Dataset interlinking module

Jérôme Euzenat, Nathalie Abadie, Bénédicte Bucher, Zhengjie Fan, Houda Khrouf, Michael Luger, François Scharffe, Raphaël Troncy

Datalift

Un ascenseur pour les données

ANR Contint – ANR-10-CORD-009

D4.2 Dataset interlinking module

Coordinator: Jérôme Euzenat
With contributions from: Nathalie Abadie (IGN), Bénédicte Bucher (IGN), Zhengjie Fan (INRIA), Houda Khrouf (EURECOM), Michael Luger (LIRMM), François Scharffe (LIRMM), Raphaël Troncy (EURECOM)

Quality reviewer:	Serena Villata (INRIA) & Laurent Bihanic (ATOS)
Reference:	Datalift/2011/D4.2/v11
Project:	Datalift ANR Contint ANR-10-CORD-009
Date:	November 8, 2011
Version:	11
State:	final
Destination:	public

Executive Summary

This document describes the first version of the data interlinking module on the Datalift platform.

We have chosen to integrate the Silk tool developed by the Frei Universität Berlin within the platform because it has already been broadly used and this is a maintained and open-source tool. So, we first briefly describe Silk and other components that may be useful within the data interlinking module.

Silk is a script-based tool: one has to write a script in the Silk-SL language telling Silk where and how to find links in the datasets. We present a script that can be used within the Datalift platform. We describe briefly how we have built the module around Silk and how Silk can take its data from the triple stores of the Datalift platform and how it can output the generated links in the platform's triple store as well. Of course, the module can still take its data out of the platform.

We also explain how we plan to extend this module so that it can go beyond Silk. This may be achieved in two different and complementary ways:

- by integrating modules within Silk that provides new interlinking methods or new ways to interlink;
- by automatically generating Silk scripts from our analysis to the data to be interlinked.

So, we describe two additional Silk modules that can be defined for matching time (events) and geographical data. We also explain how, from ontology alignment expressed in the expressive language EDOAL, it can be possible to generate links or Silk scripts that take advantage of the content of alignments. This is what we will concentrate on, in the first part of the second year.

Document Information

ANR Project Number	ANR Contint – ANR-10-CORD-009	**Acronym**	Datalift
Full Title	Un ascenseur pour les donnes		
Project URL	`http://www.datalift.org/`		
Document URL			

Deliverable	**Number**	4.2	**Title**	Dataset interlinking module
Work Package	**Number**	4	**Title**	Data interlinking

Date of Delivery	**Contractual**	M12	**Actual**	28-09-2011
Status	final		final ☒	
Nature	prototype ☒ report ☐ dissemination ☐			
Dissemination level	public ☒ consortium ☐			

Authors (Partner)	Nathalie Abadie (IGN), Bénédicte Bucher (IGN), Zhengjie Fan (INRIA), Houda Khrouf (EURECOM), Michael Luger (LIRMM), François Scharffe (LIRMM), Raphaël Troncy (EURECOM)			
Resp. Author	**Name**	Jérôme Euzenat	**E-mail**	Jerome.Euzenat@inria.fr
	Partner	INRIA		

Abstract (for dissemination)	This report presents the first version of the interlinking module for the Datalift platform as well as strategies for future developments.
Keywords	data interlinking, linked data, instance matching

Version Log			
Issue Date	**Rev No.**	**Author**	**Change**
20/09/2011	1	J. Euzenat	Initial entry with existing data
21/09/2011	2	J. Euzenat	Added info about Michael Luger's platform
24/09/2011	3	J. Euzenat	Added extended abstract
26/09/2011	4	Z. Fan	Described actual implementation
29/09/2011	5	B. Bucher	Added geographical part
29/09/2011	6	Z. Fan	Cleaned up biblio
30/09/2011	7	J. Euzenat	Intro/conclusion
01/10/2011	8	J. Euzenat	Overall revision
03/10/2011	9	R. Troncy	Added additional measures integrated into Silk
03/10/2011	10	J. Euzenat	Final version (more details in §4)
04/11/2011	11	Z. Fan	Taken quality control comments into account

Table of Contents

1. Introduction

This deliverable presents the design of the interlinking module of the Datalift platform.

There are lots of isolated RDF data sets being published everyday. A web of linked RDF data set is highly required for web users to share information with each other. For that purpose, the dataset published from the Datalift platform must be linked to other, likely external, datasets. It is the purpose of the interlinking module to propose this feature.

We decided to provide more context to the actual prototype by considering tools that could be used for the module as well as peripheral tools. We also provide information about how this module is supposed to develop in the future through extensions.

Hence, this deliverable discusses existing tools that we may reuse in the platform (§2). Existing linking algorithms are, for their part, the object of Deliverable D4.1. We then briefly motivate the choice that has been made for the implementation of the module and give a general view of how this implementation will be developed (§3). We describe the current implementation and its capabilities (§4). Finally we present the blueprint for future work: implementing specialised Silk plug-in that may be used with the dataset considered in the Datalift project (§5) and discussing how to use alignments for generating Silk specifications (§6).

2. Context

In this chapter, we describe in some details existing tools that may be used in the platform or with the platform. This description may skip and the reader may come back to it if he or she needs more contextual information.

We describe mainly two tools: Silk (§2.1.1), a system for establishing links among data sources, and MDR (§2.2) a system for manipulating data sources which may ultimately be used for generating Silk scripts (manually). Since MDR may use an alignment server, we briefly present one (§2.3). Even if MDR is not used within the platform, Alignment servers may be used to retrieve alignments prior to interlinking data.

2.1 Silk

Silk[1], or the Silk Link Discovery Engine, is also described and compared to other tools in Deliverable D4.1. It is an interlinking software that helps to find out similar items in RDF data sets. It accepts RDF data sets both in file and in SPARQL endpoint. After interlinking computation, it produces RDF link set both in file and in SPARQL endpoint.

In order to fulfill the interlinking computation, Silk needs the specification of several information. They are,

- the concepts from which to find the similar items,
- the properties of the concepts upon which to match the similar items,
- the techniques that are used to match the properties' values, and
- the threshold of linking or not.

All these information are specified in a link specification file. Silk will run such file to produce link sets between different data sets.

Silk provides different variants for their functions. They are,

- Silk Single Machine, which is used to produce RDF links on a single machine,
- Silk MapReduce, which is used to deploy Silk on groups of machines and
- Silk Server, which acts as an identity resolution module in any application background that checks whether there are new links arising when new RDF data sets are coming.

Silk also provides a workbench to produce the link specification file.

In what follows, we embedded Silk Single Machine in the Datalift interlinking module. However, we may use the other settings.

