Discovering and Maintaining Links on the Web of Data

Julius Volz[1], Christian Bizer[2], Martin Gaedke[1], and Georgi Kobilarov[2]

[1] Chemnitz University of Technology
Distributed and Self-Organizing Systems Group
Straße der Nationen 62, 09107 Chemnitz, Germany
volz@hrz.tu-chemnitz.de
gaedke@cs.tu-chemnitz.de

[2] Freie Universität Berlin
Web-based Systems Group
Garystr. 21, 14195 Berlin, Germany
chris@bizer.de
georgi.kobilarov@fu-berlin.de

Abstract. The Web of Data is built upon two simple ideas: Employ the RDF data model to publish structured data on the Web and to create explicit data links between entities within different data sources. This paper presents the Silk – Linking Framework, a toolkit for discovering and maintaining data links between Web data sources. Silk consists of three components: 1. A link discovery engine, which computes links between data sources based on a declarative specification of the conditions that entities must fulfill in order to be interlinked; 2. A tool for evaluating the generated data links in order to fine-tune the linking specification; 3. A protocol for maintaining data links between continuously changing data sources. The protocol allows data sources to exchange both linksets as well as detailed change information and enables continuous link recomputation. The interplay of all the components is demonstrated within a life science use case.

Key words: Linked data, web of data, link discovery, link maintenance, record linkage, duplicate detection

1 Introduction

The central idea of Linked Data is to extend the Web with a data commons by creating typed links between data from different data sources [1, 2]. Technically, the term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web in a way that data is machine-readable, its meaning is explicitly defined, it is linked to other external datasets, and can in turn be linked to from external datasets. The data links that connect data sources take the form of RDF triples, where the subject of the triple is a URI

reference in the namespace of one dataset, while the object is a URI reference in the other [2, 3].

The most visible example of adoption and application of Linked Data has been the Linking Open Data (LOD) project[3], a grassroots community effort to bootstrap the Web of Data by interlinking open-license datasets. Out of the 6.7 billion RDF triples that are served as of July 2009 by participants of the project, approximately 148 million are RDF links between datasets[4].

As Linked Data sources often provide information about large numbers of entities, it is common practice to use automated or semi-automated methods to generate RDF links. In various domains, there are generally accepted naming schemata, such as ISBN and ISSN numbers, ISIN identifiers, EAN and EPC product codes. If both datasets already support one of these identification schemata, the implicit relationship between entities in the datasets can easily be made explicit as RDF links. This approach has been used to generate links between various data sources in the LOD cloud. If no shared naming schema exists, RDF links are often generated by computing the similarity of entities within both datasets using a combination of different property-level similarity metrics.

While there are more and more tools available for publishing Linked Data on the Web [3], there is still a lack of tools that support data publishers in setting RDF links to other data sources, as well as tools that help data publishers to maintain RDF links over time as data sources change. The Silk – Linking Framework contributes to filling this gap. Silk consists of three components: 1. A link discovery engine, which computes links between data sources based on shared identifiers and/or object similarity; 2. A tool for evaluating the generated RDF links in order to fine-tune the linking specification; 3. A protocol for maintaining RDF links between continuously changing data sources.

Using the declarative *Silk - Link Specification Language (Silk-LSL)*, data publishers can specify which types of RDF links should be discovered between data sources as well as which conditions data items must fulfill in order to be interlinked. These link conditions can apply different similarity metrics to multiple properties of an entity or related entities which are addressed using a path-based selector language. The resulting similarity scores can be weighted and combined using various similarity aggregation functions. Silk accesses data sources via the SPARQL protocol and can thus be used to discover links between local or remote data sources.

The main features of the Silk link discovery engine are:

- it supports the generation of owl:sameAs links as well as other types of RDF links.
- it provides a flexible, declarative language for specifying link conditions.
- it can be employed in distributed environments without having to replicate datasets locally.

[3] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[4] http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics

- it can be used in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist.
- it implements various caching, indexing and entity preselection methods to increase performance and reduce network load.

Datasets change and are extended over time. In order to keep links between two data sources current and to avoid dead links, new RDF links should be continuously generated as entities are added to the target dataset and invalidated RDF links should be removed. For this task, we propose the *Web of Data – Link Maintenance Protocol (WOD-LMP)*. The protocol automates the communication between two cooperating Web data sources: The *link source* and the *link target*, where the link source is a Web data source that publishes RDF links pointing at data published by the target data source. The protocol supports:

- notifying the target data source that the link source has published a set of links pointing at the target source. This allows the target to track incoming links and decide whether it wants to set back-links.
- the link source to request a list of changes from the target source. Based on the changes, the link source can recompute existing links and generate additional links pointing at new resources.
- the link source to monitor resources in the target dataset by subscribing to be informed about changes that occur to these resources.

