Silk Server - Adding missing Links while consuming Linked Data

Robert Isele, Anja Jentzsch, and Christian Bizer

Freie Universität Berlin, Web-based Systems Group
Garystr. 21, 14195 Berlin, Germany
robertisele@googlemail.com
mail@anjajentzsch.de
chris@bizer.de

Abstract. The Web of Linked Data is built upon the idea that data items on the Web are connected by RDF links. Sadly, the reality on the Web shows that Linked Data sources set some RDF links pointing at data items in related data sources, but they clearly do not set RDF links to all data sources that provide related data. In this paper, we present Silk Server, an identity resolution component, which can be used within Linked Data application architectures to augment Web data with additional RDF links. Silk Server is designed to be used with an incoming stream of RDF instances, produced for example by a Linked Data crawler. Silk Server matches the RDF descriptions of incoming instances against a local set of known instances and discovers missing links between them. Based on this assessment, an application can store data about newly discovered instances in its repository or fuse data that is already known about an entity with additional data about the entity from the Web. Afterwards, we report on the results of an experiment in which Silk Server was used to generate RDF links between authors and publications from the Semantic Web Dog Food Corpus and a stream of FOAF profiles that were crawled from the Web.

Keywords: Linked Data, Link Discovery, Identity Resolution

1 Introduction

The Web of Linked Data [3] is built upon two simple ideas: Structured data is published on the Web using dereferencable URIs to represent data items wherein related data items are connected using RDF links. At its present state, the Web of Linked Data contains only a fraction of the links that would be desirable to be set[1]. According to Rodriguez [14], the Web of Data graph merely consists of two weakly connected components with a large diameter of 10 and an average path length of 3.4. A Linked Data application which wants to exploit the relationships

[1] http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics

between data items from different data sources thus might want to augment Web data with additional links before using it in the application context.

In order to tackle this problem, we provide the Silk Link Discovery Framework [15]. Silk generates RDF links between data items based on user-provided link specifications which are expressed using the Silk Link Specification Language (Silk-LSL). Silk is provided in three different variants which address different use cases:

- *Silk Single Machine* is used to generate RDF links between two datasets on a single machine.
- *Silk MapReduce* is based on Hadoop and enables Silk to scale out to very big datasets by distributing the link generation to multiple machines.
- *Silk Server* can be used as an identity resolution component within applications that consume Linked Data from the Web.

This paper is focused on *Silk Server* which has been recently added as a new component to the Silk Link Discovery Framework. *Silk Single Machine* and *Silk MapReduce* are described on the Silk homepage[2].

Silk Server is designed to be used with an incoming stream of RDF instances, produced for example by a Linked Data crawler such as LDSpider[3]. *Silk Server* matches incoming instances against a local set of known instances and discovers missing links between them. Incoming instances which do not match a known instance are added to the local set of instances continuously. Based on this assessment, an application can store data about newly discovered instances in its repository or fuse data that is already known about an entity with additional data about the entity from the Web.

The main features of the *Silk Server* are:

- It runs as an HTTP server and offers a REST interface [9] that allows applications to check whether an entity that has been discovered on the Web is already known to the system. If the entity is already known, Silk Server returns an RDF link pointing at the URI identifying the known entity.
- It provides a flexible, declarative language for specifying the conditions which determine whether an entity is already known to the system.
- It is high-performing by holding the data about all known instances in an in-memory cache, which is updated as soon as new instances are discovered. In addition, the performance can be further enhanced using a blocking feature.

The paper is structured as follows: Section 2 explains the role Silk Server can play within Linked Data application architectures. In Section 3, the architecture and workflow of the Silk Server are presented. Section 4 reports on the results of an experiment in which Silk Server was used to generate RDF links between the data about authors and publications from the Semantic Web Dog Food Corpus [12] and a stream of FOAF[4] profiles that were crawled from the Web. Section 5 compares Silk Server with related work.

[2] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`
[3] `http://code.google.com/p/ldspider/`
[4] `http://www.foaf-project.org`

2 Silk Server within Linked Data Application Architectures

This section discusses the role of *Silk Server* within Linked Data application architectures. Figure 1 gives an overview of the architecture of a fully-fledged Linked Data application which operates on top of the public Web of Linked Data [3].

