

Silk Link Discovery Framework

Book assembled from <https://www.assembla.com/spaces/silk/wiki/> on 07-Feb-2013 by vladimir.alexiev@ontotext.com

- [Home](#)
- [Silk Variants](#)
 - [Silk Single Machine](#)
 - [Silk MapReduce](#)
 - [Silk Server](#)
- [Silk Workbench](#)
 - [Workspace](#)
 - [Linkage Rule Editor](#)
 - [Generating Links](#)
 - [Managing Reference Links](#)
 - [Learning Linkage Rules](#)
 - [REST API](#)
- [Basic Concepts](#)
 - [Data Sources](#)
 - [Linkage Rule](#)
 - [Path Input](#)
 - [Transformation](#)
 - [Comparison](#)
 - [Aggregation](#)
 - [Output](#)
 - [Reference Links](#)
- [Link Specification Language](#)
- [API](#)
 - [Modules](#)
 - [Tasks](#)
- [Silk Link Discovery Engine](#)
- [Performance](#)
- [Developer Documentation](#)
- [Publications](#)

About Silk

The Silk Link Discovery Framework is a tool for discovering relationships between data items within different Linked Data sources.

Data publishers can use Silk to set RDF links from their data sources to other data sources on the Web.

The official homepage of Silk can be found on <http://www4.wiwiw.fu-berlin.de/bizer/silk/>

Introduction

The Web of Data is built upon two simple ideas: First, to employ the RDF data model to publish structured data on the Web. Second, to set explicit [RDF links](#) between data items within different data sources. Background information about the Web of Data is found at the wiki pages of the [W3C Linking Open Data community effort](#), in the overview article [Linked Data – The Story So Far](#) and in the tutorial on [How to publish Linked Data on the Web](#).

The Silk Link Discovery Framework supports data publishers in accomplishing the second task. Using the declarative [Silk Link Specification Language \(Silk-LSL\)](#), developers 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 linkage rules may combine various similarity metrics and can take the graph around a data item into account, which is addressed using an RDF path language. Silk accesses the data sources that should be interlinked via the SPARQL protocol and can thus be used against local as well as remote SPARQL endpoints.

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

- [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 MapReduce](#) is used to generate RDF links between data sets using a cluster of multiple machines. Silk MapReduce is based on Hadoop and can for instance be run on Amazon Elastic MapReduce. Silk MapReduce 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 while keeping track of known entities. It can be used for instance together with a Linked Data crawler to populate a local duplicate-free cache with data from the Web.

All variants are based on the [Silk Link Discovery Engine](#) which offers the following features:

- Flexible, declarative language for specifying linkage rules
- Support of RDF link generation (owl:sameAs links as well as other types)
- Employment in distributed environments (by accessing local and remote SPARQL endpoints)
- Usable in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist
- Scalability and high performance through efficient data handling (speedup factor of 20 compared to Silk 0.2):
 - Reduction of network load by caching and reusing of SPARQL result sets
 - Multi-threaded computation of the data item comparisons (Over 1 billion comparisons per hour on a Core i7, 4GB RAM)
 - Optional blocking of data items

Silk Workbench

[Silk Workbench](#) is a web application which guides the user through the process of interlinking different data sources.

Silk Workbench offers the following features:

- It enables the user to manage different sets of data sources and linking tasks.
- It offers a graphical editor which enables the user to easily create and edit link specifications.
- As finding a good linking heuristics is usually an iterative process, the Silk Workbench makes it possible for the user to quickly evaluate the links which are generated by the current link specification.
- It allows the user to create and edit a set of reference links used to evaluate the current link specification.

Usage Examples

[Interlinking drugs in Sider and Drugbank](#)

Support and feedback

For questions and feedback please use the [Silk Google Group](#).

Download

All stable releases can be [downloaded](#).

The framework can be used under the terms of the [Apache Software License](#).

The latest source code is available in the [Git repository](#).

Version History

Version	Comment	Release Date
2.5.3	Various improvements and bugfixes.	2012-03-06
2.5.2	Added support for active learning of linkage rules.	2011-11-17
2.5.1	Various improvements to the Workbench.	2011-10-21
2.5	Added support for learning linkage rules.	2011-10-03
2.4.2	SPARQL/Update output. Dump file input. Improved indexing. Many improvements to the Silk Workbench	2011-07-22
2.4.1	Introduces per-comparison thresholds. New data transformations and distance measures including token-based distance measures. Improved Silk Workbench.	2011-07-01
2.4	Includes the new Silk Workbench, a web application which guides the user through the process of interlinking different data sources.	2011-06-01
2.3	Improved loading performance: Multiple parallel SPARQL queries are executed, while their results are merged on the fly. Improved matching performance: New blocking method offers greatly improved performance. Improved overall performance: Matching tasks are now executed concurrently to loading data instead of waiting for the complete data set to be loaded.	2011-01-31
2.2	Added Silk MapReduce	2010-10-06
2.1	Added Silk Server Added a geographical distance metric by Konrad Höffner (MOLE subgroup of Research Group AKSW, University of Leipzig) Bugfixes	2010-09-15
2.0	Reimplementation of the Silk framework in Scala. Improved scalability and performance. Prematching replaced by a more transparent blocking. Configuration is checked for consistency prior to link generation. Support of the OAEI Alignment format. (Anja and Robert)	2010-07-01
0.2	Added prematching of data items (Julius).	2009-

	The Silk 0.2 language specification is still available and Silk 0.2 framework can be downloaded from GoogleCode.	03-02
0.1	Initial Release of the Python version of the Silk framework (Julius and Chris)	2009-02-01

Silk Variants

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

- [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 MapReduce](#) is used to generate RDF links between data sets using a cluster of multiple machines. Silk MapReduce is based on Hadoop and can for instance be run on Amazon Elastic MapReduce. Silk MapReduce 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 while keeping track of known entities. It can be used for instance together with a Linked Data crawler to populate a local duplicate-free cache with data from the Web.

Silk Single Machine

Introduction

This document describes *Silk Single Machine* which can be 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.

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 linkage rules may combine various similarity metrics and can take the graph around data items into account, which is addressed using an RDF path language. Silk accesses the datasets that should be interlinked via the SPARQL protocol and can thus be used against local as well as remote SPARQL endpoints.

The main features of the Silk Single Machine are:

- Flexible, declarative language for specifying linkage rules
- Support of RDF link generation (owl:sameAs links as well as other types)
- Employment in distributed environments (by accessing local and remote SPARQL endpoints)
- Usable in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist
- Scalability and high performance through efficient data handling (Silk 2.0 is about 20 times faster than Silk 0.2):
 - Reduction of network load by caching and reusing of SPARQL result sets
 - Multi-threaded computation of the data item comparisons (Over 1 billion comparisons per hour on a Core i7, 4GB RAM)
 - Optional blocking directive which allows users to reduce the number of comparisons on cost of recall, if necessary.

In order to run Silk Single Machine, developers need to:

1. Have SPARQL access to the datasets that should be interlinked.
2. Write a link specification. For details refer to the documentation of the [Silk Link Specification Language](#).
3. Install the Silk framework as described in the *Installation and Usage* section.

Installation and Usage

Running Silk from the Command Line

In order to use Silk Single Machine, you need:

1. **Silk Link Discovery Framework:** Get the [most recent version](#).
2. **Java Runtime Environment:** The Silk Link Discovery Framework runs on top of the JVM. Get the most recent [JRE](#).

What to do:

1. Write a Silk-LSL configuration file to specify which resources should be interlinked.
2. Run Silk Single Machine:

```
java -DconfigFile=<Silk-LSL file> [-DlinkSpec=<Interlink ID>] [-Dthreads=<threads>] [-DlogQueries=(true/false)] [-Dreload=(true/false)] -jar silk.jar
```

3. Review Results: Open the output files designated in the Silk-LSL configuration and review the generated links.

Using the Silk API

In order to use the Silk API, you need:

1. Silk Link Discovery Framework. Check out the most recent version from the Silk SVN repository.

2. Java Development Kit The Silk Link Discovery Framework runs on top of the JVM. Get the most recent JDK from <http://java.sun.com>.
3. Maven is used for project management and build automation. Get it from: <http://maven.apache.org>.

What to do:

1. Write a Silk-LSL configuration file to specify which resources should be interlinked.
2. Call `executeFile` on the Silk object.

```
Silk.executeFile(configFile, [linkSpecId], [numThreads])
```

3. Review Results: Open the output files designated in the Silk-LSL configuration and review the generated links.

Silk MapReduce

Introduction

Silk MapReduce is used to generate RDF links between data sets using a cluster of multiple machines. Silk MapReduce is based on Hadoop and can for instance be run on Amazon Elastic MapReduce. Silk MapReduce enables Silk to scale out to very large datasets by distributing the link generation to multiple machines.

Usage

In order to run the Silk MapReduce, you need:

1. **Silk Link Discovery Framework.**
2. **Java Runtime Environment** The Silk Link Discovery Framework runs on top of the JVM. Get the most recent [\[\[http://java.com|JRE\]\]](http://java.com|JRE).

The Silk MapReduce linking workflow is divided into 2 phases:

Load phase

In the **Load phase** Silk loads all data sets which are specified by the user-provided link specifications into the instance cache.

```
hadoop jar silkmr.jar load configFile ouputDir [linkSpec]
```

The following parameters are accepted:

- `configFile` The path to the Silk configuration file, which contains the link specifications. For details on the Silk – Link Specification Language, please read the Specification.
- `ouputDir` The directory, where the instance cache will be written to. This will be the input directory of the Link Generation phase.
- `linkSpec` (optional) If given, only the specified link specification will be loaded. If not given, all link specifications in the provided configuration will be loaded.

Example:

```
hadoop jar silkmr.jar load ./config.xml ./cache
```

Link Generation phase

In the **Link Generation phase** Silk generates the links from the previously loaded instance cache.

```
hadoop jar silkmr.jar match inputDir ouputDir [linkSpec]
```

The following parameters are accepted:

- `inputDir` The path to the previously loaded instance cache.
- `ouputDir` The directory, where the generated links will be written to.
- `linkSpec` (optional) The link specification for which links should be generated. Can be omitted if the provided configuration only contains one link specification.

Example:

```
hadoop jar silkmr.jar match ./cache ./links
```

Usage example

We employed Silk to find `owl:sameAs` links between cities in [DBpedia](#) and in [LinkedGeoData](#). The used link specification can be found [online](#). As both datasets are very large, we used a reduced dataset consisting of

10,5000 settlements from DBpedia and 59,000 cities and towns from LinkedGeoData (omitting villages).

We executed the link specification using two different configurations:

1. Silk Single Machine running on a Intel Core2Duo E8500 with 8GB of RAM
2. Silk MapReduce running on Amazon Elastic MapReduce cluster consisting of 10 Amazon EC2 instances (High-CPU Medium Instance Profile)

For each configuration, we executed the link specifications twice:

1. Without the use of the blocking feature. In this case the Silk Linking Engine has to evaluate the full cartesian product resulting in over 6 billion instance comparisons.
2. Using the blocking feature. The link specification has been extended to block the cities by name using 50 blocks

The number of generated links and the time needed to generate the links for each combination is given in the following table:

Variant	Link Generation time	Number of links
	Without Blocking	
Silk Single Machine	54 hours	9,283
Silk MapReduce	6.7 hours	9,283
	With Blocking	
Silk Single Machine	155.5 minutes	9,224
Silk MapReduce	14.4 minutes	9,224

The table clearly shows how Silk MapReduce reduces the execution time significantly by scaling to clusters with multiple machines. The performance has been further improved by employing the blocking feature included into Silk while losing less than 1 % of the links compared to the link specification without blocking.