This paper is structured as follows: Section 2 gives an overview of the Silk – Link Specification Language along a concrete usage example. In Section 3, we present the Silk user interface to evaluate generated links. We describe the Web of Data – Link Maintenance Protocol in Section 4 and give an overview of the implementation of the Silk framework in Section 5. Section 6 reviews related work.

2 Link Specification Language

The *Silk – Link Specification Language (Silk-LSL)* is used to express heuristics for deciding whether a semantic relationship exists between two entities. The language is also used to specify the access parameters for the involved data sources, and to configure the caching, indexing and preselection features of the framework. Link conditions can use different aggregation functions to combine similarity scores. These aggregation functions as well as the implemented similarity metrics and value transformation functions were chosen by abstracting from the link heuristics that were used to establish links between data sources in the LOD cloud.

Figure 1 contains a complete Silk-LSL link specification. In this particular use case, we want to discover `owl:sameAs` links between the URIs that are used by DBpedia [4], a data source publishing information extracted from Wikipedia, and by the Linked Data version of DrugBank [5], a pharmaceutical database, to identify medical drugs. In line 22 of the link specification, we thus configure the `<LinkType>` to be `owl:sameAs`.

```
<Silk>
    <DataSource id="dbpedia">
        <EndpointURI>http://dbpedia.org/sparql</EndpointURI>
        <Graph>http://dbpedia.org</Graph>
        <DoCache>1</DoCache>
        <PageSize>1000</PageSize>
    </DataSource>
    <DataSource id="drugbank">
        <EndpointURI>http://www4.wiwiss.fu-berlin.de/drugbank/sparql</EndpointURI>
    </DataSource>
    <Metric id="jaroSets">
        <Param name="item1" />
        <Param name="item2" />
        <AVG>
            <Compare metric="jaroWinklerSimilarity">
                <Param name="str1" path="?item1" />
                <Param name="str2" path="?item2" />
            </Compare>
        </AVG>
    </Metric>
    <Interlink id="drugs">
        <LinkType>owl:sameAs</LinkType>
        <SourceDataset dataSource="dbpedia" var="a">
            <RestrictTo>
                ?a rdf:type dbpedia:Drug
            </RestrictTo>
        </SourceDataset>
        <TargetDataset dataSource="drugbank" var="b">
            <RestrictTo>
                ?b rdf:type drugbank:drugs
            </RestrictTo>
        </TargetDataset>
        <LinkCondition>
            <AVG>
                <MAX weight="1">
                    <Compare metric="maxSimilarityInSets">
                        <Param name="set1" path="?a/rdfs:label" />
                        <Param name="set2" path="?b/rdfs:label" />
                        <Param name="submetric" value="jaroSets" />
                    </Compare>
                    <Compare metric="maxSimilarityInSets" optional="1">
                        <Param name="set1" path="?a/rdfs:label" />
                        <Param name="set2" path="?b/drugbank:synonym" />
                        <Param name="submetric" value="jaroSets" />
                    </Compare>
                    <Compare metric="maxSimilarityInSets" optional="1">
                        <Param name="set1" path="?a/rdfs:label" />
                        <Param name="set2" path="?b/drugbank:genericName" />
                        <Param name="submetric" value="jaroSets" />
                    </Compare>
                </MAX>
                <MAX optional="1" weight="5">
                    <Compare metric="stringEquality" optional="1">
                        <Param name="str1">
                            <Transform function="concat">
                                <Param name="str1" path="?a/dbpedia:atcprefix" />
                                <Param name="str2" path="?a/dbpedia:atcsuffix" />
                            </Transform>
                        </Param>
                        <Param name="str2" path="?b/drugbank:atcCode" />
                    </Compare>
                    <Compare metric="stringEquality" optional="1">
                        <Param name="str1" path="?a/dbpedia:casNumberLink" />
                        <Param name="str2" path="?b/drugbank:casRegistryNumber" />
                    </Compare>
                    <Compare metric="stringEquality" optional="1">
                        <Param name="str1" path="?a/dbpedia:pubchem" />
                        <Param name="str2" path="?b/drugbank:pubchemCompoundId" />
                    </Compare>
                </MAX>
                <Compare metric="numSimilarity" optional="1" weight="2">
                    <Param name="num1" path="?a/dbpedia:molecularweight" />
                    <Param name="num2" path="?b/drugbank:molecularWeightAverage" />
                </Compare>
            </AVG>
        </LinkCondition>
        <Thresholds accept="0.9" verify="0.7" />
        <Limit max="1" method="metric_value" />
        <Output acceptedLinks="drug_accepted_links.n3"
         verifyLinks="drug_verify_links.n3" format="n3" mode="truncate" />
    </Interlink>
</Silk>
```