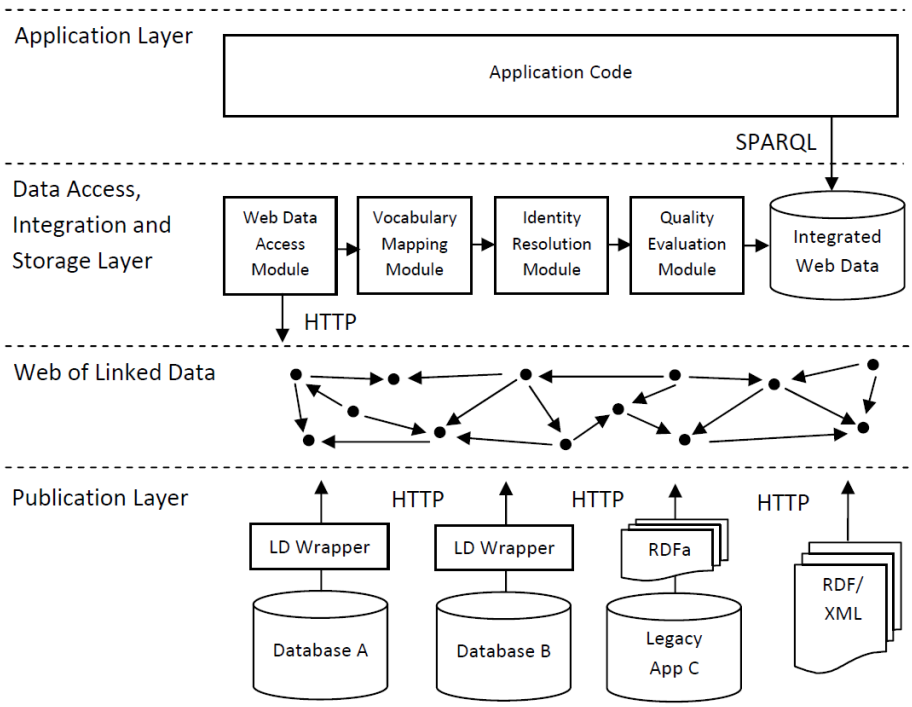

Fig. 1. Schematic Architecture of Linked Data Applications

All data that is published on the Web according to the Linked Data principles becomes part of a *giant global graph* - the Web of Linked Data. This logical graph is depicted in the *Web of Linked Data* layer in Figure 1. Applications that utilize this graph might implement the modules (or a subset of the modules) depicted in the *Data Access, Integration and Storage Layer*. In the following, we will describe the functionality of the different modules.

1. **Web Data Access Module** The basic means to access Linked Data on the Web is to dereference HTTP URIs into RDF descriptions and to discover additional data by traversing RDF links. Such link traversal can for instance

be implemented using readily available Linked Data crawlers such as LDspider. In addition, the data access module might download RDF data set dumps or utilize SPARQL endpoints (for an overview about SPARQL-based distributed query architectures please refer to [10]). Data set dumps and SPARQL endpoints might be discovered by the data access module by relying on VOID descriptions [1] and Semantic Web Sitemaps [6] published on the Web by the data sources.

2. **Vocabulary Mapping Module** Different Linked Data sources often use different RDF vocabularies to represent the same type of information. In addition, data sources often use a mixture of terms from widely-used vocabularies, such as FOAF, SIOC or Dublin Core, and proprietary terms to represent data. In order to understand as much Web data as possible, Linked Data applications might employ a vocabulary mapping module to translate terms from different vocabularies into the application's target schema. This tranlation can rely on `owl:sameClass` or `owl:sameProperty` mappings as well as on `rdfs:subClass` and `rdfs:subProperty` statements that are published on the Web together with the vocabulary definitions or can employ more expressive mapping languages and discovery features, as for instance provided by the R2R Famework[5].
3. **Identity Resolution Module** Different Linked Data sources use different URIs to identify the same entity in order to enable clients to directly retrieve data describing the entity from the different sources using the HTTP protocol. In addition, data sources might publish `owl:sameAs` links pointing at URIs that are used by other data sources to identify the same entity. In contrast, it is often desirable for Linked Data applications to locally use only a single URI as the subject of all RDF statements about an entity while keeping track of the provenance of the statements. Thus in addition to using the `owl:sameAs` statements that are part of the ordinal Web data, applications might also employ an local identity resolution module, which generates additional `owl:sameAs` statements and interlinks newly discovered data about entities with data about them that is already known by the application. Silk Server provides this functionality and can thus be used as an identity resolution module within Linked Data applications.
4. **Quality Evaluation Module** Due to the open nature of the Web, any Web data needs to be treated with suspicion and Linked Data applications should thus consider RDF statements which they discover on the Web as claims by a specific source rather than as facts. In order to determine which claim to accept and trust, Linked Data applications should employ a data quality evaluation module. This module may filter RDF SPAM, prefer data from sources that are known for good quality and optionally resolves data conflicts [4]. An overview about the different information quality assessment heuristics that can be used by the quality evaluation module is given in [2].
5. **Integrated Web Data** At the end of the processing pipeline, the cleaned Web data is stored in a repository together with provenance information to

