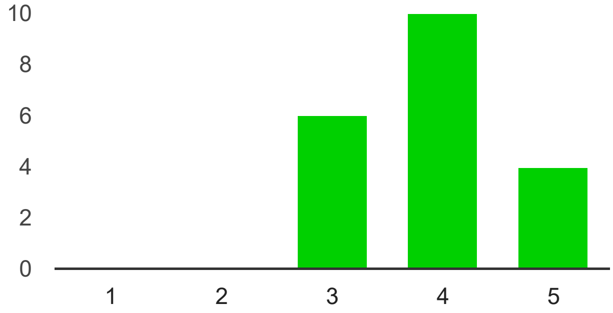
Muito difícil: 1	**0**	0%
2	**0**	0%
3	**1**	5%
4	**7**	35%
Muito fácil: 5	**12**	60%

Tarefa 2 - Mapeamento



Muito difícil: 1	**0**	0%
2	**1**	5%
3	**3**	15%
4	**13**	65%
Muito fácil: 5	**3**	15%

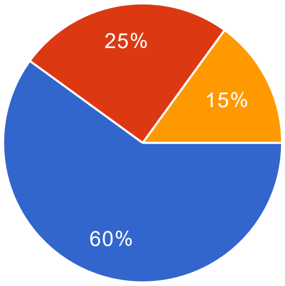
Tarefa 3 - Pesquisa Complexa



Muito difícil: 1	**0**	0%
2	**0**	0%
3	**6**	30%

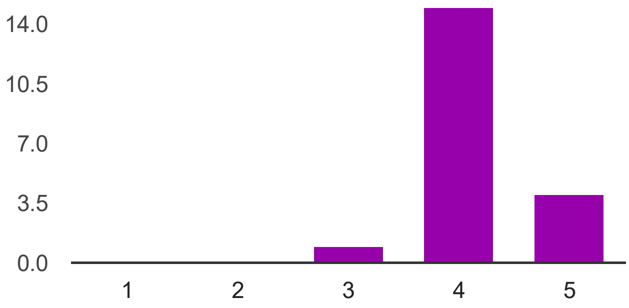
4	**10**	50%
Muito fácil: 5	**4**	20%

Qual foi a tarefa que demorou mais tempo a realizar?



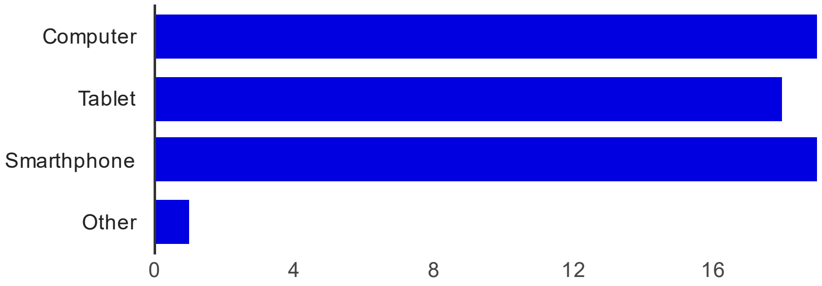
1	**12**	60%
2	**5**	25%
3	**3**	15%

Tendo em conta o objectivo final do sistema, o que pensa da sua utilidade?



Desnecessária: 1	**0**	0%
2	**0**	0%
3	**1**	5%
4	**15**	75%
Essencial: 5	**4**	20%

Em que equipamentos acha que este sistema devería estar disponível?



Computer	**19**	95%
Tablet	**18**	90%
Smarthphone	**19**	95%
Other	**1**	5%

Sugestões

Na sua opinião, quais fontes de dados seríam interessantes explorar com esta ferramenta?

Dados de produtos tecnológicos, calçado, material de desporto, carros

Lojas de material informático, escolar, etc. julgo ser uma ferramenta que se enquadra com qualquer tipo de pesquisa.

Em qualquer produto que não seja unico

material informático. Produtos alimentares

Aplicações relacionadas com venda de produtos alimentares

Produtos de tecnologia, i.e computadores, telemoveis, tablets, etc..

Telemóveis

classificação de mercadorias, classificação de empresas

artigos académicos

Dados sobre desporto

Dados veterinários

Não só os dados dos medicamentos mas também dos médicos e doentes.

Que alterações faría na interface que lhe foi apresentada?