Fig. 1. Example: Interlinking drugs in DBpedia and DrugBank

2.1 Data Access

For accessing the source and target data sources, we first specify access parameters for the DBpedia and DrugBank SPARQL endpoints using the `<DataSource>` directive. The only mandatory data source parameter is the SPARQL endpoint URI. Besides this, it is possible to define other data source access options, such as the graph name and to enable in-memory caching of SPARQL query results. In order to restrict the query load on remote SPARQL endpoints, it is possible to set a delay between subsequent queries using the `<Pause>` parameter, specifying the delay time in milliseconds. For working against SPARQL endpoints that restrict result sets to a certain size, Silk uses a paging mechanism. The maximal result size is configured using the `<PageSize>` parameter. The paging mechanism is implemented using SPARQL `LIMIT` and `OFFSET` queries. Lines 2 to 7 in Figure 1 show how the access parameters for the DBpedia data source are set to select only resources from the named graph `http://dbpedia.org`, enable caching and limit the page size to 1,000 results per query.

The configured data sources are later referenced in the `<SourceDataset>` and `<TargetDataset>` clauses of the link specification. Since we only want to match drugs, we restrict the sets of examined resources to instances of the classes `dbpedia:Drug` and `drugbank:drugs` in the respective datasets by supplying SPARQL conditions within the `<RestrictTo>` directives in lines 25 and 30. These statements may contain any valid SPARQL expressions that would usually be found in the `WHERE` clause of a SPARQL query.

2.2 Link Conditions

The `<LinkCondition>` section is the heart of a Silk link specification and defines how similarity metrics are combined in order to calculate a total similarity value for a pair of entities. For comparing property values or sets of entities, Silk provides a number of built-in similarity metrics. Table 1 gives an overview of these metrics. The implemented metrics include string, numeric, date, URI, and set comparison methods as well as a taxonomic matcher that calculates the semantic distance between two concepts within a concept hierarchy using the distance metric proposed by Zhong et al. in [8]. Each metric in Silk evaluates to a similarity value between 0 or 1, with higher values indicating a greater similarity.

These similarity metrics may be combined using the following aggregation functions:

- AVG – weighted average
- MAX – choose the highest value
- MIN – choose the lowest value
- EUCLID – Euclidian distance metric
- PRODUCT – weighted product

To take into account the varying importance of different properties, the metrics grouped inside the AVG, EUCLID and PRODUCT operators may be

Table 1. Available similarity metrics in Silk

jaroSimilarity	String similarity based on Jaro distance metric[6]
jaroWinklerSimilarity	String similarity based on Jaro-Winkler metric[7]
qGramSimilarity	String similarity based on q-grams
stringEquality	Returns 1 when strings are equal, 0 otherwise
numSimilarity	Percentual numeric similarity
dateSimilarity	Similarity between two date values
uriEquality	Returns 1 if two URIs are equal, 0 otherwise
taxonomicSimilarity	Metric based on the taxonomic distance of two concepts
maxSimilarityInSet	Returns the highest encountered similarity of comparing a single item to all items in a set
setSimilarity	Similarity between two sets of items

weighted individually, with higher weighted metrics having a greater influence on the aggregated result.

In the `<LinkCondition>` section of the example (lines 33 to 76), we compute similarity values for the the labels, PubChem IDs[5], CAS registry numbers[6], ATC codes[7] and molecular weights between datasets and calculate a weighted average of these values.

Most metrics are configured to be optional since the presence of the respective RDF property values they refer to is not always guaranteed. In cases where alternating properties refer to an equivalent feature (such as `rdfs:label`, `drugbank:synonym` and `drugbank:genericName`), we choose to perform comparisons for both properties and select the best evaluation by using the `<MAX>` aggregation operator. The `<MAX>` operator is also used to choose the maximum value of the comparisons between any of the exact drug identifiers. Weighting of results is used within the metrics comparing these exact values (line 52), with the metric weight raised to 5, as well as within the molecular weight comparison using a weighting factor of 2.

Lines 11 to 20 demonstrate how a user-defined metric is specified. User-defined metrics may be used like built-in metrics. In the example, the defined `jaroSets` metric is used as a submetric for the `maxSimilarityInSets` evaluations in lines 36-50 for the pairwise comparison of elements of the compared sets. In this case, the user-defined metric is mainly a wrapper around a `jaroWinklerSimilarity` call to achieve type-compatibility with the set comparison interface.

Property values are often represented differently across datasets and thus need to be normalized before being compared. For handling this task, it is possible to apply data transformation functions to parameter values before passing them to a similarity metric. The available transformation functions are shown in Table 2. In the drug linking example, a drug's ATC code in the DBpedia dataset

[5] `http://pubchem.ncbi.nlm.nih.gov/`
[6] `http://www.cas.org/expertise/cascontent/registry/regsys.html`
[7] `http://www.who.int/classifications/atcddd/en/`

is split into a prefix and a suffix part, while it is stored in a single property on the DrugBank side. Hence, we use the `concat` transformation function to concatenate the code's prefix and suffix parts on the DBpedia side before comparing it to the single-property code in DrugBank (lines 55 to 58).