[5] `http://www4.wiwiss.fu-berlin.de/bizer/r2r/`

be used by the application layer. A commonly used model for representing Web data together with provenance information are Named Graphs [5]. Different vocabularies for exposing provenance information are currently compared by the W3C Provenance Incubator Group[6].

3 The Silk Server

Silk Server is an identity resolution component that can be used within Linked Data application architectures. It runs as an HTTP server and matches instances of an incoming RDF stream against a local set of known instances based on user-provided link specifications. In the following, we will describe the architecture of the Silk Server as well as the general linking workflow.

3.1 Architecture

The Silk Server is composed of the following three layers:

The in-memory **instance cache** builds the bottom layer which holds all known instances and keeps track of newly discovered instances. For each instance, the values of all relevant properties which are later required for the comparison are stored. As soon as a new instance is discovered, it is added to the instance cache. This enables the server to generate links to the newly discovered instance in future requests. Currently, the instance cache is held in memory, but can be replaced by a persistent cache in future versions of Silk. The current implementation of the instance cache can fit approximately 10 million instances into 8GB of main memory.

The **Silk Linking Engine** generates the links based on a set of link specifications and forms the central part of Silk Server. The details of the link generation process are covered in Section 3.3.

The **REST interface** enables applications to commit newly discovered resources and receive the generated links. New resources are accepted through an HTTP POST request using one of the supported RDF serialization formats, such as RDF/XML or N-Triples. The response contains all generated links optionally including statements declaring unknown instances i.e. instances for which no link could be generated. The server can process multiple requests in parallel.

3.2 Data Processing Workflow

Figure 2 illustrates the Silk Server workflow.

[6] http://www.w3.org/2005/Incubator/prov/

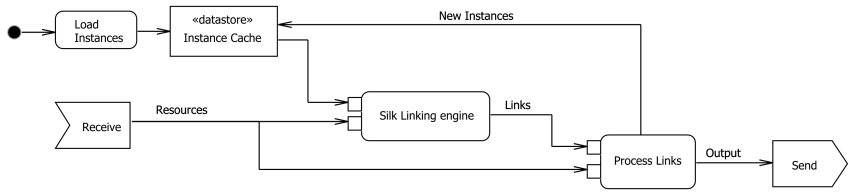

Fig. 2. Silk Server Workflow

The Silk Server workflow is divided into 2 phases:

In the **Setup phase** the server loads all data sets which are specified by the user-provided link specifications. For each link specification, one instance cache is used to hold the part of the data that is later required for matching instances.

The **Service phase** starts as soon as all data sets have been loaded. If an application discovers new instances on the Web, it issues a request to the server containing the newly found data. The request may contain multiple instances with different types. On receiving the request, the Server matches the given instances with its link specifications. If a link specification can be applied to a specific instance, the server forwards it to the Silk Linking Engine. The Silk Linking Engine generates links for the given instances based on the corresponding link specifications as described in Section 3.3.

The generated links are processed by the server to find the set of instances which are not matched by any known instance. The instance cache is updated with the set of unmatched instances. Thus, in future request the server will also generate links to the newly found unmatched instances.

After the update has been completed, the generated links along with statements containg the unmatched instances are returned.

3.3 The Silk Linking Engine

When receiving new instances to be matched, the Silk Linking Engine generates new buckets consisting of a provided instance and a set of instances from the cache. Each bucket is processed in 3 subsequent phases:

The optional **Blocking** phase partitions the incoming buckets into clusters. Since comparing every source resource to every single target resource results in a number of $n * m$ comparisons (n being the number of source resources, m the number of target resources), blocking can be used to reduce the number of comparisons. Blocking partitions similar data items into clusters limiting the comparisons to items in the same cluster. For example, given a set of books to be compared, in order to reduce the number of comparisons, one could block the books by publisher. In this case only books from the same publisher will be compared.