Silk Server

Introduction

Silk Server is an extension to the *Silk Link Discovery Framework*. *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](#). *Silk Server* matches data describing incoming instances against a local set of known instances and discovers missing links between them based on user-provided link specifications. Incoming instances which do not match a known instance are added to the local set of instances continuously. Using the [Silk Link Specification Language](#) (Silk-LSL) conditions data items must fulfill in order to be interlinked can be specified by combining various similarity metrics and taking the graph around a data item into account. 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. *Silk Server* can be used within Linked Data application architectures as an identity resolution component to add missing RDF links to data that is consumed from the Web of Linked Data.

The main features of the *Silk Server* are:

- It runs as an HTTP server and offers a REST interface that allows applications to check whether data that is discovered on the Web describes an entity that 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 that are checked in order to determine whether an entity is already known to the system.
- It is high-performing by holding the data about all known instances that is required for the comparisons in an in-memory cache, which is also updated as soon as new instances are discovered. In addition, the performance can be further enhanced using a blocking feature.
- It is available under an open source license and can be run on all major platforms.

Configuration

Server configuration parameters

Parameter	Description
configDir	The directory where the Silk Link Specifications can be found. On startup, the Silk Server will load all Link Specifications in this directory. For details on writing a Link Specification for the Silk Server see the next Section.
directory	Specifies whether unknown instances should be added to the instance cache.
writeUnknownEntities	Specifies whether unknown entities should be added to the instance cache.
returnUnknownEntities	Specifies whether the server entities should contain unknown instances, too.

Writing Link Specifications for the Server

For general information on how to write a Linking Specification refer to [Silk Link Specification Language Specification](#). In addition, there are a few points which should be considered when writing a Link Specification for the Server:

DataSources

In a typical use case there is some initial dataset, which shall be loaded by the server on start-up. This can be accomplished by specifying a source dataset in the Link Specification. The target dataset will be formed by the incoming stream and thus is ignored by the server.

Example:

...

```

<DataSources>
  <DataSource id="source" type="file">
    <Param name="file" value="./initialData.rdf"/>
    <Param name="format" value="RDF/XML"/>
  </DataSource>
  <DataSource id="inputStream" type="rdf">
    <Param name="format" value="N-TRIPLE"/>
    <Param name="input" value="" />
  </DataSource>
</DataSources>

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

    <SourceDataset dataSource="source" var="a">
      <RestrictTo>
        ?a rdf:type foaf:Person .
      </RestrictTo>
    </SourceDataset>

    <TargetDataset dataSource="inputStream" var="b">
      <RestrictTo>
        ?b rdf:type foaf:Person .
      </RestrictTo>
    </TargetDataset>

    ...
  </Interlink>
</Interlinks>

```

Output

As the Silk Server returns the generated links in the response, the output section of the Link Specification may be left empty.

Usage

Running the Server

In order to run the Silk Server, you need:

- **Silk Link Discovery Framework** Check out the latest version at: [git@git.assembla.com:silk.git](https://git.assembla.com/silk.git)
- **Java Runtime Environment** The Silk Link Discovery Framework runs on top of the JVM. Get the most recent [JRE](#).
- **Maven** is used for project management and build automation. Get it from: <http://maven.apache.org>.

What to do:

1. Write Silk-LSL configuration files to specify which resources should be interlinked.
2. Modify the Server configuration if needed.
3. Install the Silk Framework in the local Maven Repository: Navigate to the main Silk folder and execute: `mvn install`
4. Run the Silk Server: Navigate to the server folder and execute: `mvn jetty:run`

Note: In order to configure the underlying Jetty Server (e.g. the port on which requests are accepted), you need to edit the pom.xml of the server module. Refer to the [official homepage](#) for details.

Making requests the Server

The Server accepts HTTP Post requests on the URL `http://{ip:port}/api/process?format={format}`.

The input data must be included in the body of the request. By default, the Server expects the input data to be serialized as RDF/XML. Other input formats can be specified using the `format` parameter. Supported input formats are: "RDF/XML", "N-TRIPLE", "TURTLE", "TTL" and "N3"

The Server response contains the generated links as N-Triples. If the `returnUnknownInstances` configuration parameter is set, it will additionally contain a statement of the form `unknownInstance`
<http://www4.wiwiss.fu-berlin.de/bizer/silk/matchingResult> <http://www4.wiwiss.fu-berlin.de/bizer/silk/UnknownInstance> for each unknown instance in the request.

If the `writeUnknownInstances` configuration parameter is set, each unknown instances will be added to the instance cache. In that case, it will be included in the link generation in future requests.

Usage example

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.

The Link Specification used

We have used the following link configuration for linking data items describing the same person:

```
01      <?xml version="1.0" encoding="utf-8" ?>
02      <Silk>
03          <Prefixes>
04              <Prefix id="rdf" namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
05              />
06              <Prefix id="rdfs" namespace="http://www.w3.org/2000/01/rdf-schema#" />
07              <Prefix id="owl" namespace="http://www.w3.org/2002/07/owl#" />
08              <Prefix id="dcterms" namespace="http://purl.org/dc/terms/" />
09              <Prefix id="foaf" namespace="http://xmlns.com/foaf/0.1/" />
10              <Prefix id="vcard" namespace="http://www.w3.org/2006/vcard/ns#" />
11          </Prefixes>
12          <DataSources>
13              <DataSource id="sw_dog_food" type="file">
14                  <Param name="file" value="semantic_web_dog_food.rdf"/>
15                  <Param name="format" value="RDF/XML"/>
16              </DataSource>
17              <DataSource id="input_stream" type="rdf">
18                  <Param name="format" value="N-TRIPLE"/>
19                  <Param name="input" value="" />
20              </DataSource>
21          </DataSources>
22          <Interlinks>
23              <Interlink id="persons">
24                  <LinkType>owl:sameAs</LinkType>
25                  <SourceDataset dataSource="input_stream" var="a">
26                      <RestrictTo>
27                          ?a rdf:type foaf:Person .
28                      </RestrictTo>
29                  </SourceDataset>
30                  <TargetDataset dataSource="sw_dog_food" var="b">
31                      <RestrictTo>
```

```

31         ?b rdf:type foaf:Person .
32     </RestrictTo>
33 </TargetDataset>
34 <LinkageRule>
35     <Aggregate type="average">
36         <Aggregate type="max" required="true">
37             <Compare metric="jaroWinkler">
38                 <TransformInput function="lowerCase">
39                     <Input path="?a/foaf:name"/>
40                 </TransformInput>
41                 <TransformInput function="lowerCase">
42                     <Input path="?b/foaf:name"/>
43                 </TransformInput>
44             </Compare>
45         </Aggregate>
46         <Aggregate type="max" weight="2" required="true">
47             <Compare metric="jaroWinkler">
48                 <TransformInput function="lowerCase">
49                     <Input path="?a/foaf:homepage"/>
50                 </TransformInput>
51                 <TransformInput function="lowerCase">
52                     <Input path="?b/foaf:homepage"/>
53                 </TransformInput>
54             </Compare>
55             <Compare metric="jaroWinkler">
56                 <Input path="?a/foaf:mbox_shasum"/>
57                 <Input path="?b/foaf:mbox_shasum"/>
58             </Compare>
59         </Aggregate>
60     </Aggregate>
61 </LinkageRule>
62 <Filter threshold="0.9"/>
63 </Interlink>
64 </Interlinks>
65 </Silk>

```

The complete link configuration for discovering RDF links between persons as well as publications is available [online](#).

Linkage Rules

The linkage rule 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.

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

In the given experiment's linkage rules we compute similarity values for the FOAF names, homepages, and mailbox hash sums (lines 34 to 61). 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 46).

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 62). 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.

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 output file.

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.

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 (`http://data.semanticweb.org/person/tom-heath`) 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 (`http://data.semanticweb.org/person/ralph-hodgson`) not only uses the account `http://twitter.com/ralphthq` 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 (`http://www.deri.ie/publications/`).

Silk Workbench

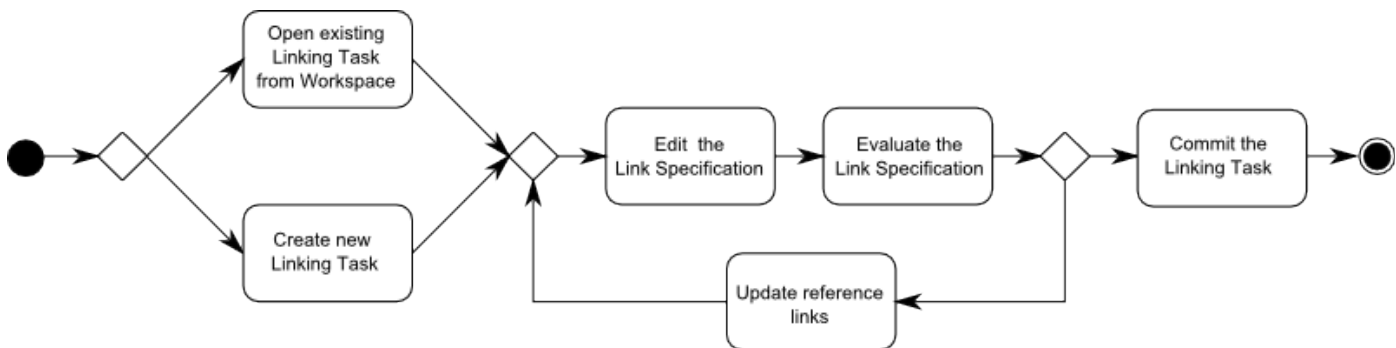
Introduction

Silk Workbench is a web application which guides the user through the process of creating a link specification for interlinking two data sources.

The Silk Workbench provides the following components:

- **Workspace Browser** Enables the user to browse the projects in the workspace. Linking Tasks can be loaded from a project and committed back to it later.
- **Linkage Rule Editor** A graphical editor which enables the user to easily create and edit link specifications. The widget will show the current link specification in a tree view while allowing editing using drag-and-drop.
- **Evaluation** Allows the user to execute the current Link Specification. The links are displayed while they are generated on-the-fly. Generated links for which the reference link set does not specify their correctness, the user may confirm or decline their correctness. The user may request detailed summaries on how the similarity score of specific links is composed of.

The typical workflow of creating a new Link Specification consists of:



1. The user opens an existing Linking Task from the [Workspace](#) or creates a new Linking Task.
2. The user uses the [Linkage Rule Editor](#) to refine the current Linkage Rule.
3. The output of the Link Specification is evaluated based on the reference links using the [Evaluation](#).
4. If all links are correct, the user commits the Link Specification to the [Workspace](#). If some links are wrong the user proceeds with the next step.
5. The user confirms or declines the correctness of a number of links

Installation and Usage

There are two ways to run the Silk Workbench. It can either be run on the command-line or deployed in a servlet container such as Jetty.