Table 2. Available transformation functions in Silk

removeBlanks	Remove whitespace from string
removeSpecialChars	Remove special characters from string
lowerCase	Convert a string to lower case
upperCase	Convert a string to upper case
concat	Concatenate two strings
stem	Apply word stemming to a string
alphaReduce	Strip all non-alphabetic characters from a string
numReduce	Strip all non-numeric characters from a string
replace	Replace all occurrences of a string with a replacement
regexReplace	Replace all occurences of a regex with a replacement
stripURIPrefix	Strip the URI prefix from a string
translateWithDictionary	Translate string using a provided CSV dictionary file

After specifying the link condition, we finally specify within the `<Thresholds>` clause that resource pairs with a similarity score above 0.9 are to be interlinked, whereas pairs between 0.7 and 0.9 should be written to a separate output file in order to be reviewed by an expert. The `<Limit>` clause is used to limit the number of outgoing links from a particular entity within the source dataset. If several candidate links exist, only the highest evaluated one is chosen and written to the output files as specified by the `<Output>` directive. In this example, we permit only one outgoing `owl:sameAs` link from each resource.

Discovered links can be outputted either as simple RDF triples and/or in reified form together with their creation date, confidence score and the URI identifying the employed interlinking heuristic.

2.3 Silk Selector Language

Especially for discovering semantic relationships other than entity equality, a flexible way for selecting sets of resources or literals in the RDF graph around a particular resource is needed. Silk addresses this requirement by offering a simple RDF path selector language for providing parameter values to similarity metrics and transformation functions. A Silk selector language path starts with a variable referring to an RDF resource and may then use several path operators to navigate the graph surrounding this resource. To simply access a particular property of a resource, the forward operator (/) may be used. For example, the path "`?drug/rdfs:label`" would select the set of label values associated with a drug referred to by the `?drug` variable.

Sometimes, we need to navigate backwards along a property edge. For example, drugs in DrugBank contain a `drugbank:target` property pointing to the drug's target molecule. However, there exists no explicit reverse property like `drugbank:drug` in the drug target's resource. So if a path begins with a drug target and we need to select all of the drugs that apply to it, we may use the backward operator (\ ) to navigate property edges in reverse. Navigating backwards along the property `drugbank:target` would select the applicable drugs.

The filter operator ([ ]) can be used to restrict selected resources to match a certain predicate. To select only drugs amongst the ones applicable to a target molecule which have been marked as *approved*, we could for instance use the RDF path "`?target\drugbank:target[drugbank:drugType drugType:approved]`". The filter operator also supports numeric comparisons. For example, to select drugs with a molecular weight above 200, the path "`?target\drugbank:target [drugbank:molecularWeightAverage > 200]`" can be used.

2.4 Pre-Matching

To compare all pairs of entities of a source dataset S and a target dataset T would result in an unsatisfactory runtime complexity of $O(|S| \cdot |T|)$. Even after using SPARQL restrictions to select suitable subsets of each dataset, the required time and network load to perform all pair comparisons might prove to be impractical in many cases. To avoid this problem, we need a way to quickly find a limited set of target entities that are likely to match a given source entity. Silk provides this by offering rough index pre-matching.

When using pre-matching, all target resources are indexed by the values of one or more specified properties (most commonly, their labels) before any detailed comparisons are performed. During the subsequent resource comparison phase, the previously generated index is used to look up potential matches for a given source resource. This lookup uses the BM25[8] weighting scheme for the ranking of search results and additionally supports spelling corrections of individual words of a query. Only a limited number of target resources found in this lookup is then considered as candidates for a detailed comparison.

An example of such a pre-matching specification that could be applied to our drug linking example is presented in Figure 2. This directive instructs Silk to index the drugs in the target dataset by their `rdfs:label` and `drugbank:synonym` property values. When performing comparisons, the `rdfs:label` of a source resource is used as a search term into the generated indexes and only the first ten target hits found in each index are considered as link candidates for detailed comparisons.

If we neglect a slight index insertion and search time dependency on the target dataset size, we now achieve a runtime complexity of $O(|S| + |T|)$, making it feasible to interlink even large datasets under practical time constraints. Note however that this pre-matching may come at the cost of missing some links

[8] `http://xapian.org/docs/bm25.html`

```
<PreMatchingDefinition sourcePath="?a/rdfs:label" hitLimit="10">
    <Index targetPath="?b/rdfs:label" />
    <Index targetPath="?b/drugbank:synonym" />
</PreMatchingDefinition>
```

Fig. 2. Pre-Matching

during discovery, since it is not guaranteed that a pre-matching lookup will always find all matching target resources.