2.1.1 Silk-LSL: a linking specification language

Below is a Silk-LSL specification to interlink cities in the two data sets DBpedia (http://dbpedia.org/ontology/ and Geonames (http://www.geonames.org/ontology#) [4]:

```
<Silk>

  <Prefix id="rdfs" namespace=
          "http://www.w3.org/2000/01/rdf-schema#" />
  <Prefix id="dbpedia" namespace=
          "http://dbpedia.org/ontology/" />
  <Prefix id="gn" namespace=
```

[1] http://www4.wiwiss.fu-berlin.de/bizer/silk/

```
          "http://www.geonames.org/ontology#" />

  <DataSource id="dbpedia">
    <EndpointURI>http://demo_sparql_server1/sparql
    </EndpointURI>
    <Graph>http://dbpedia.org</Graph>
  </DataSource>

  <DataSource id="geonames">
    <EndpointURI>http://demo_sparql_server2/sparql
    </EndpointURI>
    <Graph>http://sws.geonames.org/</Graph>
  </DataSource>

  <Interlink id="cities">
    <LinkType>owl:sameAs</LinkType>

    <SourceDataset dataSource="dbpedia" var="a">
      <RestrictTo>
         ?a rdf:type dbpedia:City
      </RestrictTo>
    </SourceDataset>

    <TargetDataset dataSource="geonames" var="b">
      <RestrictTo>
        ?b rdf:type gn:P
      </RestrictTo>
    </TargetDataset>

    <LinkCondition>
      <AVG>
        <Compare metric="jaroSimilarity">
          <Param name="str1" path="?a/rdfs:label" />
          <Param name="str2" path="?b/gn:name" />
        </Compare>
        <Compare metric="numSimilarity">
          <Param name="num1"
                 path="?a/dbpedia:populationTotal" />
          <Param name="num2" path="?b/gn:population" />
        </Compare>
      </AVG>
    </LinkCondition>

    <Thresholds accept="0.9" verify="0.7" />
    <Output acceptedLinks="accepted_links.n3"
      verifyLinks="verify_links.n3"
      mode="truncate" />
  </Interlink>

</Silk>
```

This specification fulfills two roles:

- It is an alignment: it specifies the classes in which entities to link can be found. Restrictions to `dbpedia:City` and `gn:P` are in fact an alignment between these two concepts. Similarly, the compared properties `dbpedia:populationTotal` and `gn:population` and `rdfs:label` and `gn:name`, respectively provide the correspondences between properties.
- It specifies how to link entities. Indeed, what Silk brings in addition to an alignment is the specification of how to decide if two entities should be linked: when the average

Figure 2.1: The Alignment view of DMT.

(`AVG`) of their respective distances (`Compare`) is over a threshold (`Threshold`, there are two thresholds, one for accepting automatically the equivalence and one for drawing the attention of a user).

2.2 Data management tool

The Data management tool of Michael Luger (U. Innsbruck) is a tool for navigating and interlinking datasets graphically. It allows for:

- interactively displaying pairs of datasets and select properties to compare;
- using ontology alignments from an Alignment server to identify properties to compare;
- analysing the datasets in order to establish keys;
- matching datasets (work in progress).

This tool could be used in parallel of the platform in order to analyse datasets or it could be integrated within the platform.

The Alignment view (see Figure 2.1) displays all the classes and properties of the two datasets and offers features to deal with schema alignments and dataset analysis. Its purpose is the discovery of information about the datasets, in particular possible schema-level correspondences which are suitable restrictions for the Matcher, i.e., candidate selection.

There are two reasons for creating an Alignment view separate from the Dataset Explorer:

- The selection of resources serves a different purpose, so it will not trigger the dataset filtering SPARQL queries as in the Dataset Explorer.
- The features that are part of this view would clutter the interface too much, when combined with the browsing feature.

Upon activating this view, an Alignment server [7] is checked for schema alignments, and the schema alignments are then displayed both inline and in the bottom left corner (each pair of table cell's color increases the hue of a base color). These alignments can be selected for restricting the input of the Matcher view.

When selecting two classes or two properties from each of the two schema-views, they will show up at the bottom of the view. They can be stored on the Alignment Server or directly used as restriction input for the Matcher view.

There is an option to calculate the discriminability and coverage factors for each property, and results will be shown inline. The algorithms are adapted from [24].

Property Correspondences Discovery, i.e., candidate selection, allows for finding schema-level correspondences between properties of the two datasets by investigating instance-level information: Taking for example the sider and dailymed datasets, suppose we perform string-based matching between the object values for all possible property combinations between the two datasets, there would be a total of approximately 2 000 000 000 different comparisons.

The number of comparisons can be reduced massively while still maintaining totally useful results (in comparison) as follows:

- only match literals.
- only match literals with the same datatype (detect datatypes of type integer, double, uri, xml, string, description (100+ character strings)) - also according to the datatype, suitable matching methods can applied.
- only match properties that fulfill certain discriminability and coverage criteria.
- for each remaining property-pair, only match a subset of the property-values from one dataset with all the property-values of the other dataset. This means, for each property-pair, we put for example the constraint LIMIT 10 for the property-values of the one dataset's selected property and match each of those 10 property-values with all the property-values of the second dataset's selected property. Then for each of the 10 property-values, we remember the highest score and calculate the average over each of those 10 highest scores. Suppose a value range between 0.0 and 1.0 for such a score: when comparing the results of using LIMIT 10 with LIMIT 100, with the above dataset there was a maximum difference of 0.07 between the scores for each property-pair (except one outlier).

This means, that we can effectively reduce the number of comparisons and still obtain very useful results: Datatype detection reduces the number of comparisons from 2 000 000 000 to 150 000 000, and the LIMIT constraint to 750 000 for the above pair of datasets. Taking into account the property coverage and discriminability factors obtain an even higher reduction, but such constraints have to be used properly. Indexing could further improve performance.

For scoring those property-similarities, one option would be to count the percentage of exact or near-exact matches, another to calculate the average over the highest scores for each value from the LIMIT-constrained dataset. In addition to displaying those scores in the user interface after triggering this method, the best matches between the property-values could be displayed (perhaps a pop-up for each of the e.g. 10 matches or a selection) to allow the user to investigate suitable candidates.

This candidate selection feature could be further improved by using other schema alignment methods such as also comparing the classes- and properties-titles and looking into the structure of the schemas.

The matcher view is not available yet. Basically, the idea for the overall workflow is that property restrictions for the matcher can be selected from the Dataset Explorer and the Alignments views:

- Properties can be selected as restrictions for the matcher.
- Classes can be selected as restrictions, and consequently all the properties under the classes constraint are used as input for the matcher.

– Properties can be constrained by their object-values and those constrained properties serve as input for the matcher.

This needs to be defined in more detail, including how to best handle the picking of union and intersection of properties.

Then, the user can specify the matching method and weights for each of the properties. Again, some features like matching datatypes as well as coverage and discriminability can be displayed to assist the user.

During or after the matching phase, matched instances along with all their associated classes, properties, and property-values can be displayed to the user who can then supply feedback.

It would also be possible for the matching view to generate Silk specifications instead of actually performing the match.

2.3 Alignment server

Alignment servers are independent software components which offer a library of matching methods and an alignment store that can be used by their clients. MDR uses an Alignment server for retrieving alignments in order to help linking data. The Datalift platform may take advantage of them for the same purpose.

In a minimal configuration, alignment servers contribute to storing and communicating alignments. Ideally, they can offer all the following services:

Matching two ontologies possibly by specifying the algorithm to use and its parameters (including an initial alignment).
Storing an alignment in persistent storage.
Retrieving an alignment from its identifier.
Retrieving alignment metadata from its identifier can be used for choosing between specific alignments.
Finding (stored) alignments between two specific ontologies.
Comparing alignments for determining what their differences are.
Editing an alignment by adding or discarding correspondences (this is typically the result of a graphic editing session).
Trimming alignments, i.e., selecting correspondences within thresholds.
Generating code implementing ontology transformations, data translations or bridge axioms from a particular alignment.
Translating a message with regard to an alignment.
Finding a similar ontology is useful when one wants to align two ontologies through an intermediate one.

All these services are potentially useful for Datalift if users want to match ontologies and manipulate alignments before interlinking data. However, the main services to be used in our case is the storing and retrieving service. It will allow to provide already stored alignments to the platform.