Running the Silk Workbench from the Command Line

In order to run the Silk Workbench from the command line, you need:

1. **Silk Link Discovery Framework:** Get the [most recent version](#).
2. **Java Runtime Environment:** The Silk Link Discovery Framework runs on top of the JVM. Get the most recent [JRE](#).

What to do:

1. Run the Silk Workbench:

```
java -jar workbench.war
```

2. Navigate to <http://localhost:8080>

Deploying the Silk Workbench on a Servlet Container

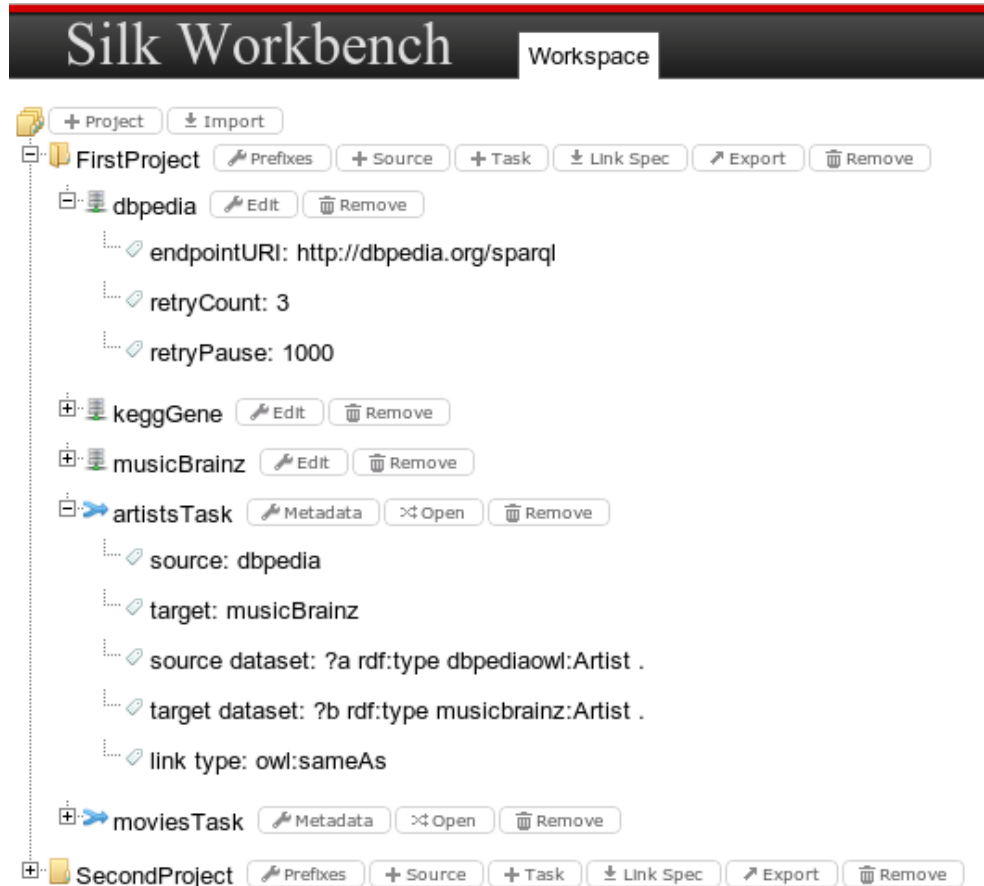
The Silk web archive included in the release can easily be deployed on a servlet container. In order to deploy the Silk Workbench on [Jetty 6](#) you need to:

1. Copy `workbench.war` to the `webapps` directory of your Jetty installation
2. As the default memory settings of Jetty might not be sufficient, increase the maximum heap space.
3. Navigate to the Silk Workbench. By default it is deployed in <http://localhost:8080/silk/>

Workspace

The Workspace allows users to manage data sources and linking tasks defined for each project.

The Workspace Browser shows a tree view of all Projects in the current Workspace:



Projects

A project holds the following information:

1. All URI prefixes which are used in the project.
2. A list of data sources
3. A list of linking tasks

Users are able to create new projects or import existing ones. Existing projects can be deleted or exported to a single file.

Data Sources

A data source holds all information that is needed by Silk to retrieve entities from it.

Users can add new data sources and edit their properties:



The following properties can be edited:

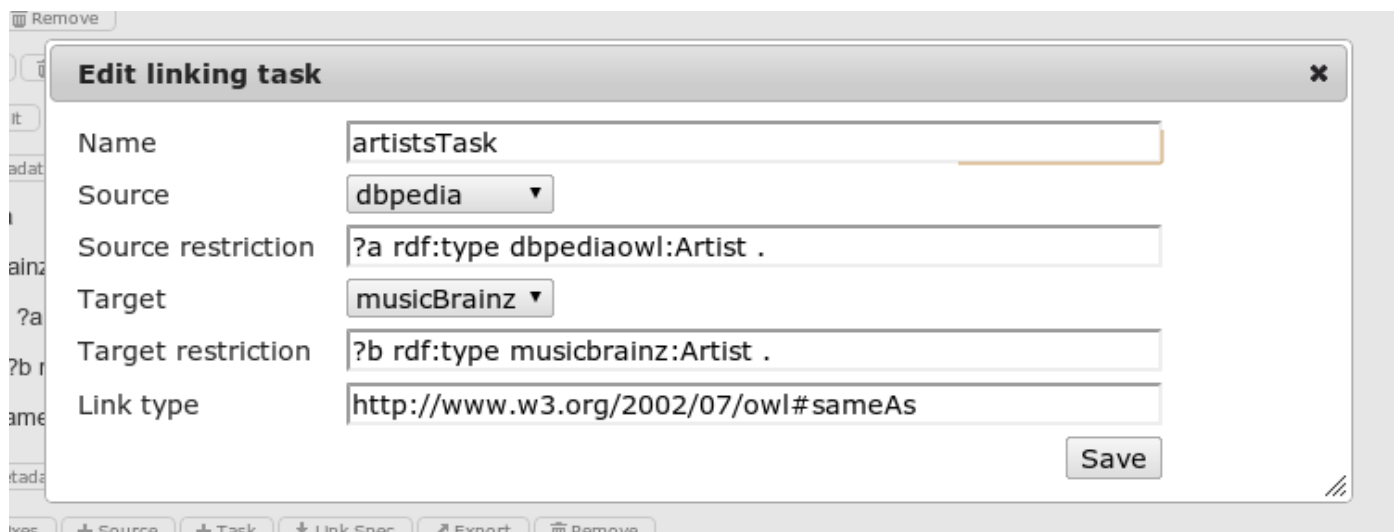
- **Endpoint URI** The URI of the SPARQL endpoint
- **Graph URI** Only retrieve instances from a specific graph
- **Retry count** To recover from intermittent SPARQL endpoint connection failures, the 'retryCount' parameter specifies the number of times to retry connecting.
- **Retry pause** To recover from intermittent SPARQL endpoint connection failures, the 'retryPause' parameter specifies how long to wait between retries.

Linking Tasks

A linking task consists of the following elements:

1. Metadata
2. A link specification
3. Positive and negative reference links

Linking Tasks can be added to an existing project and removed from it. Clicking on **Metadata** opens a dialog to edit the meta data of a linking task:



The following properties can be edited:

- **Name** The unique name of the linking task
- **Source** The source data set
- **Source restriction** Restricts source dataset using SPARQL clauses
- **Target** The target data set
- **Target restriction** Restricts target dataset using SPARQL clauses
- **Link type** Type of the generated link e.g. owl:sameAs

Clicking on the **open** button opens the [Link Specification Editor](#)

Linkage Rule Editor

The Linkage Rule Editor allows users to edit linkage rules in graphical way. Linkage rules are created as a tree, resembling the Silk LSL, by dragging and dropping the rule elements.

The editor is divided in two parts:

The left pane contains the most frequent used [property paths](#) for the given data sets and restrictions. It also contains a list of all Silk operators ([transformations](#), [comparators](#) and [aggregators](#)) as draggable elements.

The right part (editor pane) allows for drawing the flow chart by combining the elements chosen.

Editing

- Drag elements from the left pane to the editor pane.
- Connect the elements by drawing connections from and to the element endpoints (dots to the left and right of the element box).
- Build a flow chart by connecting the elements, ending in one single element (either a comparison or aggregation).

The editor will guide the user in building the flow chart by highlighting connectable elements when drawing a new connection line.

Property Paths

Property paths for both data sources to be interlinked are loaded on the left pane and added in the order of their frequency in the data source.

Users can also add custom paths by dragging the `{(custom path)}` element to the editor pane and [editing the path](#).

Operators

The following operator panes are shown below the property paths:

- [Transformations](#)
- [Comparators](#)
- [Aggregators](#)

Hovering over the operator elements will show you more information on them.

Threshold

`Threshold` defines the minimum similarity of two data items which is required to generate a link between them. Please provide values between 0 and 1.

Link Limit

`Link Limit` defines the number of links originating from a single data item. Please choose between 1 and n (unlimited).

Generating Links

As finding a good linking heuristics is usually an iterative process, the Silk Workbench allows the user to quickly evaluate the links which are generated by the current link specification.

Silk Workbench Workspace: Example Editor: drugs Generate Links Reference Links About

Start Cancel Matching: 58 tasks 2350522 links. (13.7%) Help

Expand All Collapse All Prev 1 2 3 4 5 6 7 8 9 10 11 Next Filter:

Source	Target	Confidence	Correct?
http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/9904	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs...	30.0%	✓ ? ✕
<div>Aggregation(max) 30.0%</div> <div>Comparison(levenshteinDistance) 30.0%</div> <div>Input ?a/rdfs:label nandrolone</div> <div>Input ?b/rdfs:label timolol</div> <div>Comparison(levenshteinDistance) -100.0%</div> <div>Input ?a/rdfs:label nandrolone</div> <div>Input ?b/drugbank:synonym timolol maleate timololum [inn-latin]</div>			
http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/9904	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs...	10.0%	✓ ? ✕
http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/9904	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs...	20.0%	✓ ? ✕

After clicking on the **Start** button, the linking engine starts to generate links in the background. The view is updated whenever new links have been found to show the all generated links. For further examination, a drill-down view can be shown by clicking on a link. The drill-down shows a detailed summary how the individual comparisons and aggregations contribute to the overall fitness of the link, the overall similarity between two links is composed by clicking on a link. This information allows the user to spot parts of the similarity evaluation which did not behave as expected.

Based on its correctness, each link can be associated to one of the following 3 categories:

- ✓ Confirms the link as correct. Confirmed links are part of the positive reference link set.
- ? Contains link whose correctness is not decided i.e. which are not contained in the reference link sets.
- ✕ Confirms the link as incorrect. Incorrect links are part of the negative reference link set.

In order to evaluate the correctness of a link specification, the user typically wants to evaluate the correctness of links which are close the similarity threshold. Clicking on the confidence header sorts all links by their confidence which allows to find these links. In addition, a filter can be added, so that only links are shown which contain a specific string.

Managing Reference Links

In order to iteratively increase the quality of a link specification, the workbench holds a set of reference links for which their correctness has been confirmed or declined by the user.