3 Evaluating Links

In real-world settings, data is often not as clean and complete as we would wish it to be. For instance, two data sources might both support the same identification schema, like EAN, ISBN or ISIN numbers, but due to a large number of missing values, it is nevertheless necessary to use similarity computations in addition to identifier matching to generate links. Such data quality problems are usually not known in advance but discovered when a data publisher tries to compute links pointing to a target data source. Therefore, finding a good linking heuristic is usually an iterative process. In order to support users with this task, Silk provides a Web interface for evaluating the correctness and completeness of generated links. Based on this evaluation, users can fine-tune their linking specification, for example by changing weights or applying different metrics or aggregation functions.

3.1 Resource Comparison

The resource comparison component of the Silk web interface allows the user to quickly evaluate links according to the currently loaded linking specification. A screenshot of this interface is shown in Figure 3.

The user first enters a set of RDF links into the box at the top of the screen. Silk then recomputes these links and displays the resulting similarity scores for each link in an overview table. For further examination, a drill-down view of a specific pair comparison can be shown by clicking on one of the table rows. This drill-down shows in a tree-like fashion the exact parameterizations and evaluations of all submetrics and aggregations employed. This information allows the user to spot parts of the similarity evaluation which did not behave as expected.

An example drill-down of a comparison between the DrugBank and DBpedia resources describing the drug Lorazepam is shown in Figure 4. As evident from the illustration, the two drug entries are matched successfully with a high total similarity score although several subcomparisons return infavorable results. For example, the comparison of the DBpedia resource's label with the synonyms on the DrugBank side finds only a similarity of 0.867. However, since perfectly matching labels exist on both sides, the `<MAX>` operator grouping these name-related property comparisons evaluates to a total similarity value of 1. Similarly,

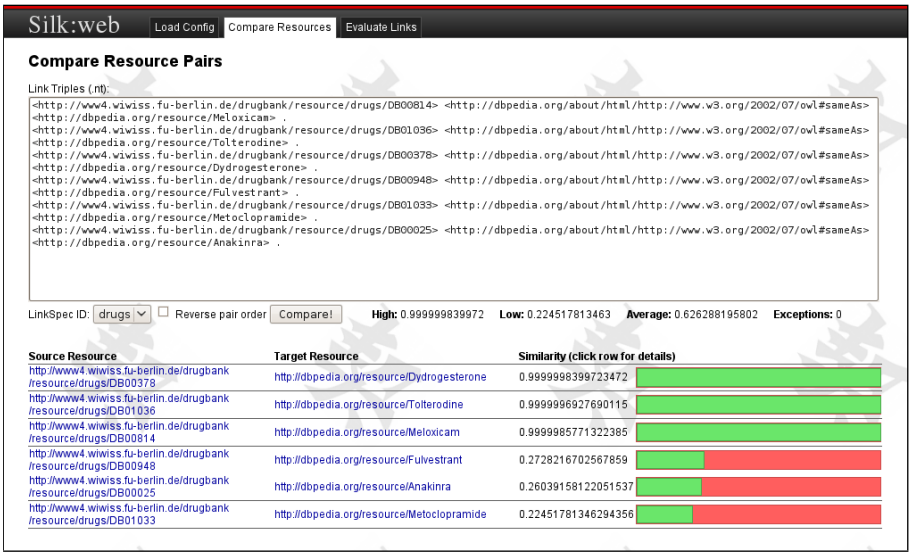

Fig. 3. Comparing resource pairs with the Silk web interface

due to a dataset error, the section aggregating exact numeric drug identifiers contains a similarity value of 0 for the CAS registry numbers. This erroneously low value is corrected by the availability of other exactly matching identifiers in a `<MAX>` aggregation.

3.2 Evaluation against a Reference Linkset

A methodology that proved useful for optimizing link specifications is to manually create a small reference linkset and then optimize the Silk linking specification to produce these reference links, before Silk is run against the complete target data source. Once such a reference linkset is available, the Silk web interface provides a linkset evaluation component which allows the comparison of generated linksets to the reference set. This component is shown in Figure 5.

Silk displays which links are missing from the generated set as well as which resource pairs were interlinked erroneously. To give an overall indication about the linkset quality, Silk also computes statistical measures pertaining to completeness and correctness of the generated links. A *Precision* value indicates the correctness of generated links, while a *Recall* value measures the completeness of discovered links. Finally, the F_1-*measure* calculates the weighted harmonic mean of both, providing an overall-quality measure of the linkset.