Mostraria um índice das páginas existentes, highlight no campo do valor da pesquisa e apresentação da tabela final (3 tabelas ou tentar evitar colunas repetidas)

Mais explicativa, e mais apelativa visualmente.

Simplificar

No mapeamento a informação pode ser um bocado confusa. E mudar o nome das regras

Na pesquisa complexa a opcao sources deveria explicar que se trata de 3 modos de vista relativamente ao mapeamento realizado

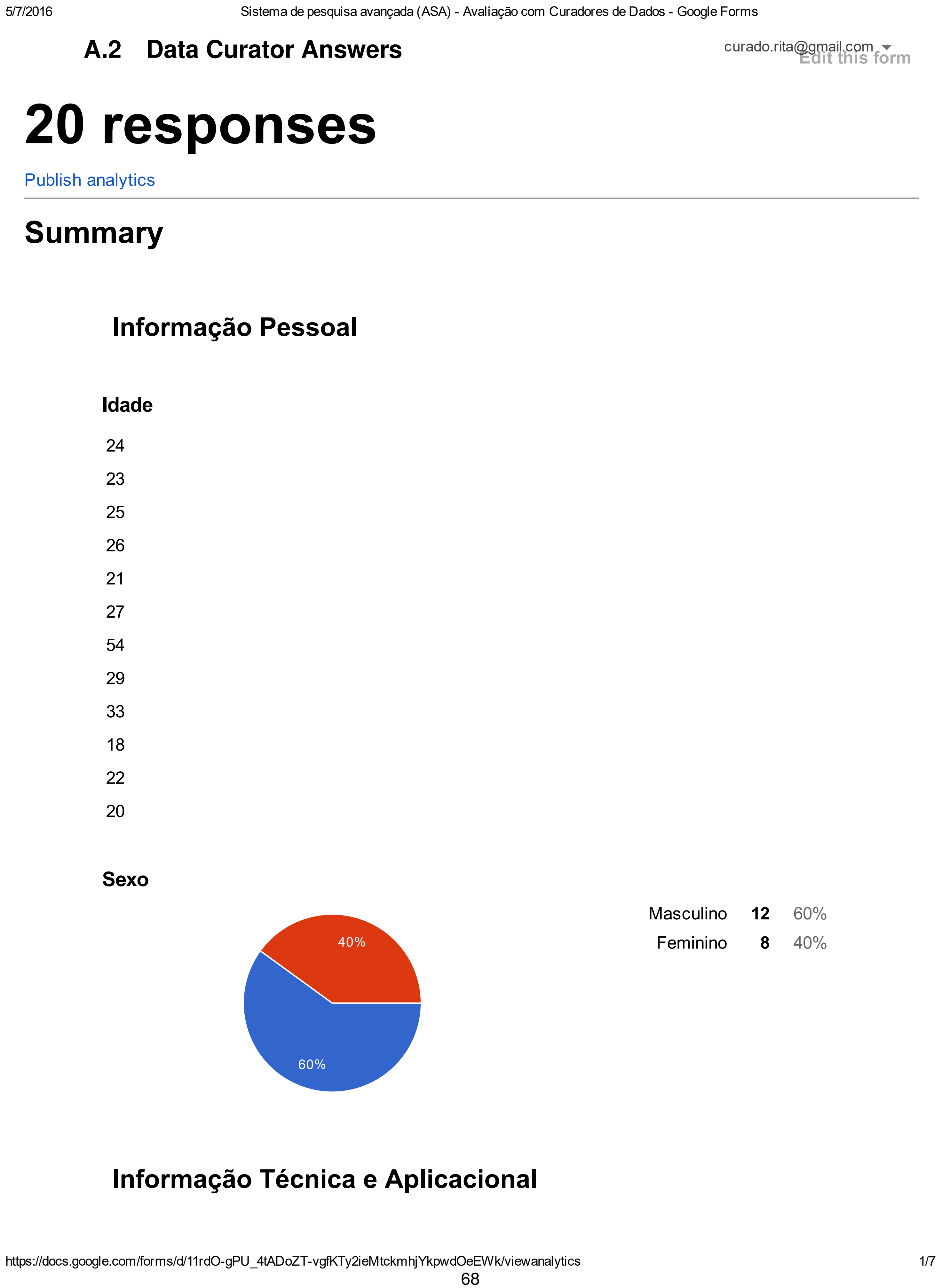
Nomear mais evidentemente as várias páginas

No mapeamento, retirar o botão de confirmação.