Silk Workbench

Workspace: ExampleEditor: drugsGenerate LinksReference LinksAbout

PositiveNegativeImport Reference LinksHelp

Expand AllCollapse All

Prev1Next

Filter:

Source	Target	Confidence	Status	Correct?
<div><div>▼ http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/4052</div><div><div>Aggregation(max)100.0%</div><div><div>Comparison(levenshteinDistance)100.0%</div><div><div>Input?a/rdfs:labelmeloxicam</div><div>Input?b/rdfs:labelmeloxicam</div></div><div><div>Comparison(levenshteinDistance)0.0%</div><div><div>Input?a/rdfs:labelmeloxicam</div><div>Input?b/drugbank:synonymmeloxicamum [latin]</div></div></div></div></div></div>	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00...	100.0%	found	<div></div>
► http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/51577	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00...	100.0%	found	<div></div>
► http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/16231	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00...	100.0%	found	<div></div>

Reference Links may be imported and exported in the Ontology Alignment Format specified at <http://alignapi.gforge.inria.fr/format.html>.

Learning Linkage Rules

The Silk Workbench supports learning linkage rules. In order to be able to learn linkage rules, the current linking task must contain reference links. Usually reference links can be added in two ways:

- By loading existing reference links from a file. See [Managing Reference Links](#) for details.
- By using the sampling tab

After reference links have been added, the learning can be started by pressing the start button. While learning, the current population of candidate linkage rules is displayed. At any time, the learning can be stopped by pressing the stop button. The learning will stop automatically as soon as either the full f-Measure is reached or the maximum number of iterations has been exceeded.

As soon as the learning has finished, the user can view the learned linkage rules and select a linkage rule for loading it into the editor.

Silk Workbench

Workspace: MoviesEditor: MoviesGenerate LinksSample LinksReference LinksLearnAbout

StartLearning Task finished in 322ms (100.0%)

Expand AllCollapse All

Prev12345Next

Description	Score	MCC	F-Measure	Actions
2 Comparisons and 0 Transformations	100.0%	100.0%	100.0%	
Aggregation: min <ul style="list-style-type: none">Comparison: jaccard (0.3909611306553715)<ul style="list-style-type: none">Input: ?a/<http://silktest.org/date>Input: ?b/<http://silktest.org/date>Comparison: jaccard (0.6340771811543341)<ul style="list-style-type: none">Input: ?a/<http://www.w3.org/2000/01/rdf-schema#label>Input: ?b/<http://www.w3.org/2000/01/rdf-schema#label>				
2 Comparisons and 0 Transformations	100.0%	100.0%	100.0%	
2 Comparisons and 2 Transformations	100.0%	100.0%	100.0%	
2 Comparisons and 1 Transformations	100.0%	100.0%	100.0%	
2 Comparisons and 2 Transformations	100.0%	100.0%	100.0%	
2 Comparisons and 4 Transformations	100.0%	100.0%	100.0%	
2 Comparisons and 0 Transformations	68.3%	68.3%	75.0%	
2 Comparisons and 1 Transformations	68.3%	68.3%	75.0%	

REST API

Currently, Silk Workbench does not offer a public REST API. This page is intended to specify the initial API

General

The REST API is available under the path `{deploymentURI}/api/`. All paths given on this page are relative to this.

Data Sources

Data Sources are defined using the [Silk Link Specification Language](#).

Resource	Description
GET {projectID}/source	Retrieves all data sources.
GET {projectID}/source/{taskID}	Retrieves a data source.
PUT {projectID}/source/{taskID}	Creates or updates a data source.
DELETE {projectID}/source/{taskID}	Deletes a data source.

Example

Request:

```
GET my_project/source/dbpedia
```

Response:

```
<DataSource id="dbpedia" type="sparqlEndpoint">
  <Param name="endpointURI" value="http://dbpedia.org/sparql" />
  <Param name="retryCount" value="10" />
  <Param name="retryPause" value="1000" />
</DataSource>
```

Link Specifications

Link Specifications are defined using the [Silk Link Specification Language](#).

Resource	Description
GET {projectID}/linkSpec	Retrieves all link specifications.
GET {projectID}/linkSpec/{taskID}	Retrieves a link specification.
PUT {projectID}/linkSpec/{taskID}	Creates or updates a link specification.
DELETE {projectID}/linkSpec/{taskID}	Deletes a link specification.

Reference Links

Reference Links are defined using the Ontology Alignment Format specified at <http://alignapi.gforge.inria.fr/format.html>.

Resource	Description
GET {projectID}/referenceLinks/{taskID}	Retrieves a reference links set.
PUT {projectID}/referenceLinks/{taskID}	Creates or updates a reference links set.
DELETE {projectID}/referenceLinks/{taskID}	Deletes a reference links set.

Example

Request:

```
GET my_project/referenceLinks/movies
```

Response:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#">
  <Alignment>
    <map>
      <Cell>
        <entity1 rdf:resource="http://dbpedia.org/resource/What%27s_Up,_Doc%3F_%281972_film%29"></entity1>
        <entity2 rdf:resource="http://data.linkedmdb.org/resource/film/1982"></entity2>
        <relation>=</relation>
        <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.1</measure>
      </Cell>
    </map>
    ...
  </Alignment>
</rdf:RDF>
```

Basic Concepts

This section introduces the basic concepts in link discovery such as data sources, linkage rules and reference alignments.

Data Sources

Overview

Data sources hold the access parameters to local or remote SPARQL endpoints or RDF files. The defined data sources may later be referred to and used by their ID. Data Sources can be defined using either the API or [XML](#).

Available Data Sources

SPARQL Endpoint Data Source Definitions

For SPARQL endpoints (dataSource type: `sparqlEndpoint`) the following parameters exist:

Parameter	Description	Default
endpointURI	The URI of the SPARQL endpoint.	
login	Login required for authentication	No login
password	Password required for authentication	No password
instanceList	A list of instances to be retrieved. If not given, all instances will be retrieved. Multiple instances can be separated by a space.	Retrieve all instances
pageSize	Limits each SPARQL query to a fixed amount of results. Silk implements a paging mechanism which translates the pageSize parameter into SPARQL LIMIT and OFFSET clauses.	1000
graph	Only retrieve instances from a specific graph.	
pauseTime	To allow rate-limiting of queries to public SPARQL servers, the pauseTime statement specifies the number of milliseconds to wait in between subsequent queries.	0
retryCount	To recover from intermittent SPARQL endpoint connection failures, the retryCount parameter specifies the number of times to retry connecting.	3
retryPause	Specifies how long to wait between retries.	1000

Example (XML)

```
<DataSource id="dbpedia" type="sparqlEndpoint">
  <Param name="endpointURI" value="http://dbpedia.org/sparql" />
  <Param name="retryCount" value="100" />
</DataSource>
```

Example (Scala API)

Note that all parameters except the endpoint URI are optional and can be left out.

```
Source("dbpedia",
  SparqlDataSource(
    endpointURI = "http://dbpedia.org/sparql",
    login= "user",
    password= "password",
    graph= "http://dbpedia.org",
    pageSize = 1000,
    pauseTime = 0,
    retryCount = 3,
    retryPause = 1000
```

```
)  
)
```

RDF File Data Source Definitions

For RDF files (dataSource type: `file`) the following parameters exist:

Parameter	Description	Default
file (mandatory)	The location of the RDF file.	
format (mandatory)	The format of the RDF file. Allowed values: "RDF/XML", "N-TRIPLE", "TURTLE", "TTL", "N3"	

Currently the data set is held in memory.

Example (XML)

```
<DataSource id="musicbrainz" type="file">  
  <Param name="file" value="musicbrainz_dump.nt" />  
  <Param name="format" value="N-TRIPLE" />  
</DataSource>
```

Example (Scala API)

```
Source("musicbrainz",  
  FileDataSource(  
    file = "musicbrainz_dump.nt",  
    format = "N-TRIPLE"  
  )  
)
```

Linkage Rule

A linkage rule specifies how two data items are compared for similarity. A linkage rule consists of four basic components:

- **Path Input** Retrieves values from an entity by a given RDF path e.g. `?movie/dbpedia:director/rdfs:label`
- **Transformation** Applies a data transformation to all values e.g. `lowerCase`
- **Comparison** Evaluates the similarity of two inputs based on a user-defined distance measure and returns a confidence.
- **Aggregation** Aggregates multiple confidence values.

Path Input

Overview

An input retrieves all values which are connected to the entities by a specific path.

Every path statement begins with a variable (as defined in the datasets), which may be followed by a series of path elements. If a path cannot be resolved due to a missing property or a too restrictive filter, an empty result set is returned.

The following operators can be used to traverse the graph:

Operator	Name	Use	Description
/	forward operator	<code><path_segment>/<property></code>	Moves forward from a subject resource (set) through a property to its object resource (set).
\	reverse operator	<code><path_segment>\<property></code>	Moves backward from an object resource (set) through a property to its subject resource (set).
[]	filter operator	<code><path_segment>[<property> <comp_operator> <value>] <path_segment>[@lang <comp_operator> <value>]</code>	Reduces the currently selected set of resources to the ones matching the filter expression. comp_operator may be one of >, <, >=, <=, =, !=

Examples

XML

```
# Select the English label of a movie
<Input path="?movie/rdfs:label[@lang = 'en']" />

# Select the label (set) of the director(s) of a movie
<Input path="?movie/dbpedia:director/rdfs:label" />

# Select the albums of a given artist (albums have an dbpedia:artist property)
<Input path="?artist\dbpedia:artist[rdf:type = dbpedia:Album]" />
```

Scala API

```
# Select the English label of a movie
Path.parse("?movie/rdfs:label[@lang = 'en']")

# Select the label (set) of the director(s) of a movie
Path.parse("?movie/dbpedia:director/rdfs:label")

# Select the albums of a given artist (albums have an dbpedia:artist property)
Path.parse("?artist\dbpedia:artist[rdf:type = dbpedia:Album]")
```

Transformation

Overview

As different datasets usually use different data formats, a transformation can be used to normalize the values prior to comparison.

Parameters

TODO

Examples

XML

```
<TransformInput function="lowerCase">
  <TransformInput function="replace">
    <Input path="?a/rdfs:label" />
    <Param name="search" value="_" />
    <Param name="replace" value=" " />
  </TransformInput>
</TransformInput>
```

Scala API