For instance, someone wanting to interlink a data set using ontology o with a data set using an ontology o'' can ask for matching the two ontologies and using the alignment for generating the Silk script. A more complete scenario involves (1) asking for alignments between o and o'', maybe resulting in no alignment, (2) asking for an ontology close to o'' which may result in ontology o', (3) asking for the alignments between o and o', which may return several alignments a, a' and a'', (4) asking for the metadata of these alignments and

(5) choosing a' because it is certified by a trusted authority, (6) matching o' and o'' with a particular algorithm, (7) suppressing the resulting correspondences whose confidence is under a reasonable threshold for this algorithm, (8) editing the results so that it is correct, (9) storing it in the server for sharing it with other parties, (10) retrieving alignment a' and this latter one, (11) finally using these two alignments in order to generate a script to interlink the two data sources.

Alignment servers are middleware components of the semantic web infrastructure. They can be used by semantic web applications as well as by the infrastructure itself. They can be used at design time or at run time. For Datalift, the Alignment server is to be used when platform users need alignments for generating links.

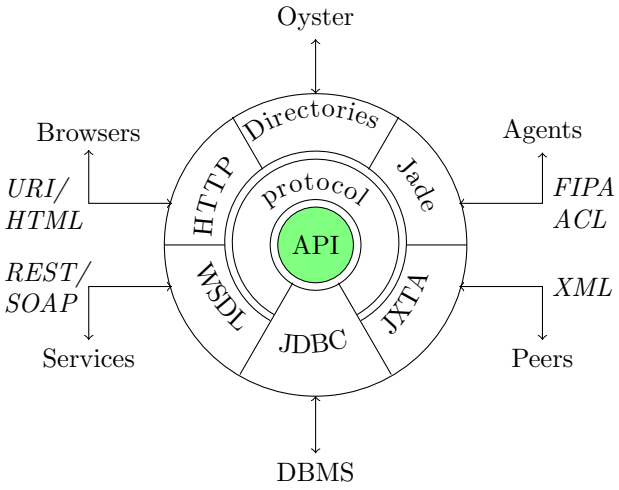

Figure 2.2: The Alignment server is built on the Alignment API that is seated on top of a relational database for sharing alignments and is wrapped around a simple protocol. Each access method is a plug-in that interacts with the server through the protocol. Currently, HTML, agent and web service plug-ins are available.

The Alignment server architecture is made of four layers (shown in Figure 2.2):

The Alignment API which provides operations for manipulating alignments including extension points for adding new matchers and renderers.

A storage system which offers persistent storage and retrieval of alignments. It implements only basic storage and runtime memory caching functions. The storage is made through a DBMS interface and can be replaced by any database management system as soon as it is supported by JDBC.

A protocol manager which handles the server protocol. It accepts queries from plug-in interfaces and uses the server resources for answering them. It uses the storage system for caching results.

Protocol drivers (WSDL, HTTP, Jade, JXTA) which accept incoming queries in a particular communication system and invoke the protocol manager in order to answer them. These drivers are ideally stateless and only translators for the external queries.

2.4 Conclusion

We have presented tools that we may use in the Datalift data interlinking module. We will consider how to use them in the next chapter.

3. Implementation choices

As Deliverable D4.1 shows, there are many approaches to develop an interlinking module. We have chosen to build our first prototype using Silk as a linking engine instead of writing a new one from scratch. We motivate below this choice. This should allow us to experiment with data interlinking within the platform.

If Silk appears to be too limited in the end, we may reconsider this choice and develop another module.

3.1 Why Silk?

We decided to experiment with Silk for several reasons:

- It is available in an appropriate license (compatible with inclusion in the Datalift platform);
- It is maintained;
- It is supposed to be quite extensible;
- It is popular: independent projects have used it and it is heavily used in the LATC project.

However, Silk does come with a few draw backs:

- It is not that extensible: introducing modifications (like using alignments) requires major changes in the software;
- It is script based, so any extension must have been considered in the scripting language;
- It is written in Scala, which at that point of mastery, can be considered an obstacle.

The main problem is that Silk follows a specific and rather strict framework and does not provide an API for bypassing this framework.

We are in contact with Silk developers and discuss with them about the opportunity to further open their software. However, the two development tracks below are independent from this evolution.

3.2 Building an interlinking module around Silk

In Silk, the user has to write a script telling the system how to find links: between which classes, with which methods and which thresholds. On the other side, in Datalift, we want to define a system to find links without such an input.

A first way to use Silk under the platform is to develop the module so that it finds the parameters needed by Silk and generates (see Figure 3.1). One particular implementation of this approach will be the object of §6.

Another way of working consists of extending Silk with particular Datalift capabilities and to run Silk so that it uses such capabilities (see Figure 3.2). Such an extension may be quite simple such as developing measures for some particular data (see §5) or using alignments from inside Silk for piloting the script execution process. The second solution is difficult due to Silk relative closedness.

Both solutions are viable and we plan to pursue them together.

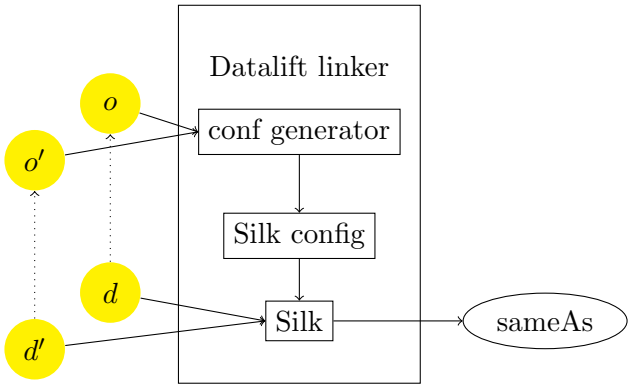

Figure 3.1: Developing a tool for generating Silk scripts.

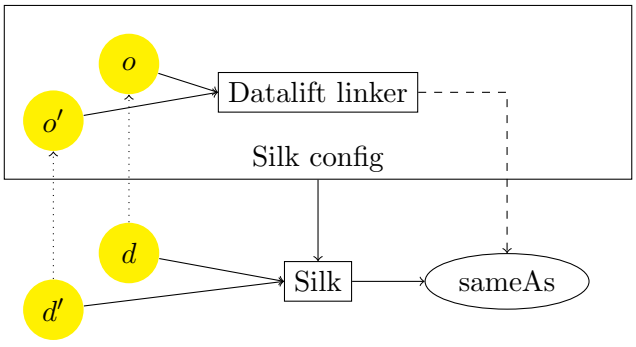

Figure 3.2: Integrating a linker directly within Silk.

3.3 Conclusion

We have justified the decision of embedding Silk in the Datalift interlinking module and presented ways to extend it. We will now describe the currently implemented module.

4. Current prototype description

The current prototype integrates Silk within the Datalift platform. It produces `owl:sameAs` links from the data in the platform or elsewhere and store them in the public repository.

The module uses the last version of Silk (2.4.2). It is available in the Datalift code repository since version f1c94ccb681499b5dc5488de1dfbc2d01fac3550.

4.1 Principles

A first version of the Data interlinking module has been developed embedding Silk.

The current interlinking module is able to:

- Load its (RDF) data from the platform triple store;
- Load its Silk script from the platform environment;
- Run the Silk script;
- Store the (RDF) results in the platform triple store.

This first version would be fully functional for integrating data if it was possible to upload the Silk script on the platform. This is our next immediate goal. Further work is presented in conclusion.

We have currently used Silk as a standalone server but it can also be used as a web service or deployed to the cloud. It is even possible that the Silk developers offer online services that could be used by the Datalift platform.

