Silk – Generating RDF Links while publishing or consuming Linked Data

Anja Jentzsch, Robert Isele, Christian Bizer

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

Abstract. The central idea of the Web of Data is to interlink data items using RDF links. However, in practice most data sources are not sufficiently interlinked with related data sources. The Silk Link Discovery Framework addresses this problem by providing tools to generate links between data items based on user-provided link specifications. It can be used by data publishers to generate links between data sets as well as by Linked Data consumers to augment Web data with additional RDF links. In this poster we present the Silk Link Discovery Framework and report on two usage examples in which we employed Silk to generate links between two data sets about movies as well as to find duplicate persons in a stream of data items that is crawled from the Web.

Keywords: Linked Data, Link Discovery, Identity Resolution

1 Introduction

The Web of Data is built on the fundamental idea that data items are published using dereferencable URIs wherein related data items are connected using RDF links [1]. While the Web of Data is constantly growing[1], it still forms a weakly interlinked graph and contains only a small fraction of the RDF links that could in theory be set [2]. In order to tackle this problem, we provide the Silk Link Discovery Framework. Silk generates RDF links between data items based on user-provided link specifications. Silk specifications are expressed using a declarative language and define the conditions that data items must fulfill in order to be interlinked.

The Silk Link Discovery Framework addresses the following two use cases: It can be used by data providers to generate RDF links pointing at existing datasets on the Web. These RDF links can then be published together with the primary data sets on the Web. Furthermore, Silk can be used as an identity resolution component within applications that consume Linked Data from the Web in order to augment Web data with additional RDF links which have not been set by the data providers.

Silk is provided in three different variants which address different use cases:

[1] http://lod-cloud.net/

- *Silk - Single Machine* is used to generate RDF links on a single machine. The datasets that should be interlinked can either reside on the same machine or on remote machines which are accessed via the SPARQL protocol. *Silk - Single Machine* provides multithreading and caching. In addition, the performance can be further enhanced using an optional blocking feature.
- *Silk - Map Reduce* is used to generate RDF links between data sets using a cluster of multiple machines. *Silk - Map Reduce* is based on Hadoop and can for instance be run on Amazon Elastic MapReduce. *Silk - Map Reduce* 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. *Silk - Server* provides an HTTP API for matching instances from an incoming stream of RDF data against a local set of known instances. It can be used for instance together with a Linked Data crawler to populate a local duplicate-free cache with data from the Web.

The Silk Link Discovery Framework is implemented in Scala[2] and can be downloaded from the project homepage[3] under the terms of the Apache Software License.

In the following, we will give an overview of the Silk Link Discovery Framework, report on two usage examples in which we employed the framework and present planned future work.

2 The Silk Link Discovery Framework

The Silk Link Discovery Framework consists of a console application used to interlink two data sets as well as of the Silk Server, an HTTP server, which receives an incoming RDF stream and creates links between data items. Both applications provide a flexible configuration language, the Silk Link Specification Language (Silk-LSL), to specify the conditions data items must fulfill in order to be interlinked [3]. For this purpose, the user may apply similarity metrics, such as string, date or URI comparison methods, to multiple property values of an entity or related entities. The resulting similarity scores can be combined and weighted using various similarity aggregation functions. A Silk link configuration may contain several link specifications if links for different types of data items should be generated.

The central part of the Silk Link Discovery Framework is the Silk Linking Engine, which generates the links between data items according to user-provided link specifications. The Silk Linking Engine processes the incoming data items, which are usually originating from a SPARQL endpoint, in subsequent phases:

The optional **Blocking** phase partitions the incoming data items into clusters. Since comparing every source resource to every single target resource results

[2] `http://scala-lang.org`
[3] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`

in a number of $n * m$ comparisons which might be time-consuming, 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.

The **Link Generation** phase reads the incoming data items and computes a similarity value for each pair. 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 value 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 data item are grouped together. The number of links per source item which are forwarded to the output, is specified by an optional link limit. If a link limit is defined, only the links with the highest confidence are forwarded.

3 Silk Server

Silk Server is an identity resolution component, that can be used within Linked Data application architectures to add missing RDF links to data that is consumed from the Web of Linked Data. It is designed to be used with an incoming stream of RDF instances, produced for example by a Linked Data crawler such as LDspider. Silk Server matches data describing 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.

4 Usage Examples

Silk has been employed in several scenarios to generate links between data sets on the Web of Data. In the following we report the results of employing *Silk - Single Machine* and *Silk - Server* within two usage scenarios.

4.1 Interlinking DBpedia movies with LinkedMDB directors

We employed *Silk - Single Machine* to interlink movies in DBpedia with the corresponding director in LinkedMDB. For this purpose, Silk was fed with the 50000 movies from DBpedia and 2500 directors from LinkedMDB. For each movie, Silk was configured to set a `dbpedia:director` link from the movie to its director.[4] A single PC with a Core 2 Duo CPU and 4GB of RAM needed 55 minutes to match the complete cartesian product resulting in 125 000 000 comparisions. Silk successfully identified 5900 links between a movie and its director. In order to increase the performance, the Linking Specification was extended to employ blocking on the director names to reduce the number of comparisions. As blocking may decrease the recall of the matching, we compared the generate links. With blocking enabled Silk was still able to identify 5857 links, resulting in a loss of less than one percent, while reducing the runtime of the matching considerable to only 7 minutes.

4.2 Identifying duplicate person descriptions in a data stream

In the Web of Data we can usually find different URIs which effectively identify the same entity, e.g. `<http://tomheath.com/id/me>` and `<http://www.eswc2006.org/people/#tom-heath>` describe the same person. We employed a Linked Data crawler to crawl the FOAF web and stream the traversed profiles to the Silk Server in order to identify duplicate persons and generate `owl:sameAs` links between them. For evaluation, we used the Semantic Web Dog Food data set[5] which already interlinks some of the contained persons with their corresponding FOAF profile. Among the 56 persons for which the Semantic Web Dog Food data set provides links to their FOAF profile, Silk Server was able to reconstruct 51 links from the stream. In addition, it was able to identify the FOAF profile of another 132 persons for which Semantic Web Dog Food did not provide a link to their profile yet.

5 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 (Grant No. 257943, http://lod2.eu/).

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
2. M. A. Rodriguez. A graph analysis of the linked data cloud. *CoRR*, abs/0903.0194, 2009.
3. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*, pages 650–665, 2009.

[4] The link specification can be found on `http://www4.wiwiss.fu-berlin.de/bizer/silk/linkspecs/dbpedia_linkedmdb_directors.xml`

[5] `http://data.semanticweb.org/`