```
TransformInput(
  id = "ReplaceUnderscores",
  transformer = ReplaceTransformer("_", " ")
  inputs = PathInput(path = Path.parse("?a/rdfs:label"))
)
```

Transformations

Silk provides the following transformation and normalization functions:

Function and parameters	Description
removeBlanks	Remove whitespace from a string.
removeSpecialChars	Remove special characters (including punctuation) from a string.
lowerCase	Convert a string to lower case.
upperCase	Convert a string to upper case.
capitalize(allWords)	Capitalizes the string i.e. converts the first character to upper case. If 'allWords' is set to true, all words are capitalized and not only the first character. By default 'allWords' is set to false.
stem	Apply word stemming to the string.
alphaReduce	Strip all non-alphabetic characters from a string.
numReduce	Strip all non-numeric characters from a string.
replace(string search, string replace)	Replace all occurrences of "search" with "replace" in a string.
regexReplace(string regex, string replace)	Replace all occurrences of a regex "regex" with "replace" in a string.

stripPrefix	Strip the prefix from a string.
stripPostfix	Strip the postfix from a string.
stripUriPrefix	Strip the URI prefix (e.g. http://dbpedia.org/resource/) from a string.
concat	Concatenates strings from two inputs.
logarithm([base])	Transforms all numbers by applying the logarithm function. Non-numeric values are left unchanged. If base is not defined, it defaults to 10.
convert(string sourceCharset, string targetCharset)	Converts the string from "sourceCharset" to "targetCharset"
tokenize([regex])	Splits the string into tokens. Splits at all matches of "regex" if provided and at whitespaces otherwise.
removeValues(blacklist)	Removes specific values (i.e. stop words) from the value set. 'blacklist' is a comma-separated list of words.

Comparison

Overview

A comparison operator evaluates two inputs and computes the **similarity** based on a user-defined **distance measure** and a user-defined **threshold**.

The **distance measure** always outputs 0 for a perfect match, and a higher value for an imperfect match. Only distance values between 0 and *threshold* will result in a positive similarity score. Therefore it is important to know how the distance measures work and what the range of their output values is in order to set a threshold value sensibly.

Parameters

Parameter	Description
required (optional)	If required is true, the parent aggregation only yields a confidence value if the given inputs have values for both instances.
weight (optional)	Weight of this comparison. The weight is used by some aggregations such as the weighted average aggregation.
threshold	The maximum distance. For normalized distance measures, the threshold should be between 0.0 and 1.0.
distanceMeasure	The used distance measure. For a list of available distance measures see below.
Inputs	The 2 inputs for the comparison.

Examples

XML

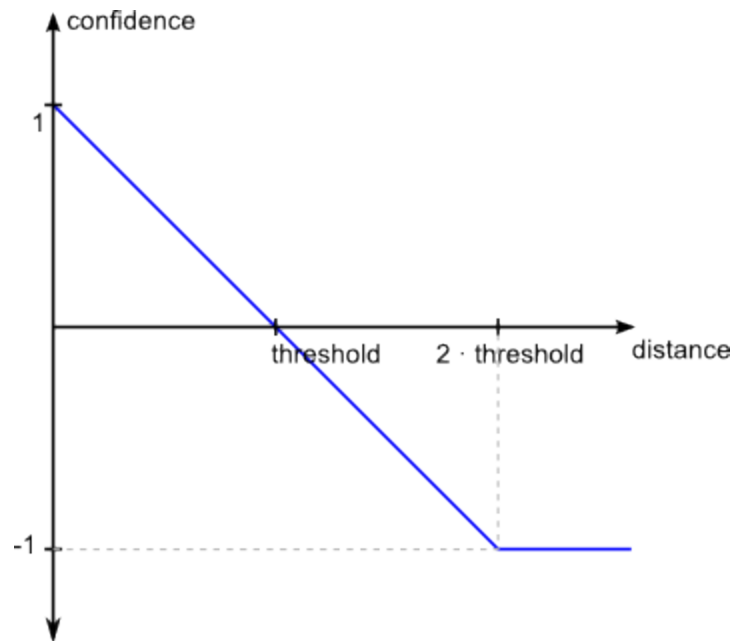
```
<Compare metric="levenshteinDistance" threshold="2.0" required="true">
  <TransformInput function="lowerCase">
    <Input path="?a/rdfs:label"/>
  </TransformInput>
  <TransformInput function="lowerCase">
    <Input path="?b/rdfs:label"/>
  </TransformInput>
</Compare>
```

Scala API

```
Comparison(
  id = "labels",
  required = false,
  weight = 1,
  threshold = 2.0,
  metric = LevenshteinDistance()
  inputs = PathInput(path = Path.parse("?a/rdfs:label")) ::
    PathInput(path = Path.parse("?b/rdfs:label")) :: Nil
)
```

Threshold

The threshold is used to convert the computed distance to a confidence between -1.0 and 1.0. Links will be generated for confidences above 0 while higher confidence values imply a higher similarity between the compared entities.



Distance Measures

Character-Based Distance Measures

Character-based distance measures compare strings on the character level. They are well suited for handling typographical errors.

Measure	Description	Normalized
levenshteinDistance	Levenshtein distance. The minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character	No
levenshtein	The levensthein distance normalized to the interval [0,1]	Yes
jaro	Jaro distance metric. Simple distance metric originally developed to compare person names.	Yes
jaroWinkler	Jaro-Winkler distance measure. The Jaro–Winkler distance metric is designed and best suited for short strings such as person names	Yes
equality	0 if strings are equal, 1 otherwise.	Yes
inequality	1 if strings are equal, 0 otherwise.	Yes

Example:

```
<Compare metric="levenshteinDistance" threshold="2">
  <Input path="?a/rdfs:label" />
  <Input path="?b/gn:name" />
</Compare>
```

Token-Based Distance Measures

While character-based distance measure work well for typographical errors, they are are number of tasks where token-base distance measures are better suited:

- Strings where parts are reordered e.g. “John Doe” and “Doe, John”
- Texts consisting of multiple words

Measure	Description	Normalized
jaccard	Jaccard distance coefficient.	Yes
dice	Dice distance coefficient.	Yes
softjaccard	Soft Jaccard similarity coefficient. Same as Jaccard distance but values within an levenhstein distance of 'maxDistance' are considered equivalent.	Yes

Example:

```
<Compare metric="jaccard" threshold="0.2">
  <TransformInput function="tokenize">
    <Input path="?a/rdfs:label" />
  </TransformInput>
  <TransformInput function="tokenize">
    <Input path="?b/gn:name" />
  </TransformInput>
</Compare>
```

Special Purpose Distance Measures

Silk offers a number of distance measures which are designed to compare specific types of data e.g. numeric values.

Measure	Description	Normalized
num(float minValue, float maxValue)	Computes the numeric difference between two numbers Parameters: <code>minValue</code> , <code>maxValue</code> The minimum and maximum values which occur in the datasource	No
date	Computes the distance between two dates ("YYYY-MM-DD" format). Returns the difference in days	No
dateTime	Computes the distance between two date time values (xsd:dateTime format). Returns the difference in seconds	No
wgs84(string unit, string curveStyle)	Computes the geographical distance between two points. Parameters: <code>unit</code> The unit in which the distance is measured. Allowed values: "meter" or "m" (default) , "kilometer" or "km" Author: Konrad Höffner (MOLE subgroup of Research Group AKSW, University of Leipzig)	No

Example:

```
<Compare metric="wgs84" threshold="50">
  <Input path="?a/wgs84:geometry" />
  <Input path="?b/wgs84:geometry" />
  <Param name="unit" value="km" />
</Compare>
```

Aggregation

Overview

An aggregation combines multiple confidence values into a single value. In order to determine if two entities are duplicates it is usually not sufficient to compare a single property. For instance, when comparing geographic entities, an aggregation may aggregate the similarities between the names of the entities and the similarities based on the distance between the entities.

Parameters

Required (Optional)

The required attribute can be set if the aggregation only should generate a result if a specific suboperator return a value

Weights (Optional)

Some comparison operators might be more relevant for the correct establishment of a link between two resources than others. For example, depending on data formats/quality, matching labels might be considered less important than matching geocoordinates when linking cities. If this modifier is not supplied, a default weight of 1 will be assumed. The weight is only considered in the aggregation types average, quadraticMean and geometricMean.

Type

The function according to the similarity values are aggregated. The following functions are included in Silk:

Id	Name	Description
average	AverageAggregator	Evaluate to the (weighted) average of confidence values.
max	MaximumAggregator	Evaluate to the highest confidence in the group.
min	MinimumAggregator	Evaluate to the lowest confidence in the group.
quadraticMean	QuadraticMeanAggregator	Apply Euclidian distance aggregation.
geometricMean	GeometricMeanAggregator	Compute the (weighted) geometric mean of a group of confidence values.

Examples

XML

```
<Aggregate type="average">
  <Compare metric="jaro" required="true">
    <Input path="?a/rdfs:label" />
    <Input path="?b/gn:name" />
  </Compare>
  <Compare metric="num">
    <Input path="?a/dbpedia:populationTotal" />
    <Input path="?b/gn:population" />
  </Compare>
</Aggregate>
```

Scala API

```
Aggregation(
  id = "id1",
  required = false,
  weight = 1,
  operators = operators,
  aggregator = MaximumAggregator()
```

Output

Overview

An output represents a destination where the generated links are written to. Outputs can have an acceptance windows (defined by `minConfidence` and `maxConfidence`) e.g. for separating accepted links and links with a lower confidence which need to be verified before being accepted.

Examples

XML

```
<Outputs>
  <Output type="output type" minConfidence="lower threshold" maxConfidence="upper threshold">
    <Param name="parameter name" value="parameter value" />
    ...
  </Output>
</Outputs>
```

API

```
Output(
  id = "MyOutput",
  writer = FileWriter(file = "output.nt", format = "ntriples")
)
```

Available Output Types

File Output

Parameter	Description
file	Writes the links to a file. Links are written to {user.dir}/.silk/output/ by default (i.e. if a relative path is provided).
format	The output format. Available formats are "ntriples" (N-Triples format) and "alignment" (Alignment format)

Example:

```
<Outputs>
  <Output type="file" minConfidence="0.1">
    <Param name="file" value="accept_links.nt"/>
    <Param name="format" value="ntriples"/>
  </Output>
</Outputs>
```

Formats

N-Triples

Writes the links as N-Triples statements.

Alignment

Writes the links in the [OAEI Alignment Format](#). This includes not only the uris of the source and target entities, but also the confidence of each link.

SPARQL/Update Output

Parameter	Description	Default
uri	The URI of the SPARQL/Update endpoint e.g. http://localhost:8090/virtuoso/sparql	
login	Login required for authentication	No login
password	Password required for authentication	No password
parameter	The HTTP parameter used to submit queries. Defaults to “query” which works for most endpoints. Some endpoints require different parameters e.g. Sesame expects “update” and Joseki expects “request”.	query
graphUri	The URI of the graph to put the links	no graph

Example:

```
<Outputs>
  <Output type="sparul" >
    <Param name="uri" value="http://localhost:8080/query"/>
  </Output>
</Outputs>
```

Detailed Alignment (Work in Progress)

Writes the links in a detailed alignment format.

Example:

```
<DetailedAlignment>
  <Cell>
    <Entity1 rdf:resource="http://dbpedia.org/resource/Hydroflumethiazide"/>
    <Entity2 rdf:resource="http://www4.wiwiiss.fu-berlin.de/drugbank/resource/drugs/D
B00774"/>
    <Aggregate similarity="1.0">
      <Compare similarity="1.0">
        <Input path="?a/rdfs:label">
          <Value>Idroflumetiazide</Value>
          <Value>Hydroflumethiazide</Value>
        </Input>
        <Input path="?b/rdfs:label"/>
          <Value>Hydroflumethiazide</Value>
        </Input>
      </Compare>
      <Compare similarity="1.0">
        <Input path="?a/rdfs:label">
          <Value>Idroflumetiazide</Value>
          <Value>Hydroflumethiazide</Value>
        </Input>
        <Input path="?b/drugbank:synonym"/>
          <Value>Idroflumetiazide</Value>
        </Input>
      </Compare>
    </Aggregate>
  </Cell>
</DetailedAlignment>
```

```
</Cell>
...
<DetailedAlignment>
```

Alternative (as RDF/XML):

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<DetailedAlignment>
  <map>
    <Cell>
      <entity1 rdf:resource="http://dbpedia.org/resource/Hydroflumethiazide"/>
      <entity2 rdf:resource="http://www4.wiwi.fu-berlin.de/drugbank/resource/drugs/DB00774"/>
      <aggregate>
        <similarity>1.0</similarity>
        <compare>
          <similarity>1.0</similarity>
          <input>
            <path>?a/rdfs:label</path>
            <value>Idroflumetiazide</value>
            <value>Hydroflumethiazide</value>
          </input>
          <input>
            <path>?b/rdfs:label</path>
            <value>Hydroflumethiazide</value>
          </input>
        </compare>
        <compare>
          <similarity>1.0</similarity>
          <input>
            <path>?a/rdfs:label</path>
            <value>Idroflumetiazide</value>
            <value>Hydroflumethiazide</value>
          </input>
          <input>
            <path>?b/drugbank:synonym</path>
            <value>Idroflumetiazide</value>
          </input>
        </compare>
      </aggregate>
    </Cell>
  </map>
  ...
</DetailedAlignment>
</rdf:RDF>
```

Reference Links

Overview

Reference Links (in record linkage usually called *Golden Set*) are a set of links whose correctness has either been confirmed or declined by the user. Reference links can be used to evaluate the completeness and correctness of a linkage rule.