4.2 A Silk script for linking INSEE and NUTS

Here is the Silk script for linking geographical RDF data in France from INSEE and Europe from Eurostat.

```
<?xml version="1.0" encoding="utf-8" ?>
<Silk>
    <Prefixes>
        <Prefix id="rdf" namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
        <Prefix id="rdfs" namespace="http://www.w3.org/2000/01/rdf-schema#" />
        <Prefix id="geo" namespace="http://rdf.insee.fr/geo/" />
        <Prefix id="dc" namespace="http://purl.org/dc/elements/1.1/" />
        <Prefix id="cc" namespace="http://creativecommons.org/ns#" />
        <Prefix id="owl" namespace="http://www.w3.org/2002/07/owl#"/>
        <Prefix id="dcterms" namespace="http://purl.org/dc/terms/" />
        <Prefix id="xmlns" namespace=
                "http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#"/>
        <Prefix id="insee" namespace="http://rdf.insee.fr/geo/" />
        <Prefix id="eurostat" namespace=
                "http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#"/>
    </Prefixes>

    <DataSources>
        <DataSource id="insee" type="sparqlEndpoint">
            <Param name="endpointURI" value="http://localhost:8080/datalift/sparql" />
        </DataSource>

        <DataSource id="eurostat" type="sparqlEndpoint">
            <Param name="endpointURI" value="http://localhost:8080/datalift/sparql" />
        </DataSource>
```

```
    </DataSources>

    <Interlinks>
        <Interlink id="region">
            <LinkType>owl:sameAs</LinkType>

            <SourceDataset dataSource="insee" var="a">
                <RestrictTo>
                    { { ?a rdf:type insee:Region } UNION
                    { ?a rdf:type insee:Departement } UNION
                    { ?a rdf:type insee:Commune } }
                </RestrictTo>
            </SourceDataset>

            <TargetDataset dataSource="eurostat" var="b">
                <RestrictTo>
                    ?b rdf:type eurostat:NUTSRegion
                </RestrictTo>
            </TargetDataset>

            <LinkCondition>
                <Aggregate type="max">
                    <Compare metric="levenshteinDistance" threshold="0">
                        <TransformInput function="lowerCase">
                            <Input path="?a/geo:nom[@lang='fr']" />
                        </TransformInput>
                        <TransformInput function="lowerCase">
                            <Input path="?b/eurostat:name" />
                        </TransformInput>
                    </Compare>
                </Aggregate>
            </LinkCondition>

            <Filter />

            <Outputs>
                <Output maxConfidence="1.0" type="file" >
                  <Param name="file" value="insee_eurostat_verify_links.xml"/>
                  <Param name="format" value="alignment"/>
                </Output>
                <Output minConfidence="0.1" type="file">
                  <Param name="file" value="insee_eurostat_accepted_links.xml"/>
                  <Param name="format" value="alignment"/>
                </Output>
            </Outputs>
        </Interlink>
    </Interlinks>
</Silk>
```

In this Silk script, the RDF data sources are queried in the Datalift SPARQL endpoint. After the linking computation, the result is output as local files. They are, "insee_eurostat_verify_links.xml" and "insee_eurostat_accepted_links.xml". Currently, the Datalift external SPARQL endpoint does not support SPARQL Update. Thus, the link data set could not be stored directly to Datalift repository by specifying a SPARQL endpoint in the script. The Datalift interlinking module, thus reads the output files and store their results in the Datalift triple store using the Sesame API.

The results can be seen in the Datalift triple store as in Figure 4.1.

When looking at the result, we can see that the script is not optimal: it has matched twice FR101 ("Paris"), once with Paris the city (COM_75056) and once with Paris the department

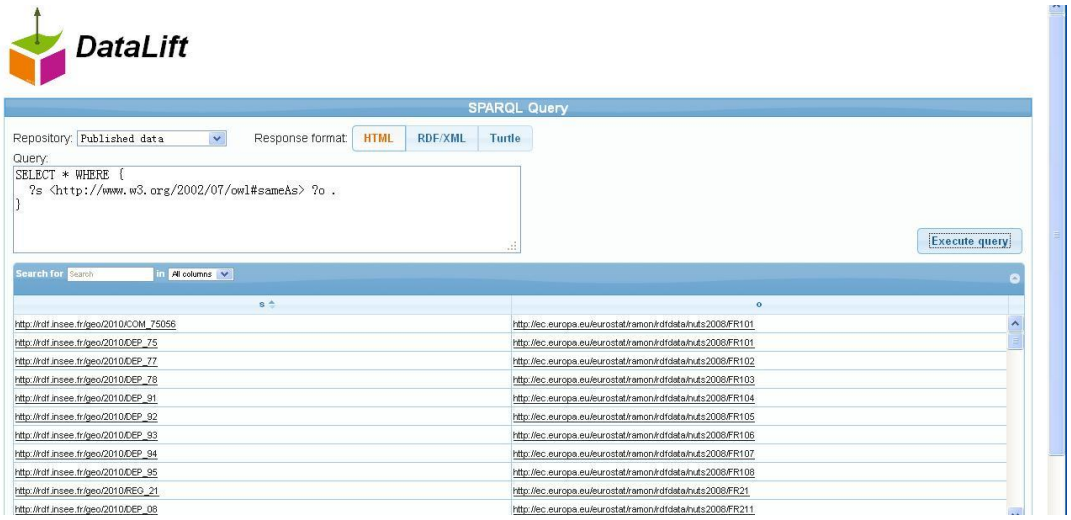

Figure 4.1: The sameAs links found by the Silk script in the Datalift triple store.

(DEP_75). In fact, it is also not efficient because wards ("communes") are not in the NUTS database and only French regions are matchable to the INSEE database. Hence, it is better to write two different pieces of scripts replacing the `RestrictTo` parts by:

```
<SourceDataset dataSource="insee" var="a">
  <RestrictTo>
    ?a rdf:type insee:Region .
  </RestrictTo>
</SourceDataset>

<TargetDataset dataSource="eurostat" var="b">
  <RestrictTo>
   { ?b rdf:type eurostat:NUTSRegion .
      ?b rdf:level 2^^xsd:int .
      ?b eurostat:hasParentRegion ?y .
      ?y eurostat:hasParentRegion &lt;http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/FR&gt; . }
  </RestrictTo>
</TargetDataset>
```

and

```
<SourceDataset dataSource="insee" var="a">
                <RestrictTo>
                    ?a rdf:type insee:Departement .
                </RestrictTo>
</SourceDataset>

<TargetDataset dataSource="eurostat" var="b">
  <RestrictTo>
    { ?b rdf:type eurostat:NUTSRegion .
      ?b rdf:level 3^^xsd:int .
      ?b eurostat:hasParentRegion ?y .
      ?y eurostat:hasParentRegion ?z .
      ?z eurostat:hasParentRegion &lt;http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/FR&gt; . }
  </RestrictTo>
</TargetDataset>
```

The first script section shows the instances from `insee:Region` should be compared with instances from `eurostat:NUTSRegion` with `rdf:level` equals 2. Meanwhile, the parent region of

those instances should be `http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/FR`. Similarly, the second script section shows the instances from `insee:Departement` should be compared with instances from `eurostat:NUTSRegion` with `rdf:level` equals 3. Their grandparent region should also be `http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/FR`.

4.3 Conclusion

We have briefly introduced the current implementation of the Datalift interlinking module which is available from the platform repository.

We will consider, in the two next chapters, the way we plan to further improve this module.

5. Additional components

When dealing with specialised types of data, more informed comparisons may be used. For example, initial experiments performed when trying to align event descriptions help us to deduce that the existing similarity metrics are not sufficiently adapted to meet our needs.

We hereafter expose our motivation behind these extensions namely: temporal inclusion (§5.1), token-wise metrics (§5.2) and space (§5.3), and we detail their algorithms [17].