The **Link Generation** phase reads the incoming buckets and computes a similarity value for each pair of instances. The incoming data items, which might be allocated to a cluster by the preceding blocking phase, are written to an internal cache. From the cache, pairs of data items are generated. If blocking is disabled, this will generate the complete cartesian product of the two data sets. If blocking is enabled, only data items from the same cluster are compared. For each pair of data items, the link condition is evaluated, which computes a similarity value between 0 and 1. Each pair generates a preliminary link with a confidence according to the similarity of the source and target data item.

The **Filtering** phase filters the incoming links in two stages: In the first stage, all links with a lower confidence than the user-defined threshold are removed. In the second stage, all links which originate from the same subject are grouped together. If a limit is defined on the number of links per subject, only the links with the highest confidence are forwarded to the output.

3.4 Implementation

Silk Server is implemented in Scala[7] and runs as a Servlet on the Jetty Web Server[8]. The REST interface has been realized using the Lift Web Framework[9]. The Silk Link Discovery Framework including Silk Server can be downloaded from the project homepage[10] under the terms of the Apache Software License.

4 Evaluation

This section reports on the results of an experiment in which we used Silk Server to generate RDF links between authors and publications from a Semantic Web Dog Food Corpus dump and a stream of FOAF profiles that we crawled from the Web. Semantic Web Dog Food Corpus publishes information on people and publications from Semantic Web conferences. FOAF is a widely used vocabulary to describe persons, their connections, projects, publications and interests. Twitter is a social networking and microblogging website which provides user information as RDFa. Given these different sources for information on persons, the experiment aims at linking duplicate person descriptions. In the following, we explain the Silk-LSL specification used by Silk Server in the experiment; we then first describe the setup of the experiment and finally report on and discuss the results of the experiment.

[7] `http://scala-lang.org`
[8] `http://jetty.codehaus.org`
[9] `http://liftweb.net`
[10] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`

4.1 The Link Specification used

Figure 3 contains the link configuration used in the experiment for linking data items describing the same person. The complete link configuration for discovering RDF links between persons as well as publications is available online[11].

The involved data sources for this experiment are the Semantic Web Dog Food Corpus dump (line 5) and an RDF input stream (line 9).

A link configuration may contain several link specifications if links for different types of data items should be generated. Silk Server will set `owl:sameAs` links between duplicates as configured in line 16.

Link specifications contain link conditions which define the conditions that data entities must fulfill in order to be interlinked. Link conditions may apply similarity metrics to multiple property values of an entity or related entities. The resulting similarity scores can be combined and weighted using various similarity aggregation functions.

Link Conditions The link condition specifies how two data entities are compared for similarity. It consists of a number of comparison operators which are combined using aggregation functions.

A comparison operator evaluates two inputs and computes their similarity based on a user-defined metric. Silk provides several similarity metrics including string, numeric, date, and URI similarity. String comparison methods cover the most common ones like Jaro, Jaro-Winkler and Levenshtein. Silk can easily be enhanced with new metrics.

Multiple comparisons can be aggregated using a specific aggregation method by using the `<Aggregate>` directive.

In the given experiment's link condition we compute similarity values for the FOAF names, homepages, and mailbox hash sums (lines 24 to 45). The overall similarity value of two data entities is derived by the weighted average of the similarity values of all comparisons. To identify a person uniquely, either a homepage or a mailbox hash sum is required. Thus, two persons are considered equal if both names and either the homepage or the mailbox hash sum match.

Some comparison operators might be more relevant for the correct establishment of a link between two resources than others and can therefore be weighted higher. If no weight is supplied, a default weight of 1 will be assumed. As a person may be known under different names, matching homepages or mailbox hash sums are more important and therefore weighted higher (line 35).

Filtering The generated links can be filtered by using the `<Filter>` directive. A threshold for the minimum similarity of two data items required to generate a link between them can be defined (line 47). The number of links originating from a single data item can be limited. Only the highest-rated links per source data item will remain after the filtering.