apresentação tabelas e os nomes das propriedades mais simplicados

Em vez de um botão para a Label, um hover listener

A.2 Data Curator Answers



20 responses

Summary

Informação Pessoal

Idade

24

23

25

26

21

27

54

29

33

18

22

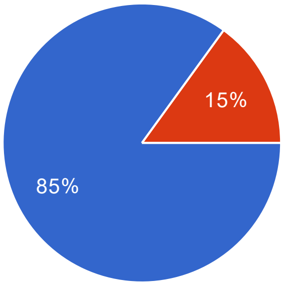
20

Sexo

Masculino	**12**	60%
Feminino	**8**	40%

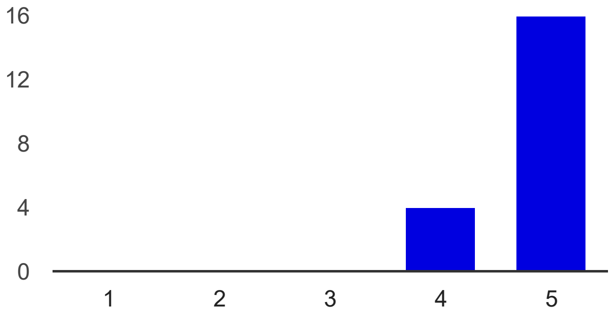
Informação Técnica e Aplicacional

Já trabalhou com algum tipo de sistema de Base de Dados?



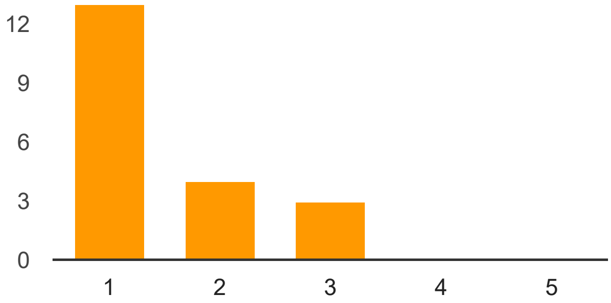
Sim	**17**	85%
Não	**3**	15%

Com que regularidade faz pesquisas na Internet?



Raramente: 1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**4**	20%
Diariamente: 5	**16**	80%

Com que frequência utiliza a opção de "Pesquisa Avançada" no seu motor de busca?



Nunca: 1	**13**	65%
2	**4**	20%
3	**3**	15%
4	**0**	0%
Sempre: 5	**0**	0%

Que opção de "Pesquisa Avançada" costuma utilizar?

Imagens - tamanho

Pesquisa de imagens

Imagem/Tamanho grande

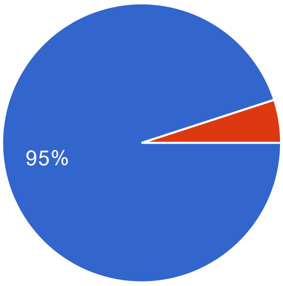
Imagens e Fundo Transparente

imagens tamanho e usage rights

size image

Avaliação da aplicação ASA

A forma como os dados lhe foram apresentados foi suficiente para a realização das tarefas?



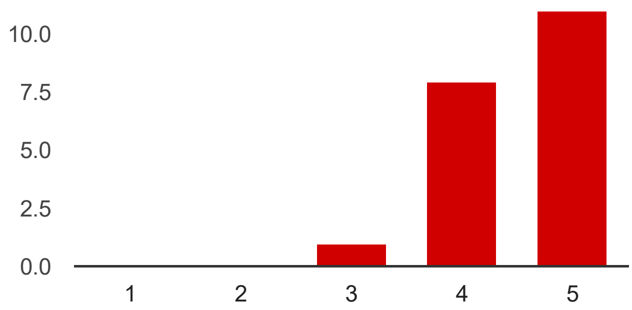
Sim	**19**	95%
Não	**1**	5%

Se não, que informação adicional gostaria de ter tido?

Um indice no inicio, e explicações sobre cada uma das opções.

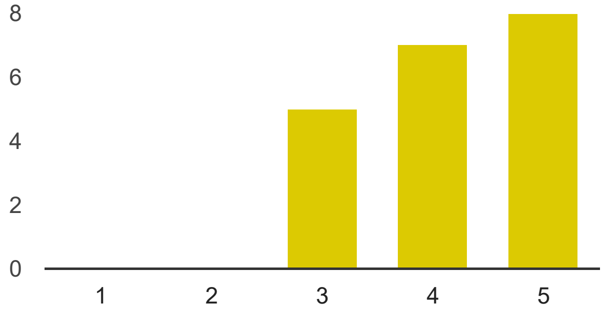
Na sua opinião as tarefas pedidas foram difíceis?