These extension can be embedded in SIlk, and hence in the Datalift platform, as new similarity measures tied to some data types (temporal and geographic).

5.1 Temporal inclusion metric

Most of the ontologies aiming at representing events, such as the LODE[1] ontology, consider that an event has a start date and might have an end date. Intuitively, we consider that two events are similar if they share, among other things, the same time or temporal interval. Hence, matching two events requires a temporal overlap detection. The Date Time metric provided by Silk only measures the difference between two dates which is normalized against a predefined threshold `MaxDays`. Indeed, the more the difference between two dates is inferior to `MaxDays`, the higher is the similarity score. This metric brings confusion on what convenient threshold `MaxDays` must be specified especially when dealing with over thousands of events. Moreover, it is not suitable to infer if two events share a common temporal interval or not. For instance, we manually detect a similarity between two events, coming from two different social media directories, which have a common label `Radiohead en Chile`, but the first event is given between `26th at 07:49:01 PM and 27th at 18:05:01 PM of March 2009`, while the second one is given on `27th March 2009 at 9:00 PM`. We hence propose to introduce a new temporal inclusion metric which detects if a specific date belongs to one temporal interval or if two temporal intervals have an overlap. In the aforementioned example, we observe a difference of nearly 3 hours between the end date of the former event and the start date of the latter one. Thus, a certain number of hours θ has to be tolerated when comparing two dates.

Given two events (e_1, e_2) which have respectively the pair start date and end date (d_1, d'_1) and (d_2, d'_2) such that $d_1, d_2 \neq 0$ and d'_1, d'_2 can be null, the temporal inclusion similarity is defined by the following function:

$$s(e_1, e_2) = \begin{cases} 1 & \text{if } |d_1 - d_2| \leq \theta \text{ where } (d'_1, d'_2) = 0 \\ 1 & \text{if } d_1 \pm \theta \in [d'_1, d'_2] \text{ where } d_2 = 0 \text{ (idem for } d'_1) \\ 1 & \text{if } min(d'_1, d'_2) - max(d_1, d_2) \geq 0 \text{ where } (d'_1, d'_2) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

5.2 Token-wise string similarity

Comparisons based on string similarity have been widely exploited by many applications such as duplicate detection and data integration. Overall, they are divided into two main classes: character-based and token-based functions, in which the similarity score ranges from 0 to 1. The former class considers a string as a sequence of characters to compute the score. A typical example is the edit distance which is the minimum number of edit operations, i.e., insertion, deletion, and substitution, to convert one string to another. One advanced

[1] `http://linkedevents.org/ontology/`

variant which is not based on the edit distance model is the Jaro metric [15], in which the distance depends on the number and the order of common characters between two strings. For example, the Jaro similarity between `"finn brother"` and `"brothers finn"` is `0.6`. We note that this score is low despite the high similarity between these strings which have similar words. Indeed, changing the order of tokens affects the sequence of characters which makes the character-based metrics sensitive to the tokens positions. The Jaro metric is mainly used to overcome misspelled data and to match short strings such as personal names [5]. For example, the Jaro distance between `"brother"` and `"brothers"` is `0.98`. The second class of string distances is the token-based distance which considers a string as a sequence of tokens. They split strings into token sets (i.e. by white-space) called bag of words, and they compute the similarity based on these sets. A typical metric is the Jaccard distance [14] where the score is the ratio of intersection size and the union size of two token sets. For example, the token sets of `"finn brother"` and `"brothers finn"` are respectively {`"finn"`, `"brother"`} and {`"brothers"`, `"finn"`}. The Jaccard distance between these two strings is 0.3 which is a low score as the close similarity between `"brother"` and `"brothers"` is ignored. Indeed, token-based functions only consider the exact overlap between the token sets and neglect approximate tokens.

We have studied the performance of these string similarity functions over the Event-Media data that consists of user-generated content featuring typos, misspelled and noisy data [18]. We find some similar events wherein the labels have a low similarity in terms of character-based and token-based functions. For example, the Jaccard score between `"Anna Plus Kristen Puferfish"` and `"Pufferfish for Anna and Kristen"` is 0.28 while the Jaro score is 0.6. To overcome these limitations, we introduce a novel hybrid metric called Token-Wise [17] combining the two string similarity classes. Our objective is to compute a fuzzy overlap between two token sets. We enhance our method by allowing to attribute a low weight for a list of words (stop-words) frequently used such as the prepositions `and`, `for`, `in`, `from`, etc. The Token-Wise metric proceeds as follows:

- The strings s and t are first split into a set of tokens $\{s_1....s_k\}$ and $\{t_1....t_p\}$
- A list L_{sp} of stop-words and related weight can be specified. These stop-words are in general frequent words which differ from one context to another. For instance, analyzing the events titles in EventMedia help us to find out some frequent words such as festival, concert, music. etc. We can also attribute a specific weight for the remaining words which do not belong to L_{sp}.
- Using one of the character-based functions such as Jaro, Jaro-winkler and Levenshtein, the similarity scores are computed between each token pair from the two sets generating a set of triples $(s_i, t_j, sim'(s_i, t_j))$. We can keep the triples, in which the scores are larger than a given sim' threshold δ. Then, for each token, we keep its correspondent triple which contains the highest score.
- The token-wise score is based on the Jaccard similarity pattern which computes the ratio of union size and intersection size of two token sets. Given two strings s, t and their respective token sets A and B, the token-wise score adopts the following pattern

$$pattern(s,t) = \frac{|A \cap B|}{|A \backslash B| + |B \backslash A| + |A \cap B|} \tag{5.2}$$

Given $sim'(s_i, t_j)$ the score of the based-character similarity between two tokens, ws_i and wt_j the respective weights of s_i and t_j, N the number of matched tokens (filtered triples), M

the number of unmatched tokens where we set $sim' = 0$, the token-wise score can be written as

(5.3)
$$token - wise(s,t) = \frac{\sum_{i=1}^{N} sim'(s_i,t_j) \times ws_i \times wt_j}{\sum_{i=1}^{N+M}(1 - sim'(s_i,t_j)) \times (ws_i^2 + wt_j^2) + \sum_{i=1}^{N} sim'(s_i,t_j) \times ws_i \times wt_j}$$

To better understand the token-wise measure, we consider the following two event titles `"Island Treasre Music"` and `"Treasure Island"`. We compute the Jaro similarity between each token pair from their respective sets (`"Island"`, `"Treasre"`, `"Music"`) and (`"Treasure"`, `"Island"`). Then, we only take the highest score obtained for each token provided that this score is above a predefined character-based (sim') threshold δ. In this example, for the sake of simplicity, we set this threshold = 0 and we consider the word `"music"` as a stop-word with the weight w=0.1. Thus, we retain the following triples including the matched tokens and their Jaro scores: (1.0, island, island); (0.95, treasre, treasure). The word `"music"` is still unmatched, so its sim'=0. In this example, we attribute the weight w=1 for the remaining words. The score between s and t is:

$$token{-}wise(s,t) = \frac{1 \times 1 \times 1 + 0.95 \times 1 \times 1}{(1-1) \times (1+1) + (1-0.95) \times (1+1) + (1-0) \times 0.1 + (1+0.95)} = 0.98$$

5.3 Interconnecting geographical data sets

The inter-connexion of geographic data has been a major research domain in geographical information science for a long time and still is. Firstly, many various information, e.g., inhabitants, climate, species, tourism, diseases, is somehow related to earth and the main usage of geographic information is the possibility to search for correlation between different kinds of information based on their spatial relationships. This requires at least one integration step that is referencing these pieces of information in one same geographic reference system. Second, acquiring located data often is a costly process so that data are usually acquired once and reused several times: when a user needs located data he will not acquire it from scratch but will first look for an existing dataset. Third, quality control mechanisms rely on comparing the examined dataset with a reference dataset or comparing together several datasets. Eventually, one current "linked data on the web" argument is shared with geographical data: data should be acquired once and should then be easily discoverable, accessed and combined. This has been reinforced by the European INSPIRE directive (EU INSPIRE Directive 2007/2/EC) which requires state member to contribute to a European infrastructure of geographic information where legal located data, useful to define and monitor Europe environmental policy, be easily accessible and combined.