We distinguish between positive and negative reference links:

- *Positive reference links* represent definitive matches
- *Negative reference links* represent definitive non-matches.

Link Specification Language

The Silk framework provides a declarative language for specifying 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. This section describes the language constructs of the Silk Link Specification Language (Silk-LSL).

The example below gives an overview of the main language constructs of Silk-LSL.

```
<Silk>

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

  <DataSources>
    <DataSource id="dbpedia">
      <Param name="endpointURI" value="http://demo_sparql_server1/sparql" />
      <Param name="graph" value="http://dbpedia.org" />
    </DataSource>

    <DataSource id="geonames">
      <Param name="endpointURI" value="http://demo_sparql_server2/sparql" />
      <Param name="graph" value="http://sws.geonames.org/" />
    </DataSource>
  </DataSources>

  [<Blocking blocks="100" />]

  <Interlinks>
    <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>

      <LinkageRule>
        <Aggregate type="average">
          <Compare metric="jaro">
            <Input path="?a/rdfs:label" />
            <Input path="?b/gn:name" />
          </Compare>
          <Compare metric="num">
            <Input path="?a/dbpedia:populationTotal" />
            <Input path="?b/gn:population" />
          </Compare>
        </Aggregate>
      </LinkageRule>
    </Interlink>
  </Interlinks>
</Silk>
```

```

        </Aggregate>
    </LinkageRule>

    <Filter threshold="0.9" />

    <Outputs>
        <Output type="file" minConfidence="0.95">
            <Param name="file" value="accepted_links.nt" />
            <Param name="format" value="ntriples" />
        </Output>
        <Output type="file" maxConfidence="0.95">
            <Param name="file" value="verify_links.nt" />
            <Param name="format" value="alignment" />
        </Output>
    </Outputs>
</Interlink>
</Interlinks>

</Silk>

```

Structure and Elements

The Silk-LSL is expressed in XML as specified by the corresponding Silk XML Schema. The root tag name is `<Silk>`. A valid document may contain four types of top-level statements beneath the root element:

- prefix definitions
- datasource definitions
- link specifications
- output definitions

```

<?xml version="1.0" encoding="utf-8" ?>
<Silk>
    <Prefixes ... />
    ...
    <DataSources ... />
    ...
    [<Blocking ... />]
    ...
    <Interlinks ... />
    ...
    [<Outputs ... />]
    ...
</Silk>

```

The Blocking and Outputs statements are optional.

Prefix Definitions

Prefix definitions are top-level statements that allow the binding of a prefix to a namespace:

```

<Prefixes>
    <Prefix id="prefix id" namespace="namespace URI" />
</Prefixes>

```

Example:


```
<Prefixes>
  <Prefix id="rdf" namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
</Prefixes>
```

Data Source Definitions

Data source definitions are top-level statements that allow the specification of access parameters to local or remote SPARQL endpoints. The defined data sources may later be referred to and used by their ID within link specification statements.

```
<DataSources>
  <DataSource id="data source ID" type="dataSource type">
    <Param name="parameter name" value="parameter value" />
    ...
  </DataSource>
</DataSources>
```

For details see [Data Sources](#)

Blocking Data Items

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) which might be too time consuming, blocking can be used to reduce the number of comparisons. Blocking partitions similar data items into clusters reducing the comparison to items in the same cluster.

For example given two datasets describing books, in order to reduce the number of comparisons, we could block the books by publisher. In this case only books from the same publisher will be compared. Given a number of 40.000 books in the first dataset and 30.000 in the second dataset, evaluating the full Cartesian product requires 1.2 billion comparisons.

If we block this datasets by publisher, each book will be allocated to a block based on its publisher. Using 100 blocks, if the books are uniformly distributed, there will be 400 respectively 300 books per block, which reduces the number of comparisons to 12 million.

The `<Blocking>` statement enables the blocking phase. Additional configuration is not required as Silk will automatically generate a blocking function from the link specification. The optional `blocks` attribute specifies the number of blocks which will be used. The default value of 100 blocks is appropriate for most use cases. If no `<Blocking>` statement is supplied in a link specification, the comparison will loop over all resource pairs (Cartesian product).

Link Specifications

Link specification statements state that a link of a given type should be established between two data items if a specified condition is satisfied. This condition may contain different similarity metrics, aggregation and transformation functions, thresholds and weights.

A Silk linking configuration may contain several link specifications if different types of links should be generated. Link specifications are structured as follows:

```
<Interlinks>
  <Interlink id="interlink id">
    <LinkType>link type URI</LinkType>
    <SourceDataset dataSource="dataSource id" var="resource variable name">
      [<RestrictTo>SPARQL restriction</RestrictTo>]
    </SourceDataset>
    <TargetDataset dataSource="dataSource id" var="resource variable name">
      [<RestrictTo>SPARQL restriction</RestrictTo>]
    </TargetDataset>
  </Interlink>
</Interlinks>
```

```

</TargetDataset>
<LinkageRule>
  <Aggregate type="average|max|min|...">
    <Compare metric="similarity metric">
      <Input path="RDF path" />
      <TransformInput function="transform function name">
        <Input path="RDF path" />
      </TransformInput>
      <Param name="parameter name" value="literal value" />
    </Compare>
    <Compare ...>
    </Compare>
    ...
  </Aggregate>
</LinkageRule>

<Filter threshold="threshold" limit="link limit" />

<Outputs>
  <Output type="output type" minConfidence="lower threshold" maxConfidence="upper threshold">
    <Param name="parameter name" value="parameter value" />
    ...
  </Output>
</Outputs>
</Interlink>
<Interlink id="...">
  ...
</Interlink>
...
</Interlinks>

```

Example:

```

<Interlinks>
  <Interlink id="countries">
    <LinkType>owl:sameAs</LinkType>
    <SourceDataset dataSource="dbpedia" var="a">
      <RestrictTo>
        ?a rdf:type dbpedia:Country
      </RestrictTo>
    </SourceDataset>
    <TargetDataset dataSource="factbook" var="b">
    </TargetDataset>
    <LinkageRule>
      <Aggregate type="average">
        <Compare metric="jaro" weight="2">
          <Input path="?a/rdfs:label" />
          <Input path="?b/rdfs:label" />
        </Compare>
        <Compare id="num">
          <Input path="?a/dbpedia:population" />
          <Param path="?b/fb:totalPopulation" />
          <Param name="factor" value="2" />
        </Compare>
      </Aggregate>
    </LinkageRule>
  </Interlink>
</Interlinks>

```

```

</LinkageRule>
<Filter threshold="0.7" limit="1" />
<Outputs>
  <Output type="file" maxConfidence="0.9" >
    <Param name="file" value="verify_links.nt"/>
    <Param name="format" value="ntriples"/>
  </Output>
  <Output type="file" minConfidence="0.9">
    <Param name="file" value="accepted_links.nt"/>
    <Param name="format" value="ntriples"/>
  </Output>
</Outputs>
</Interlink>
</Interlinks>

```

Link Type

The LinkType directive defines the type of the generated links.

Example:

```
<LinkType>owl:sameAs</LinkType>
```

Datasets

The SourceDataset and TargetDataset directives define the set of data items which are to be compared. The entities which are to be interlinked are selected by providing a restriction for each data source. In its simplest form a restriction just selects all entities of a specific type inside the data source. For instance, in order to interlink cities in DBpedia, a valid restriction may select all entities with the type dbpedia:City. For more complex restrictions, arbitrary SPARQL triple patterns are allowed to be specified.

Parameter	Description
dataSource	Selects a previously defined data source by its id.
var	Binds each data item to this variable.
RestrictTo	Restricts this dataset using SPARQL clauses.

```

<SourceDataset dataSource="dataSource id" var="resource variable name">
  [<RestrictTo>SPARQL restriction</RestrictTo>]
</SourceDataset>
<TargetDataset dataSource="dataSource id" var="resource variable name">
  [<RestrictTo>SPARQL restriction</RestrictTo>]
</TargetDataset>

```

Example:

```

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

```

Linkage Rule

The linkage rule specifies how two data items are compared for similarity. Refer to [Linkage Rule](#) for details.

Link Filter

The Link Filter allows for filtering the generated links. It only has a single parameter:

`limit` defines the number of links originating from a single data item. Only the n highest-rated links per source data item will remain after the filtering. If no limit is provided, all links will be returned.

Examples

A limit of 1 link per entity:

```
<Filter limit="1" />
```

No limit:

```
<Filter />
```

API

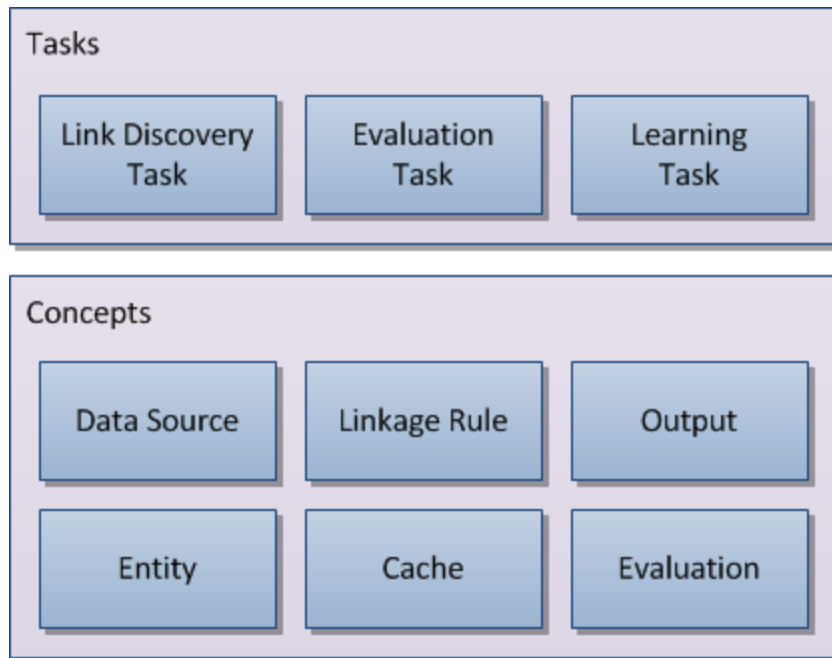
Note: We are currently in the process of documenting the API. This is work-in-progress and still incomplete.