[11] `http://www4.wiwiss.fu-berlin.de/bizer/silk/linkspecs/persons_and_publications.xml`

```
01 <Silk>
02   <Prefixes>
...
03   </Prefixes>
04   <DataSources>
05     <DataSource id="sw_dog_food" type="file">
06       <Param name="file" value="semantic_web_dog_food.rdf"/>
07       <Param name="format" value="RDF/XML"/>
08     </DataSource>
09     <DataSource id="input_stream" type="rdf">
10       <Param name="format" value="N-TRIPLE"/>
11       <Param name="input" value=""/>
12     </DataSource>
13   </DataSources>
14   <Interlinks>
15     <Interlink id="persons">
16       <LinkType>owl:sameAs</LinkType>
17       <SourceDataset dataSource="input_stream" var="a">
18         <RestrictTo> ?a rdf:type foaf:Person . </RestrictTo>
19       </SourceDataset>
20       <TargetDataset dataSource="sw_dog_food" var="b">
21         <RestrictTo> ?b rdf:type foaf:Person . </RestrictTo>
22       </TargetDataset>
23       <LinkCondition>
24         <Aggregate type="average">
25           <Aggregate type="max" required="true">
26             <Compare metric="jaroWinkler">
27               <TransformInput function="lowerCase">
28                 <Input path="?a/foaf:name"/>
29               </TransformInput>
30               <TransformInput function="lowerCase">
31                 <Input path="?b/foaf:name"/>
32               </TransformInput>
33             </Compare>
34           </Aggregate>
35           <Aggregate type="max" weight="2" required="true">
36             <Compare metric="jaroWinkler">
37                 <Input path="?a/foaf:homepage"/>
38                 <Input path="?b/foaf:homepage"/>
39             </Compare>
40             <Compare metric="jaroWinkler">
41               <Input path="?a/foaf:mbox_sha1sum"/>
42               <Input path="?b/foaf:mbox_sha1sum"/>
43             </Compare>
44           </Aggregate>
45         </Aggregate>
46       </LinkCondition>
47       <Filter threshold="0.9"/>
48     </Interlink>
49   </Interlinks>
50 </Silk>
```

Fig. 3. Example: Interlinking persons in FOAF profiles

4.2 Setup of the Experiment

For the experiment, we loaded the Semantic Web Dog Food Corpus into the Silk Server. The Semantic Web Dog Food Corpus contains profiles for 3,739 persons from which 2,580 provide either a homepage or a mailbox hash which is required to uniquely identify them. We have set up a Linked Data crawler which takes a number of FOAF profile URIs as seeds and follows linked profiles. The crawled documents are forwarded to Silk Server which generates `owl:sameAs`

links to known persons from the Semantic Web Dog Food Corpus. All generated links have been written to an ouput file which has been analyzed for the results presented in section 4.3.

The crawler was also used to traverse the RDFa of Twitter accounts for which the server identified the corresponding persons in the Semantic Web Dog Food Corpus if any.

In order to show the flexibility of Silk Server, the link configuration was further enhanced to also match publications. For this purpose the crawler was employed to also follow publication links in addition to FOAF profiles.

4.3 Results of the Experiment

Generated links to FOAF profiles At first, we evaluated how exhaustive the found links are. For this purpose, we exploited the fact that for 56 persons the Semantic Web Dog Food Corpus already sets links to their FOAF profile. For 51 of these persons, Silk Server was able to reconstruct links from the stream. For some persons even multiple duplicated profiles could be identified. For example e.g. in addition to Tom Heath's[12] official FOAF profile `<http://tomheath.com/id/me>`, Silk Server also identified him on `<http://www.eswc2006.org/people/#tom-heath>`. Because in some cases, Silk Server found a link to another profile than the one given in the data set, we checked all links manually for correctness. Thereby, all generated links have been found to be correct.

Next, we evaluated for how many persons in the Semantic Web Dog Food Corpus, the server was able to generate links to a FOAF profile. In total, Silk Server was able to find profiles for 228 persons in the data set. Thus, Silk Server was able to discover links to the FOAF profile of additional 177 persons for which the Semantic Web Dog Food Corpus did not contain a link yet.

Generated links to Twitter accounts For 89 persons in the Semantic Web Dog Food Corpus, Silk Server was able to find a corresponding Twitter account. Silk Server was able to detect more than one account for persons holding multiple accounts. For example, it found that Ralph Hodgson[13] not only uses the account `http://twitter.com/ralphtq` but also the account `http://twitter.com/oegovnews`.