If many methods and tools have been designed, reusing them demands a careful analysis of their original context and the distance to Datalift objectives. Indeed most tools are valid in a certain context only, either explicitly or because of the context in which they have been designed and tested.

Many approaches have been established in the context of traditional geographical data, i.e., data produced by professionals. In this context, heterogeneities stem from different selections of objects in the world, different levels of details, different representation choices [3]. Some researches have tackled geographical schema matching and assess the necessity to represent unambiguously the semantics of the database, i.e., the relationship with the real world or in other words the database specifications [16, 2]. [2] proposes to encode in a formal ontology semantic knowledge relevant to schema matching. This includes relationship

with an ontology of the real world but also how an object of the real world is represented through database objects. The author proposes an OWL implementation of IGN database specifications aiming at supporting data integration. [23] uses ontologies to convert spatial data from one model to another (which is a key process in INSPIRE to transform producer data into a common model). He annotates source and target schemas with a domain ontology after [19] method. Matching items are identified through inferences and an ontology of spatial analysis operations is used to evaluate if it is possible to derive new items from the target schema by applying these operations on the source schema. [22] have concentrated on data matching based on several ontologies including the data themselves. Other works have studied data matching based either on geometrical criteria, like distance and topological relationships [20] or on a mix of geometric and semantic criteria [1].

More recently researchers have matched traditional data with user contributed data, namely Open Street Map data (published as LinkedGeodata on the LOD). [13] has matched OSM network and OSGB Meridian network to assess some distance between both. The interconnection is not between the roads themselves but only between the networks using [10] distance. [9] performed a similar but more detailed analysis on France OSM data compared to IGN. They interconnected the dataset at the feature level (roads, lakes) using different distance measures that can be reused in the context of Datalift. Last, [12] has experimented the connection of linked data based on spatial criteria. More precisely they have interconnected Geonames residential areas in Geonames and in LinkedGeodata in Germany. Matching at the schema level is performed manually and then the authors combine a Levenshtein minimum string distance between the place names and a buffer intersection method between the place coordinates.

Our objective in datalift is to manage spatial aspects in ontologies interconnexion and in data interconnexion. Ontologies interconnexion can be enhanced by considering more finely the semantic and the kind of geometry. This can be modelled using [2]'s proposal. This will also result in valuable information for data interconnexion. For instance the possible error associated to the determination of the center of a populated place makes it easier to define relevant geometric distance threshold in populated places interconnexion. Data interconnexion can also be enhanced by combining several similarity criteria (shape, distance, semantics, relationships with other features). Distance algorithms can be reused from existing java implementations. Semantics similarity will be provided by the platform, shape or relationships tools are implemented on the Geoxygene platform. The combination of criteria is more tricky and we will adopt [1] results to address this. Last, it is obviously important not only to connect features or feature collections but also to comment on the connection and use carefully the similarity relationships [11]. For instance, is this the similar object at different levels of details? Can a conflation strategy [25] be used or not?

5.4 Conclusion

We have briefly presented three types of special purpose techniques for comparing particular datatypes. We plan to integrate such techniques in a further version of the platform, first through Silk plug-ins.

6. Using EDOAL to control interlinking

We consider here the use of alignments as input of the interlinking process. The type of alignments required to express interlinking with the precision of the script provided in §4.2 goes beyond simple alignments. For expressing expressive alignments, we have designed the EDOAL language [7][1] which must be considered for that purpose.

Working on interlinking the web of data, we have the incentive to use SPARQL in order to preform interlinking. This is because most linked data is offered as a SPARQL endpoint that Silk allows to restrict sources and targets with SPARQL.

Hence the question is how do we generate useful queries from alignments and, specifically, EDOAL alignments because, they may be very useful for that purpose. Relating expressive alignments and SPARQL has already been the topic of a previous paper [8].

In fact, there are several ways to generate queries from alignments (corresponding to the following sections):

- query mediator (queryprocessor) which transform a query according to the alignment (§6.1);
- query generator (easier) which generates a query for each EDOAL expression in a correspondence (§6.2);
- data transformation which generates a unique CONSTRUCT-based SPARQL query able to transform data from one ontology to another (§6.3);
- CONSTRUCT-based SPARQL link generators able to generate a query for creating links between datasets (§6.4).

6.1 Mediating queries

We proposed the QueryProcessor interface and its QueryMediator implementation which takes a query and an alignment and evaluates the query after having translated it with the help of the Alignment (see Figure 6.1).

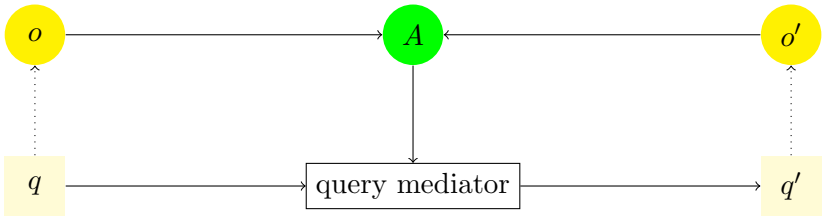

Figure 6.1: A query mediator generated from an alignment allows for transforming queries over data.

Only a trivial string-based version of this has been implemented within the Alignment API (which supports EDOAL). But it served its purpose (because, with simple alignments, it is enough to transcribe URIs).

Since, we have introduced EDOAL within the Alignment API, it would be interesting to redevelop such a mediator for EDOAL. Recently, Gianluca Correndo from Southampton contacted us because he has implemented a partial query mediator for EDOAL [6]. That would be useful to extend it and integrate it within the API.

[1] `http://alignapi.gforge.inria.fr/edoal.html`

6.2 Extracting linked data

For dealing with linked data, we actually do not need to transform queries, but to generate them. In particular, we need to generate queries from alignments in EDOAL in order to restrict the investigations of the data to be matched. What is needed is displayed by Figure 6.2.

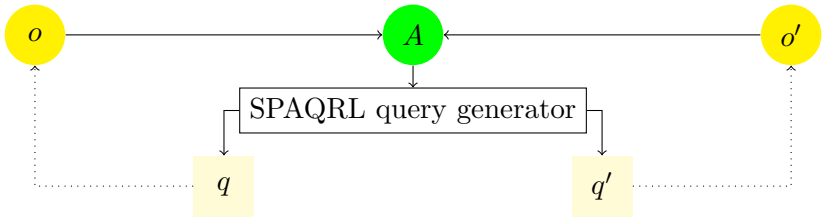

Figure 6.2: A query generator generated from an alignment allows for producing the SPARQL queries which restrict the search area in which an interlinker would look for.