Introduction

The Silk Link Discovery Framework is written in the [Scala](#) programming language, which is a modern language for the Java virtual machine. Therefore you need Java 5 or higher for developing and using the Silk API.

The current Scaladoc can be found [here](#).

TODO: add some general introduction here



Getting started

There are several ways you can use the Silk API in your project:

1. Using the Maven build system
2. By using the `silk.jar` which is included in every release.
3. By downloading the current source code

We recommend using the first option.

Using the Silk API with Maven

1. Download the current source code using git:

```
git clone git://git.assembla.com/silk.git
```

Instructions on installing git on different operating systems can be found [here](#).

2. Navigate to the silk2 folder and install the Silk API using [Maven](#):

```
mvn install
```

3. Now you can use the Silk API in your Maven project by declaring a dependency:

```
<dependency>
```

```
<groupId>de.fuberlin.wiwiss.silk</groupId>  
<artifactId>silk-core</artifactId>  
<version>2.5</version>  
</dependency>
```

Modules

Modules

The Silk Link Discovery Framework consists of the following modules:

Module	Description
silk-core	Contains the Silk Link Discovery Engine along with classes to represent basic concepts in link discovery such as data sources and linkage rules
silk-evaluation	Contains utilities to evaluate generated links
silk-jena	Contains data sources to read from Jena Models
silk-learning	Machine learning of linkage rules
silk-singemachine	Silk Single Machine
silk-mapreduce	Silk MapReduce
silk-server	Silk Server
silk-workbench	Silk Workbench

Core Module

The core module contains the [Silk Link Discovery Engine](#) along with classes to represent basic concepts in link discovery such as data sources and linkage rules

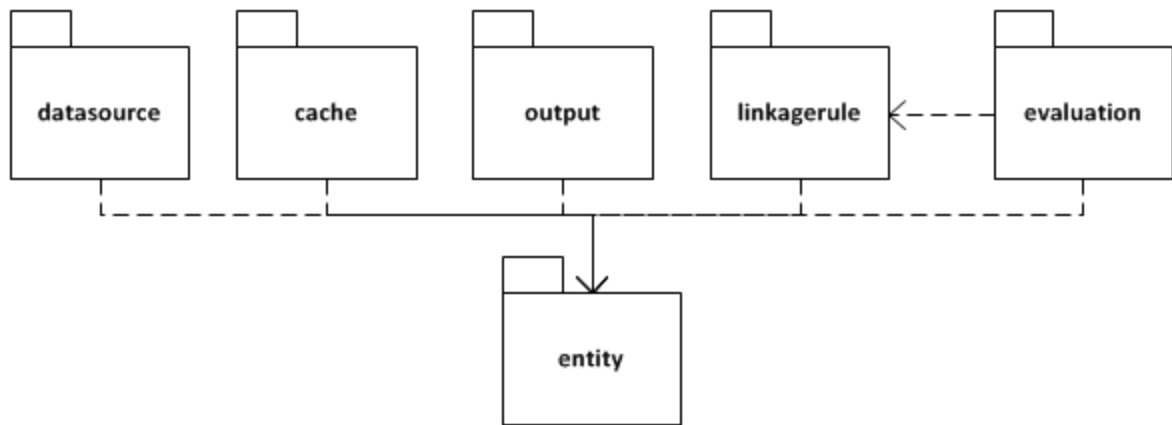
Package	Description
de.fuberlin.wiwiss.silk.entity	Contains classes to represent entities.
de.fuberlin.wiwiss.silk.cache	Provides the entity cache.
de.fuberlin.wiwiss.silk.datasource	Contains classes to read entities from external Data Sources
de.fuberlin.wiwiss.silk.linkagerule	Provides classes to represent linkage rules
de.fuberlin.wiwiss.silk.output	Contains classes to write generated links to external outputs
de.fuberlin.wiwiss.silk.plugins	Contains all plugins which are included in Silk by default.

Evaluation Module

The module contains utilities to evaluate the output of linkage rules. All classes are located in the package **de.fuberlin.wiwiss.silk.evaluation**.

Dependencies

This figure visualizes the dependencies between the core packages:



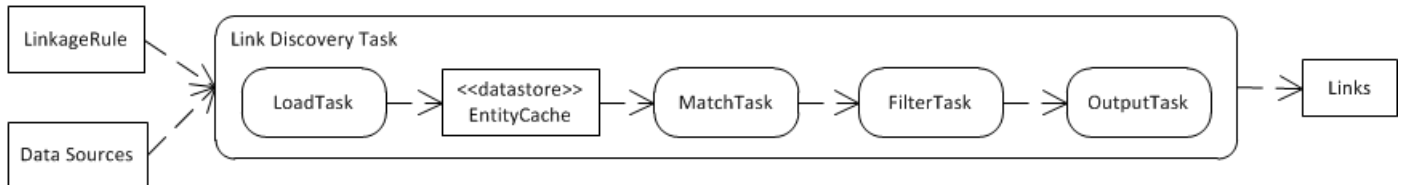
Tasks

The Silk API offers classes to execute common tasks in link discovery.

TODO: details

Link Discovery Task

TODO: description



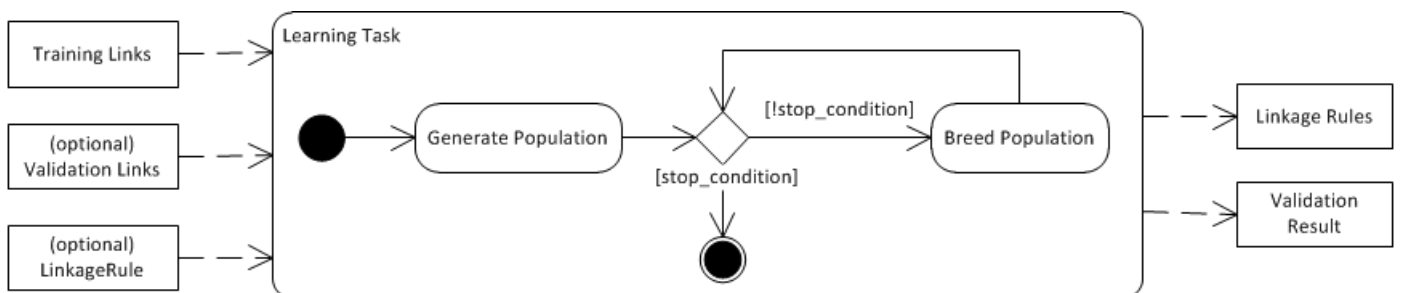
Evaluation Task

TODO: description



Learning Task

TODO: description



Silk Link Discovery Engine

Introduction

The Silk Link Discovery Engine builds the core of the Silk Framework. It is responsible for loading the instances from the data sources as well as generating the links based on the user-provided Link Specifications. This page covers the architecture of the Silk Link Discovery Engine as well as the Blocking method used in Silk.

Architecture



The **DataSource** generates a stream of data items.

The optional **Blocking** phase partitions the incoming data items in clusters.

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 datasets. 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 0.0 and 1.0. 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.

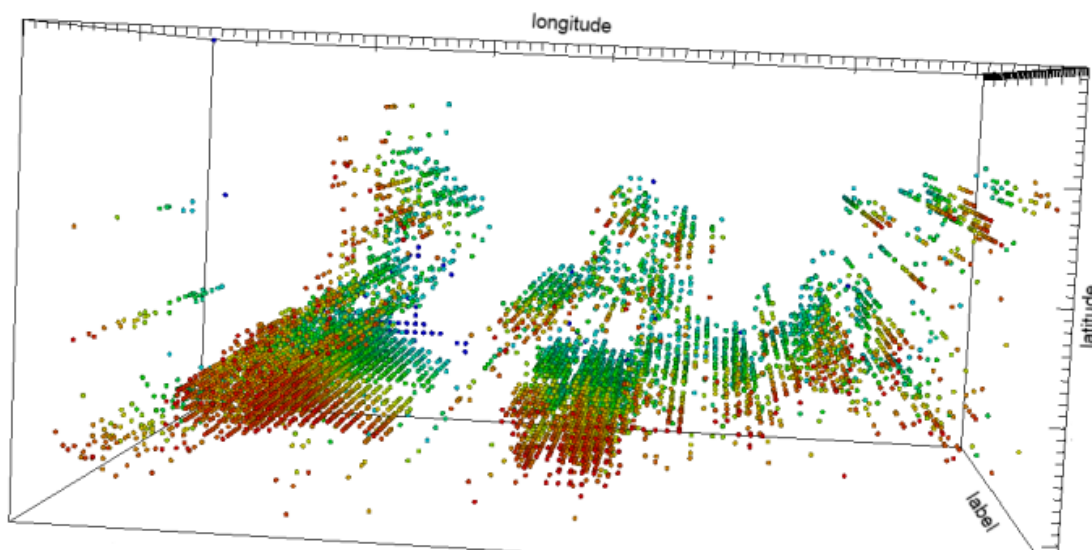
In the second stage, all links which originate from the same subject are grouped together. If a link limit is defined, only the links with the highest confidence are forwarded to the output. The number of links which are forwarded per source item, is specified by the link limit

Finally, the **Output** phase is responsible for writing the generated and filtered links to a user-defined destination.

Blocking

As the Web of Data is growing fast there is an increasing need for link discovery tools which scale to very large datasets. A number of methods have been proposed to improve the efficiency of link discovery by dismissing definitive non-matches prior to comparison. The most well-known method to achieve this is known as blocking. Unfortunately, traditional blocking methods need a separate configuration and in general lead to a decrease of recall due to false dismissals.

Silk employs a novel blocking method which maps entities to a multidimensional index. The basic idea of the mapping function is that it preserves the distances of the entities i.e. similar entities will be located near to each other in the index space. Blocking works on arbitrary link specifications which aggregate multiple different similarity measures such as string, geographic or date similarity. No separate configuration is required as the indexing is directly based on the link specification and all parameters are configured automatically. Additionally, it guarantees that no false dismissals and thus no loss of recall can occur.



Blocking is organized in three phases: *index generation*, *index aggregation* and *comparison pair generation*.

Index generation

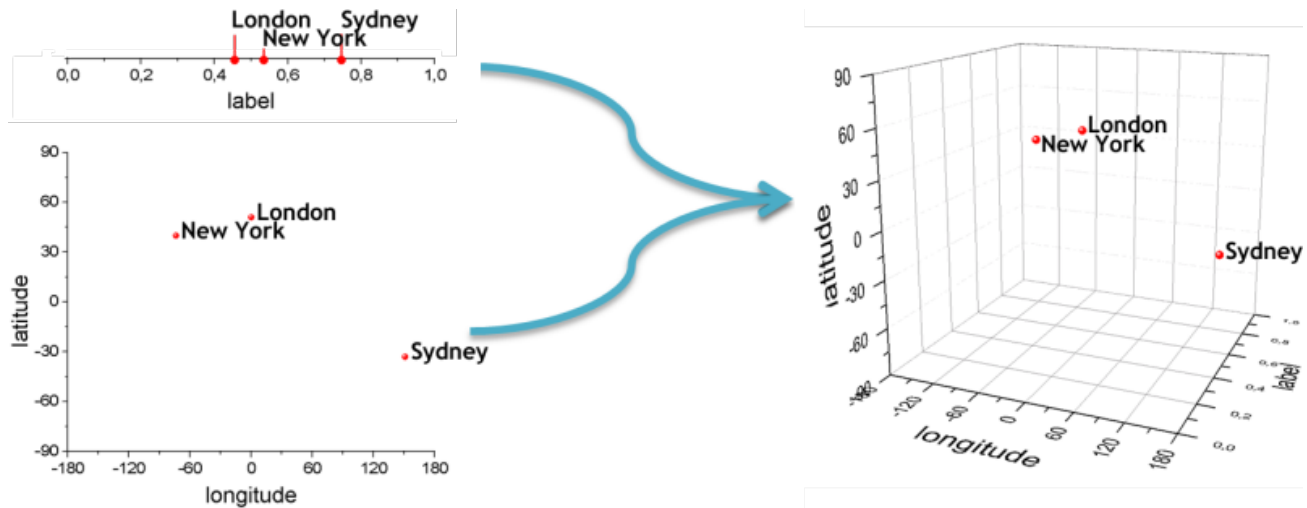
For each similarity measure in the link specification, an index is built which consists of a set of vectors which define locations in the Euclidean

space. The basic idea of the indexing method is that it preserves the distances of the entities i.e. similar entities will be located near each other in the index.