3.3 Improving the DBpedia/DrugBank Link Specification

We compared 3,134 drugs in DBpedia with 4,772 drugs in DrugBank. As a result of applying the linking specification shown in Figure 1, Silk discovered

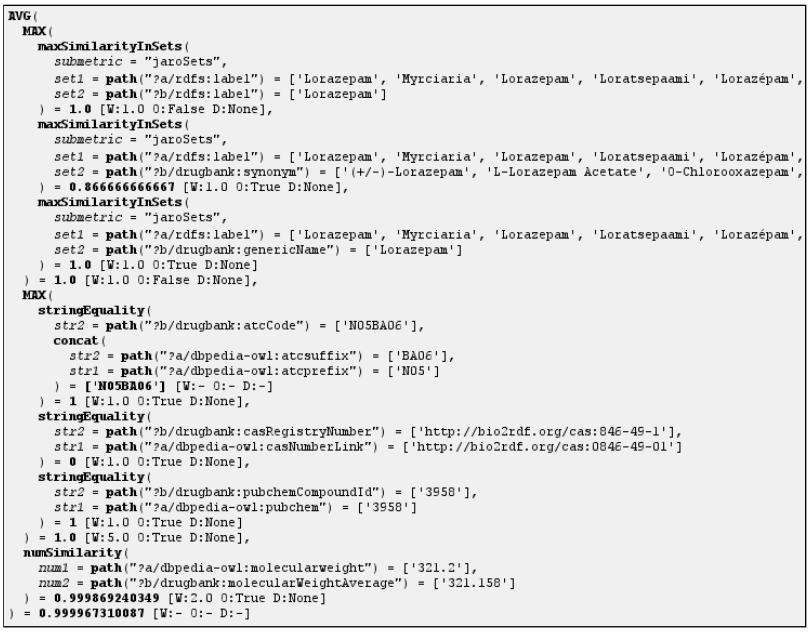

Fig. 4. Detailed drill-down into a resource pair comparison

1,227 confident links above the threshold of 0.9 and found 32 more links above the threshold of 0.7. To evaluate the quality of the retrieved links, we created a reference linkset pertaining to 50 drugs selected randomly from DrugBank and found 38 manually researched links to DBpedia. We then ran Silk a second time to find only links from these 50 selected DrugBank resources to DBpedia and compared both the generated and the reference linkset.

The evaluation revealed 4 missing links and one incorrectly discovered link. This corresponded to a Precision of 0.97, a Recall of 0.89 and an F_1-measure of 0.93. To better understand why certain links are missing and why one link was incorrect, we then compared their source and target resources via the resource comparison web interface. One link was missed because of radically differing molecular weights in both datasets. Three other missing links were not discovered due to the fact that their CAS registry numbers did not match while at the same time no other exact identifiers were present. Finally, one link was discovered incorrectly since the resource labels were very similar and no other relevant property values were present in the datasets. In a subsequent tuning of the link specification, we mitigated the effect of a single mismatching exact identifier by lowering the weight for the surrounding aggregation to 3 and setting a default value of 0.85 for the IDs in the same `<MAX>` aggregation in case the corresponding RDF properties were not available. This lowered the negative effect of a single incorrect identifier while preserving a high rating in this `<MAX>` aggregation whenever a matching value is found. After this improvement, only 2 links

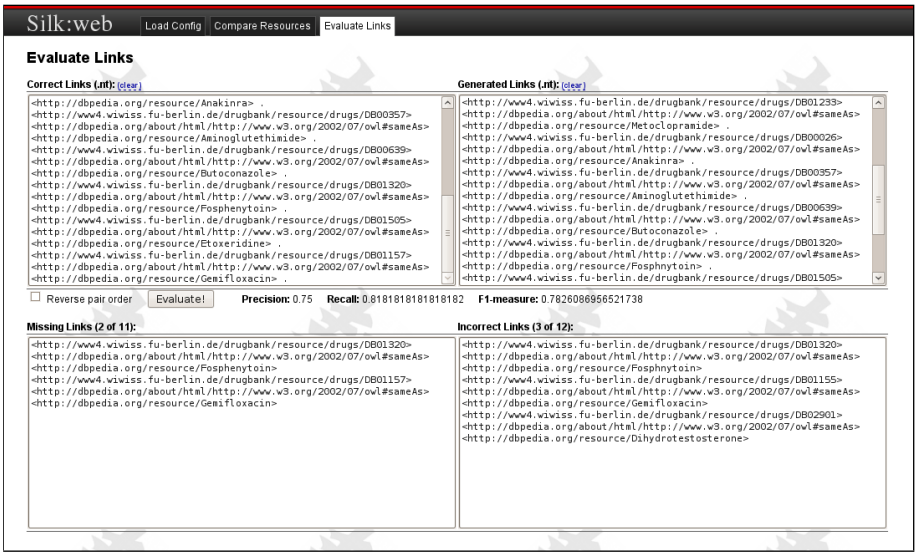

Fig. 5. Evaluating linksets with the Silk web interface

were missing, which means that we now reached a Recall value of 0.95 and an F_1-measure of 0.96.

4 Web of Data – Link Maintenance Protocol

Changes or additions in either of the interlinked datasets can invalidate existing links or imply the need to generate new ones. With the *Web of Data – Link Maintenance Protocol (WOD-LMP)*, we propose a solution to this problem.