What is needed is to retrieve all instances of EDOAL expressions appearing in both sides of a correspondence in order to see if they can be matched. For instance, with respect to the following EDOAL correspondence:

```
<Cell>
  <entity1>
    <edoal:Class rdf:about="&insee;Region" />
  </entity1>
  <entity2>
    <edoal:Class>
      <edoal:and rdf:parseType="Collection">
        <edoal:Class rdf:about="&nults;NUTSRegion"/>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Property rdf:about="&nuts;level"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&xsd;equals"/>
          <edoal:value>2</edoal:value>
        </edoal:AttributeValueRestriction>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Relation rdf:about="&nuts;hasParentRegion"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&xsd;equals"/>
          <edoal:value><edoal:Instance rdf:about="&nuts;FR1"/></edoal:value>
        </edoal:AttributeValueRestriction>
      </edoal:and>
    </edoal:Class>
  </entity2>
  <measure rdf:datatype='&xsd;float'>1.</measure>
  <relation>=</relation>
</Cell>
```

corresponding to Figure 6.3: it expresses the equivalence between the class `Region` in the insee data set with `NUTSRegion` instances whose `level` is equal to 2 and whose parent region is `FR1` in the NUTS data set.

It is possible to generate the following queries which respectively extract the two corresponding set of entities:

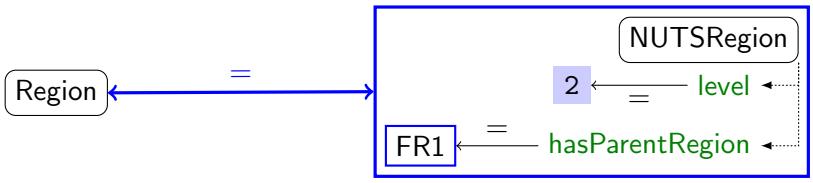

Figure 6.3: The EDOAL correspondence between INSEE regions and NUTS French region of level 2 corresponding to the example given in the Silk script of §4.2.

```
SELECT ?r
PREFIX insee: <http://rdf.insee.fr/ontologie-geo-2006.rdf#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
FROM <http://rdf.insee.fr/geo/regions-2011.rdf>
WHERE
    ?r rdf:type insee:Region .
```

and

```
SELECT ?n
PREFIX nuts: <http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
FROM <http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/>
WHERE
    ?n rdf:type nuts:NUTSRegion .
    ?n nuts:level 2^^xsd:int .
    ?n nuts:hasParentRegion nuts:FR1 .
```

6.3 Importing linked data

Another use consists of not only extracting the data but importing data from one dataset into another dataset. What is then needed is a CONSTRUCT query that will transform the data (see Figure 6.4).

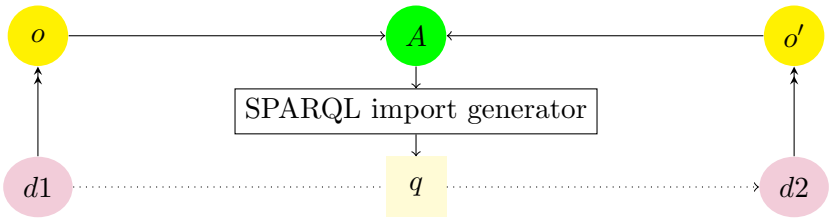

Figure 6.4: An import query can be generated from an alignment in order to import data from one data set into another.

In the case of the latter query, it is possible to generate data corresponding to the `NUTSRegions` of `level` two situated in France to `Regions`:

```
CONSTRUCT  ?n rdf:type insee:Region .
PREFIX insee: <http://rdf.insee.fr/ontologie-geo-2006.rdf#>
PREFIX nuts: <http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
FROM <http://rdf.insee.fr/geo/regions-2011.rdf>
```

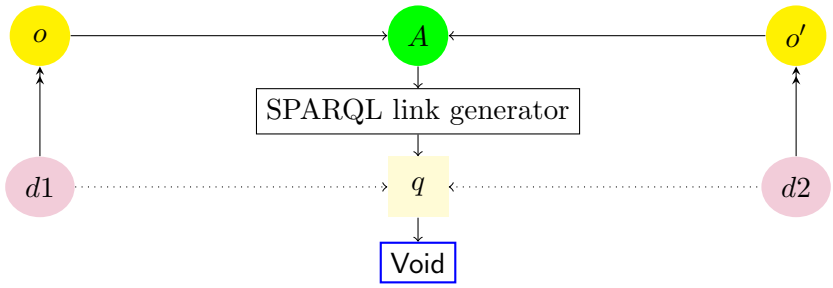

Figure 6.5: A SPARQL link generator generated from an alignment can directly link data sets with a single SPARQL query.

```
FROM <http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/>
WHERE
    ?n rdf:type nuts:NUTSRegion .
    ?n nuts:level 2^^xsd:int .
    ?n nuts:hasParentRegion nuts:FR1 .
```

or, in the reverse direction,

```
CONSTRUCT
    ?r rdf:type nuts:NUTSRegion .
    ?r nuts:level 2^^xsd:int .
    ?r nuts:hasParentRegion nuts:FR1 .
PREFIX insee: <http://rdf.insee.fr/ontologie-geo-2006.rdf#>
PREFIX nuts: <http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
FROM <http://rdf.insee.fr/geo/regions-2011.rdf>
FROM <http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/>
WHERE
    ?r rdf:type insee:Region .
```

In the latter case, this is only possible because the query pattern is definite, i.e., by constraining the properties to have an exact value, we know which value must the generated instances have.

This is, however, not what is needed in the Datalift interlinking module.

6.4 Generating links

Finally, one can imagine using SPARQL CONSTRUCT queries in order to generate `owl:sameAs` links between resources of two datasets. For that purpose, it is necessary that the EDOAL alignment provides enough information to identify the corresponding resources.

This is the case of the following correspondence which corresponds to the correspondence of §6.2 to which are added transformation information to transform the `nom` of an INSEE `Region` into the `label` of a `NUTSRegion`:

```
<Cell>
  <entity1>
    <edoal:Class rdf:about="&insee;Region" />
  </entity1>
  <entity2>
    <edoal:Class>
```

```
      <edoal:and rdf:parseType="Collection">
        <edoal:Class rdf:about="&nuts;NUTSRegion"/>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Property rdf:about="&nuts;level"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&xsd;equals"/>
          <edoal:value>2</edoal:value>
        </edoal:AttributeValueRestriction>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Relation rdf:about="&nuts;hasParentRegion"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&xsd;equals"/>
          <edoal:value><edoal:Instance rdf:about="&nuts;FR1"/></edoal:value>
        </edoal:AttributeValueRestriction>
      </edoal:and>
    </edoal:Class>
  </entity2>
  <!-- This may not be standard EDOAL -->
  <edoal:transformation>
    <edoal:Transformation edoal:type="o-">
      <edoal:entity1>
        <edoal:Property rdf:about="&insee;nom"/>
      </edoal:entity1>
      <edoal:entity2>
        <edoal:Property rdf:about="&nuts;name"/>
      </edoal:entity2>
    </edoal:Transformation>
  </edoal:transformation>
  <!-- A better rewriting -->
  <edoal:linkConstraint>
    <edoal:Cell>
      <edoal:entity1>
        <edoal:Property rdf:about="&insee;nom"/>
      </edoal:entity1>
      <edoal:entity2>
        <edoal:Property rdf:about="&nuts;name"/>
      </edoal:entity2>
    </edoal:Cell>
  </edoal:linkConstraint>
  <measure rdf:datatype='&xsd;float'>1.</measure>
  <relation>=</relation>
</Cell>
```

This information can be expressed either as a transformation or as a constraints. It could also be expressed as a linking condition. This correspondence corresponds to (part of) Figure 6.6.