Tarefa 1 - Equivalência de propriedades



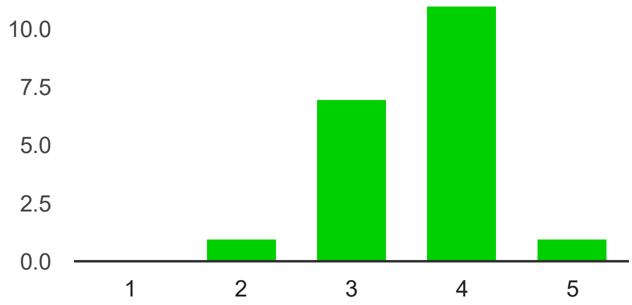
2	**0**	0%
3	**1**	5%
4	**8**	40%
Muito fácil: 5	**11**	55%

Tarefa 2 - Filtragem

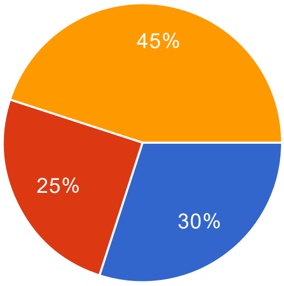


Muito difícil: 1	**0**	0%
2	**0**	0%
3	**5**	25%
4	**7**	35%
Muito fácil: 5	**8**	40%

Tarefa 3 - Mapeamento

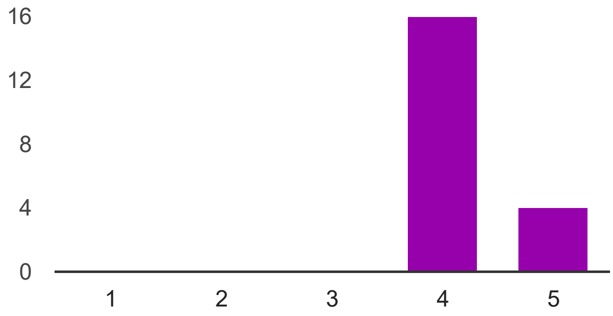


Qual foi a tarefa que demorou mais tempo a realizar?



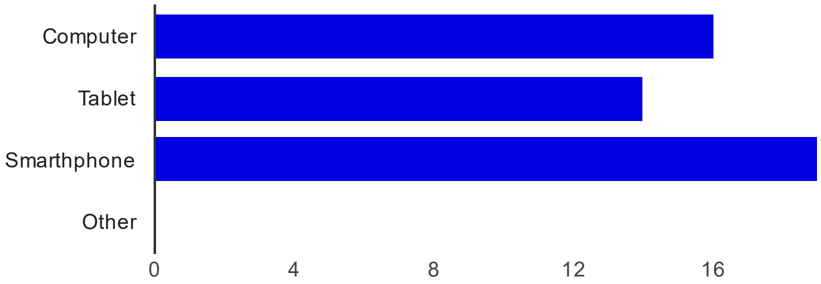
1	**6**	30%
2	**5**	25%
3	**9**	45%

Tendo em conta o objectivo final do sistema, o que pensa da sua utilidade?



Desnecessária: 1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**16**	80%
Essencial: 5	**4**	20%

Em que equipamentos acha que este sistema devería estar disponível?



Computer	**16**	80%
Tablet	**14**	70%
Smarthphone	**19**	95%
Other	**0**	0%

Sugestões

Na sua opinião, quais fontes de dados seríam interessantes explorar com esta ferramenta?

Material tecnológico e de desporto, carros, calçado

Todas as áreas

Em qualquer produto que existam equiparados

material informático, produtos alimentares

Aplicações de produtos desportivos

Produtos de Tecnologia

Telemóveis

classificação de mercadorias e de empresas

artigos académicos

Que alterações faría na interface que lhe foi apresentada?

O botão NEXT e CREATE na página de filtragem

mais apelativa e explicativa.

Simplificar

legendas no mapeamento

Maior identificações de páginas

Na Regra de Filtragem, o botão de create devia estar abaixo da regra.

identificar automaticamente que sources estou a mapear através da regra que crio

Outras sugestões