The WOD-LMP protocol automates the communication between two cooperating Web data sources. It assumes two basic roles: *Link source* and *link target*, where the link source is a Web data source that publishes RDF links pointing at data published by the target data source. The protocol covers the following three use cases:

4.1 Link Transfer to Target

In the simplest use case, a link source wants to send a set of RDF links to the target data source so that the target may keep track of incoming links and can decide whether it wants to set back-links. Afterwards, the source wants to keep the target informed about subsequent updates (i.e. additions and deletions) to the transferred links. To achieve the transfer of the initial set of links and of subsequently generated ones, a *Link Notification* message is sent to the target data source. This notification includes the generated links along with the URL of the WOD-LMP protocol endpoint at the source side. Single deletion of links

by the source is communicated to the target in a *Link Deletion Notification* message, which in turn contains the link triples to be deleted.

4.2 Request of Target Change List

In this use case, the source data source asks the target to supply a list of all changes that have occurred to RDF resources in a target dataset within a specific time period. The source may then use the provided change information for periodic link recomputation. The protocol also provides requesting only additions, updates or deletions of resources. WOD-LMP uses incremental sequence numbers to identify resource changes. The changes are requested by the remote data source by sending a *Get Changes* message, which contains both the desired change sequence number range as well as the desired change type filter options. The target replies to this with a *Change Notification*, which lists the requested changes together with their corresponding sequence numbers and change types. If no upper sequence number is supplied, the target sends all changes to the latest change.

This case of selective link recomputation requires periodic polling of the remote data source by the source but has the advantage of working without maintaining a persistent relationship between the linked data sources.

4.3 Subscription of Target Changes

The protocol also supports fine-grained link recomputation by monitoring the resources in the target dataset that were used to compute links. As illustrated in Figure 6, the source informs the target dataset via a *Link Notification* message about a group of generated links and for each transferred link, supplies the URIs of the resources in the target dataset that were used to compute the link. The target saves this information and monitors the resources. If one of them changes or is deleted, the target notifies the source about these changes by sending a *Change Notification* message. The source may then use this information to recompute affected links and possibly delete invalidated ones. In this case, it notifies the target about deleted links with a *Link Deletion Notification*, which cancels the subscription of resources relevant to these links.

The implementation of the WOD-LMP protocol is based on SOAP. The complete specification of the protocol is available at `http://www4.wiwiss.fu-berlin.de/bizer/silk/wodlmp/`.

The WOD-LMP protocol is used to maintain the links between DBpedia and DrugBank. Links generated on the DrugBank side are sent and integrated into DBpedia, while DBpedia notifies the DrugBank Silk instance about changes to subscribed resources. This synchronization will become especially important as DBpedia will start to utilize the Wikipedia live update stream to continuously extract data from changed Wikipedia pages. Thus, DBpedia resources will be continuously updated to match Wikipedia, while at the same time the DrugBank Silk instance will be able to maintain and recompute links to DBpedia.

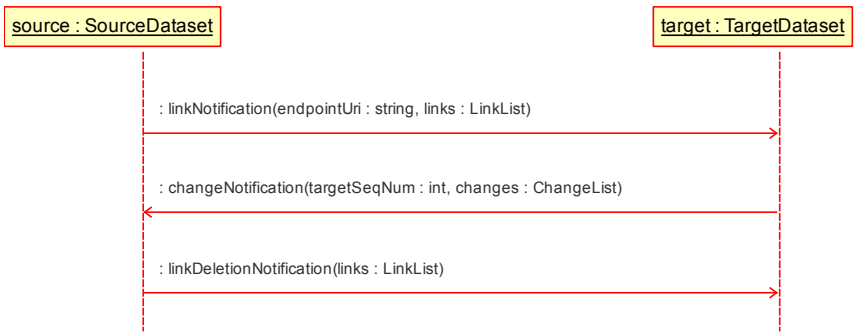

Fig. 6. Subscribing to resource changes in the target data source

5 Implementation

Silk is written in Python and is run from the command line. When generating linksets, Silk is started as a batch process. It runs as a daemon when serving the web interface or WOD-LMP protocol endpoints. The framework may be downloaded from Google Code[9] under the terms of the BSD license. For calculating string similarities, a library from Febrl[10], the Freely Extensible Biomedical Record Linkage Toolkit, is used, while Silk's pre-matching features are achieved using the search engine library Xapian[11]. The web interface was realized with the Werkzeug[12] toolkit, while the link maintenance protocol endpoints use the free soaplib[13] library for the exchange of SOAP messages.

6 Related Work

There is a large body of related work on record linkage [7] and duplicate detection [9] within the database community as well as on ontology matching [10] in the knowledge representation community. Silk builds on this work by implementing similarity metrics and aggregation functions that proved successful within other scenarios. What distinguishes Silk from this work is its focus on the Linked Data scenario where different types of semantic links should be discovered between Web data sources that often mix terms from different vocabularies and where no consistent RDFS or OWL schemata spanning the data sources exist.