The corresponding SPARQL query generating links is:

```
CONSTRUCT  ?r owl:sameAs ?n .
PREFIX insee: <http://rdf.insee.fr/ontologie-geo-2006.rdf#>
PREFIX nuts: <http://ec.europa.eu/eurostat/ramon/ontologies/geographic.rdf#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
FROM <http://rdf.insee.fr/geo/regions-2011.rdf>
FROM <http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/>
WHERE
    ?r rdf:type insee:Region .
    ?n rdf:type nuts:NUTSRegion .
    ?n nuts:level 2^^xsd:int .
```

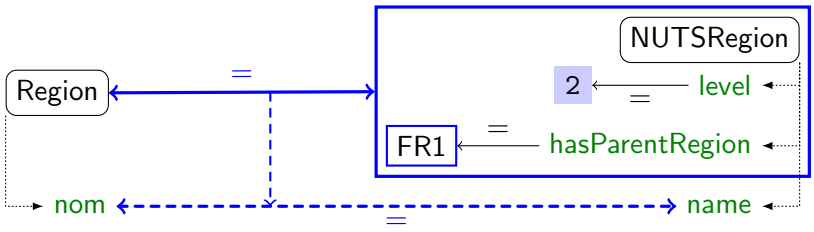

Figure 6.6: A more elaborate EDOAL correspondence indicating which properties must be compared in order to establish the link.

```
?n nuts:hasParentRegion nuts:FR1 .
?r insee:nom ?l .
?n nuts:name ?l .
```

This single SPARQL query performs what the script of §4.2 has accomplished. It can, in principle, be directly generated from the correspondences (similar Silk scripts could be generated as well).

6.5 Implementation

This is not implemented so far but will certainly have to in the context of Datalift. Gianlucca Correndo already has a beginning of implementation of the importer generation. The obvious way to implement this would be to use the Visitor pattern to traverse inductively EDOAL expressions.

It is necessary to investigate how to share part of the code of these three functions. It seems clear that the two latter share at least the query pattern that is created inductively on the structure of the EDOAL expression. It is possible that the CONSTRUCT part could be expressed in the same way. The first type of operation, query mediation, is quite different because it is a query transformation and not a data transformation.

6.6 Going further

EDOAL expression transformation is far from a simple task because:

- we want to rewrite query parts which match the most specific EDOAL expression
- it is difficult to recognise them because they may be interleaved with others.

Indeed, it is difficult to predict the interaction between the SPARQL queries at runtime. This is a well known problem of term rewriting: that of confluence. We want that whatever the order in which we evaluate the SPARQL queries, this will in the end provide the same result. This is trivially true if not two queries can match the same piece of data, not if several queries can concern the same resource. In the present case, we also would like that all rewriting concerning the same entity be treated within the same query (for both performance and coherence considerations). This problem seems to be well known in database mapping generation and is treated (for simple correspondences) in a recent paper [21].

7. Conclusion

We now have an operational data interlinking module within the platform which should allow to experiment with novel interlinking algorithms. We also have plans for enriching this module that will be implemented in next year.

Our planned next steps are:

- integrating additional components
- integrating Silk script generator based on alignments (DMT)
- integrating Silk script generator based on alignments (EDOAL based)

We plan to achieve them within three months and to be able to experiment with this setting in order to decide if this strategy is valid or if we must change for another.

References

[1] Olteanu A.-M. A multi criteria fusion approach for geographical data. In A. Stein, W. Shi, and W. Bijker, editors, *Quality Aspects in Spatial Data Mining*, pages 45–56. Taylor and Francis, 2008.

[2] N. Abadie. Schema matching based on attribute values and background ontology. In *Proc. 12th AGILE International Conference on Geographic Information Science, Hanovre (DE)*, 2009.

[3] Y. Bishr. *Semantic Aspects of Interoperable GIS*. PhD thesis, ITC, Netherlands, 1997.

[4] Christian Bizer, Julius Volz, Georgi Kobilarov, and Martin Gaedke. Silk - a link discovery framework for the web of data. In *Proc. 2nd Workshop on Linked Data on the Web (LDOW)*, April 2009.

[5] W. Cohen, P. Ravikumar, and S. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *1st International Workshop on Information Integration on the Web (IIWeb'03)*, pages 73–78, Acapulco, Mexico, 2003.

[6] Gianluca Correndo, Manuel Salvadores, Ian Millard, Hugh Glaser, and Nigel Shadbolt. Sparql query rewriting for implementing data integration over linked data. In Florian Daniel, Lois M. L. Delcambre, Farshad Fotouhi, Irene Garrigós, Giovanna Guerrini, Jose-Norberto Mazón, Marco Mesiti, Sascha Müller-Feuerstein, Juan Trujillo, Traian Marius Truta, Bernhard Volz, Emmanuel Waller, Li Xiong, and Esteban Zimányi, editors, *Proc. EDBT/ICDT Workshops, Lausanne, Switzerland, March 22-26, 2010*, 2010.

[7] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The alignment api 4.0. *Semantic web journal*, 2(1):3–10, 2011.

[8] Jérôme Euzenat, François Scharffe, and Axel Polleres. Processing ontology alignments with sparql (position paper). In *Proc. IEEE international workshop on Ontology alignment and visualization (OAaV), Barcelona (ES)*, pages 913–917, 2008.

[9] J.-F. Girres and G. Touya. Quality assessment of the french openstreetmap dataset. *Transactions in GIS*, 14(4):435–459, 2010.

[10] M. Goodchild and G. Hunter. A simple positional accuracy measure for linear features. *International Journal of Geographical Information Science*, 11(3):299–306, 1997.

[11] J. Goodwin, C. Dolbear, and G. Hart. Geographical linked data: The administrative geography of great britain on the semantic web. *Transactions in GIS*, 12(1):19–30, 2008.

[12] S. Hahmann and D. Burghardt. Connecting linkedgeodata and geonames in the spatial semantic web. In *Proc. 6th International GIScience Conference, Zurich (CH)*, 2010.

[13] M. Haklay. How good is openstreetmap information? a comparative study of openstreetmap and ordnance survey datasets for london and the rest of england. *Environment and Planning B: Planning and Design*, 37(4):682–703, 2010.

[14] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[15] Matthew A. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[16] M. Kavouras and M. Kokla. *Theories of Geographic Concepts: Ontological Approaches to Semantic Integration*. CRC Press/Taylor & Francis, Boca Raton (FL US), 2008.

[17] Houda Khrouf and Raphaël Troncy. EventMedia Live: Reconciliating Events Descriptions in the Web of Data. In *6th International Workshop on Ontology Matching*, Bonn, Germany, 2011.

[18] Houda Khrouf and Raphaël Troncy. Réconcilier les événements dans le web de données. In *22nd Journées d'Ingénierie des Connaissances (IC'11)*, Chambéry, France, 05 2011.

[19] E. Klien. *Semantic Annotation of Geographic Information*. PhD thesis, Munster University (DE), 2008.

[20] S. Mustière and T. Devogele. Matching networks with different levels of detail. *GeoInformatica*, 12(4):435–453, 2008.

[21] Carlos Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo. Generating sparql executable mappings to integrate ontologies. In *Proc. 30th international conference on conceptual modelling (ER), Brussels (BE)*, 2011.

[22] A. Rodriguez and M. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and data Engineering*, 12(2):442–456, 2003.

[23] S. Schade. Towards semantic datums for geospatial information. In *Proc. 5th International Conference on Formal Ontology in Information Systems, Saarbrücken, Germany*, 2008.

[24] D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *Proc. 10th International Semantic Web Conference (ISWC), Bonn (DE)*, 2011.

[25] S. Yuan and C. Tao. Development of conflation components. In *Proc. Geoinformatics'99 Conference*, pages 1–13, 1999.