Generated links to publications For 37 publications in the Semantic Web Dog Food Corpus Silk Server was able to find the corresponding publication in the Web of Data. The number of links is lower than the number of found FOAF profiles because many persons do not link their publications in their profile.. One exception is the Digital Enterprise Research Institute (DERI), which publishes the meta data about all publications as RDF[14].

[12] `http://data.semanticweb.org/person/tom-heath`
[13] `http://data.semanticweb.org/person/ralph-hodgson`
[14] `http://www.deri.ie/publications/`

5 Related Work

Discovering links between data items across data sets requires record linkage and duplicate detection techniques. There is a large body of related work on these topics within the database community [16][7] as well as on ontology matching in the knowledge representation community [8].

Silk builds on the research results from within these communities. Silk can be used in scenarios where different types of links should be discovered between Web data sources which often make use of terms from different vocabularies.

Besides Silk, there are two related tools for generating RDF links:

LinQuer [11] is a tool for semantic link discovery over relational data, based on string and semantic matching techniques and their combinations. The LinQuer framework consists of LinQL, a declarative language that allows specification of linkage requirements in a wide variety of applications. The framework rewrites LinQL queries into standard SQL queries that can be run over relational data sources. LinQuer is meant to be used together with relational databases to RDF wrappers such as D2R Server[15] or Virtuoso RDF Views[16].

Related work that also focuses on Linked Data includes Raimond et al. [13] 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 data sets.

While LinQuer and GNAT only allow batch processing, Silk Server is the first identity resolution component that works on an on-demand fashion and can be used together with RDF data streams.

The EU-funded project OKKAM[17] offers an Entity Name System (ENS), which supportes the storage and reuse of global entity identifiers. While OKKAM ENS contains several matching modules per default, it does not provide a flexible and comprehensive link specification language.

The RKBExplore sameAs service[18] is targeted at providing a unified view over multiple data sources by managing owl:sameAs links to identify duplicate URIs. In contrast to Silk Server the links are not generated based on user-defined link specifications, but must be provided to the system from external sources.

6 Conclusion

Vint Cerf, the inventor of the internet, said in his keynote speech at 19th International World Wide Web Conference (WWW2010) that in the age of the internet where everything should be connected, he would also expect database management systems to automatically connect new records that are added to a database with all related entities that are already stored in the database. With

[15] `http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/`
[16] http://virtuoso.openlinksw.com/whitepapers/relational%20rdf%20views%20mapping.html
[17] `http://www.okkam.org/`
[18] `http://www.rkbexplorer.com/sameAs/`

Silk Server, we make a first step to provide such functionality for the Linked Data context.

7 Acknowledgments

This work was supported in part by Vulcan Inc. as part of its Project Halo (www.projecthalo.com) and by the EU FP7 project LOD2 - Creating Knowledge out of Interlinked Data (http://lod2.eu/, Ref. No. 257943).

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J. Describing linked datasets. In *Proc. of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.
2. Christian Bizer and Richard Cyganiak. Quality-driven information filtering using the wiqa policy framework. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 7(1):1–10, 2009.
3. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
4. Bleiholder, J., Naumann, F. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.
5. Carroll, J., Bizer, C., Hayes, P., Stickler, P. Named graphs. *Journal of Web Semantics*, 3(4):247–267, 2005.
6. Cyganiak, R., Delbru, R., Stenzhorn, H., Tummarello, G., Decker, S. Semantic sitemaps: Efficient and flexible access to datasets on the semantic web. In *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, 2008.
7. Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 19(1):1–16, 2007.
8. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
9. Roy T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.
10. Olaf Hartig and Andreas Langegger. A database perspective on consuming linked data on the web. *Datenbank Spektrum*, to appear.
11. Oktie Hassanzadeh, Reynold Xin, Rene J. Miller, Anastasios Kementsietsidis, Lipyeow Lim, and Min Wang. Linkage query writer, 2009.
12. Knud Möller, Tom Heath, Siegfried Handschuh, and John Domingue. Recipes for semantic web dog food - the eswc and iswc metadata projects. In *ISWC/ASWC*, pages 802–815, 2007.
13. Y. Raimond, C. Sutton, and M. Sandler. Automatic Interlinking of Music Datasets on the Semantic Web . In *Proc. of the 1st Linked Data on the Web Workshop*, 2008.
14. Marko A. Rodriguez. A graph analysis of the linked data cloud. *CoRR*, abs/0903.0194, 2009.
15. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*, pages 650–665, 2009.
16. William E. Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census, 2006.