Related work that also focuses on Linked Data includes Raimond et al. [11] who propose a link discovery algorithm that takes into account both the similarities of web resources and of their neighbors. The algorithm is implemented within the GNAT tool and has been evaluated for interlinking music-related

[9] `http://silk.googlecode.com`
[10] `http://sourceforge.net/projects/febrl`
[11] `http://xapian.org`
[12] `http://werkzeug.pocoo.org`
[13] `http://trac.optio.webfactional.com/`

datasets. In [12], Hassanzadeh et al. describe a framework for the discovery of semantic links over relational data which also introduces a declarative language for specifying link conditions. Their framework is meant to be used together with relational database to RDF wrappers like D2R Server or Virtuoso RDF Views. Differences between LinQL and Silk-LSL are the underlying data model and Silk's ability to more flexibly combine metrics through aggregation functions. A framework that deals with instance coreferencing as part of the larger process of fusing Web data is the KnoFuss Architecture proposed in [13]. In contrast to Silk, KnoFuss assumes that instance data is represented according to consistent OWL ontologies.

Furthermore, approaches to track changes and updates in Linked Data sources include PingtheSemanticWeb[14], a central registry for Web of Data documents which offers XML-RPC and REST APIs to notify the service about new or changed documents. A further approach to making change information available is proposed by Auer et al. and implemented in Triplify[14]. Similar to the second WOD-LMP use case, change information is requested on a peer-to-peer basis instead of being aggregated into a central registry, such as PingtheSemanticWeb. This approach is also implemented by DSNotify[15], which runs as an add-on to a local data source and uses indexes to track resource changes. DSNotify supports the active notification of subscribers as well as providing change data on demand. It further uses heuristics to determine the cause of a resource change and whether a deleted link target has become available under a different URI.

7 Conclusion

We presented the Silk framework, a flexible tool for discovering links between entities within different Web data sources. The Silk-LSL link specification language was introduced and its applicability was demonstrated within a life science use case. We then proposed the WOD-LMP protocol for synchronizing and maintaining links between continuously changing Linked Data sources.

Future work on Silk will focus on the following areas: We will implement further similarity metrics to support a broader range of linking use cases. To assist users in writing Silk-LSL specifications, machine learning techniques could be employed to adjust weightings or optimize the structure of the matching specification. Finally, we will evaluate the suitability of Silk for detecting duplicate entities within local datasets instead of using it to discover links between disparate RDF data sources.

The value of the Web of Data rises and falls with the amount and the quality of links between data sources. We hope that Silk and other similar tools will help to strengthen the linkage between data sources and therefore contribute to the overall utility of the network.

The complete Silk – LSL language specification, WoD Link Maintenance Protocol specification and further Silk usage examples are found on the Silk project website at `http://www4.wiwiss.fu-berlin.de/bizer/silk/`.

[14] `http://pingthesemanticweb.com`

References

1. Berners-Lee, T.: Linked Data - Design Issues. http://www.w3.org/DesignIssues/LinkedData.html
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Journal on Semantic Web and Information Systems (in press), 2009.
3. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/
4. Bizer, C., et al.: DBpedia - A Crystallization Point for the Web of Data. Journal of Web Semantics: Sci. Serv. Agents World Wide Web, doi:10.1016/j.websem.2009.07.002, 2009.
5. Jentzsch, A., et al.: Enabling Tailored Therapeutics with Linked Data. In: Proceedings of the 2nd Workshop about Linked Data on the Web, 2009.
6. Jaro, M.: Advances in Record-linkage Methodology as Applied to the 1985 Census of Tampa, Florida. Journal of the American Statistical Society, 84(406):414-420, 1989.
7. Winkler, W.: Overview of Record Linkage and Current Research Directions. Bureau of the Census - Research Report Series, 2006.
8. Zhong, J., et al.: Conceptual Graph Matching for Semantic Search. The 2002 International Conference on Computational Science, 2002.
9. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering, 19(1):1-16, 2007.
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg, 2007.
11. Raimond, Y., Sutton, C., Sandler, M.: Automatic Interlinking of Music Datasets on the Semantic Web. In: Proceedings of the 1st Workshop about Linked Data on the Web, 2008.
12. Hassanzadeh, O., et al.: Semantic Link Discovery Over Relational Data. Proceedings of the 18th ACM Conference on Information and Knowledge Management, 2009.
13. Nikolov, A., et al.: Integration of Semantically Annotated Data by the KnoFuss Architecture. In: 16th International Conference on Knowledge Engineering and Knowledge Management, 265-274, 2008.
14. Auer, S., et al.: Triplify – Light-Weight Linked Data Publication from Relational Databases. In: Proceedings of the 18th International World Wide Web Conference, 2009.
15. Haslhofer, B., Popitsch, N.: DSNotify – Detecting and Fixing Broken Links in Linked Data Sets. In: Proceedings of 8th International Workshop on Web Semantics, 2009.