The specific indexing method which defines the number of index vectors which are generated per entity as well as their dimensionality depends on the data type of the field. For instance, for each numeric field a one-dimensional index is built and for each field which contains a geographic coordinate a two dimensional index is built using the latitude and longitude.

Index aggregation

In the index aggregation phase, all indexes which have been built in the index generation phase are aggregated into one compound index. The aggregation function preserves the property of the index that two entities within a given distance share the same index vector. Generally, aggregating the indexes of multiple similarity measures will lead to an increase in dimensionality, but the concrete aggregation function depends on the specific aggregation type. For instance, when aggregating an 2-dimensional geographic index and an 1-dimensional string index using an average aggregation, the resulting index will be 3-dimensional:



Comparison pair generation

Finally, the comparison pair generation employs the index to generate the set of entity pairs which are potential links. These pairs are then evaluated using the link specification to compute the exact similarity and determine the actual links.

Performance

This page discusses how to achieve maximum performance using Silk and presents a number of experiments. If you are experiencing performance problems, please follow the best practices below. If you still get a significantly worse performance than achieved in the presented experiments, please contact us so we can have a look at the cause.

Best Practices

Recommended Distance Measures

Silk is extensible and allows arbitrary distance measures to be plugged in. For this reason not all distance measures have the same level of maturity. The following distance measures are recommended to be used in performance critical applications:

- Equality/Inequality
- Levensthein Distance
- Jaccard Distance
- wgs84
- Numeric similarity

Only compare properties of a single language

Many databases such as DBpedia provide labels in multiple languages for each entity. In many use cases it is sufficient to compare only the english labels of the entities in the link specification. In this cases the matching performance can be improved significantly by using the language filter on the properties.

Example:

```
<Input path="?a/rdfs:label[@lang='en']" />
```

Prefer min/max aggregations over average aggregations

Experiments

All experiments have been executed using Silk 2.5.1 on a 3GHz Intel® Core i7 CPU with 4 cores while the heap has been restricted to 2GB.

Interlinking cities in DBpedia and LinkedGeoData

Input

- 101,928 settlements from DBpedia
- 560,123 settlements from LinkedGeoData

Linkage Rule

```
<LinkageRule>
  <Aggregate type="min">
    <Aggregate type="max" required="true" >
      <!-- We need two comparators because some resources in LinkedGeoData do not provide an english label -->
      <Compare metric="{see results}" threshold="{see results}">
        <Input path="?a/rdfs:label[@lang='en']" />
        <Input path="?b/rdfs:label[@lang='en']" />
      </Compare>
      <Compare metric="{see results}" threshold="{see results}">
```

```

    <Input path="?a/rdfs:label[@lang='en']" />
    <Input path="?b/rdfs:label[@lang='']" />
  </Compare>
</Aggregate>
<Compare metric="wgs84" threshold="30" required="true">
  <Input path="?a/wgs84:geometry" />
  <Input path="?b/wgs84:geometry" />
  <Param name="unit" value="km" />
</Compare>
</Aggregate>
</LinkageRule>

```

Results

Distance Measure	Distance Threshold	Generated Links	Runtime
levenshteinDistance	1	32,728	98 s
qGrams	0.1	30,969	114 s

Developer Documentation

How to checkout the code from git

For those not so familiar with git

```
1) Upload your ssh key to assembla and clone a repository
git clone git@git.assembla.com:silk.git
```

```
2) Make a new branch
git branch experiment
git checkout experiment
```

```
3) the folder structure
4) create a remote branch for yourself
git push origin master:refs/heads/my-branch-name
```

Then later you will push your changes to "my-branch-name" branch via following comm and

```
git push origin experiment:my-branch-name
```

Then later pull changes from master branch

```
git pull origin master
```

Add file to commit

```
git add filename
```

Commit changes

```
git commit -m"here the message"
```

Find out what is the status

```
git status
```

```
silk
|
+silk2
  |
  +doc
  +silk-jena
  +silk-singlemachine
  +silk-learning
  +silk-workbench
  +silk-core
  +silk-mapreduce
  +target
  +silk-evaluation
  +silk-server
  +pom.xml
```

How to compile and build workbench.war from commandline

For those less familiar with maven.

```
1) build required jars
cd silk/silk2
mvn clean install

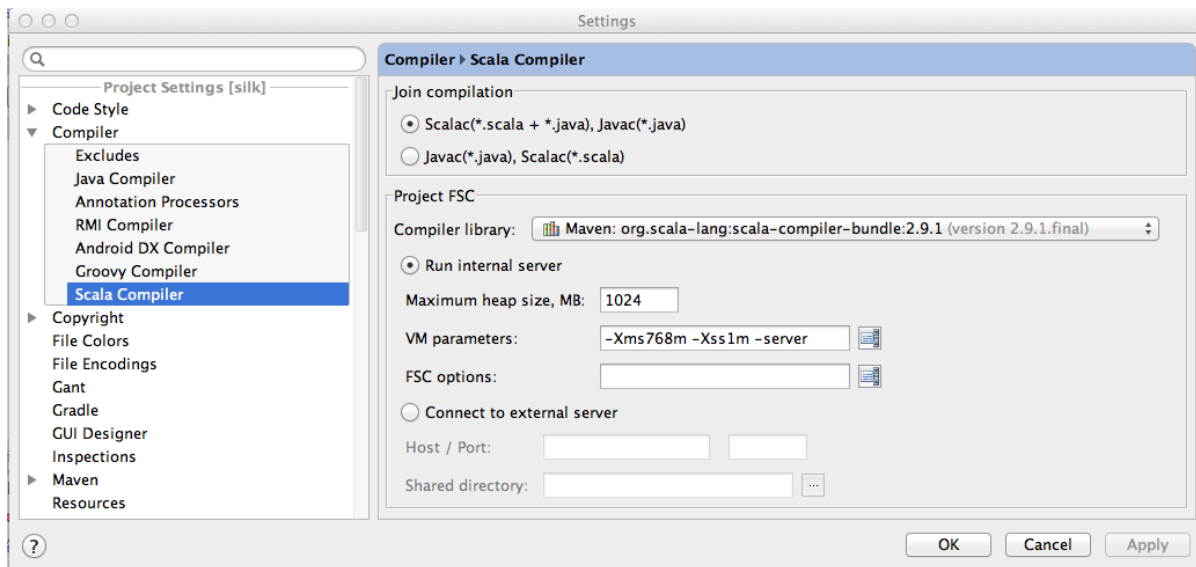
2) build the workbench war
cd silk-workbench/silk-workbench-webapp
mvn war:war
```

Start to code in IntelliJ IDEA

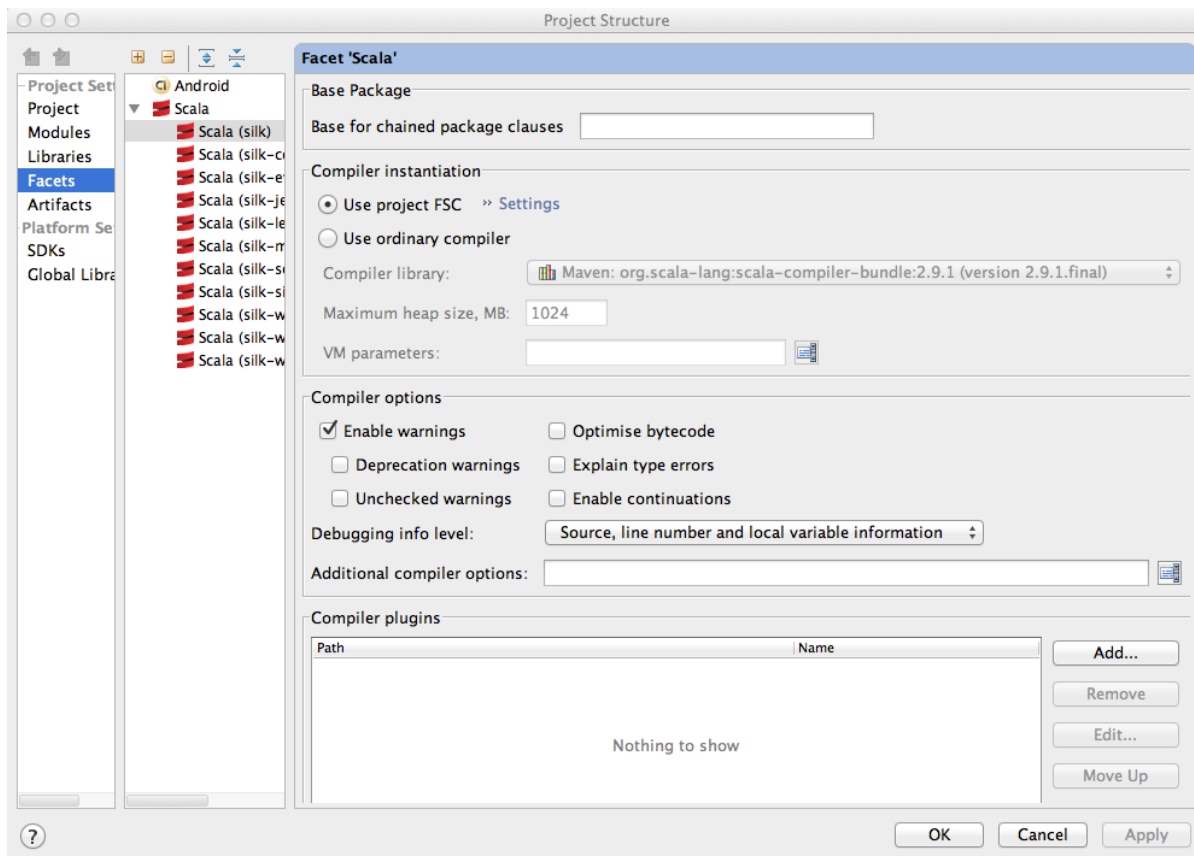
- 1) Download an IntelliJ IDEA free edition
- 2) Install scala plugin
- 3) IMPORTANT: Before you create new project from an existing source
cd silk/silk2
mvn idea:idea
- 4) Then open an IDE and create new project from maven project

- 5) OPTIONAL (but very handy) Enable **FSC** Fast Scala Compiler

- a) go to Settings->Project Settings->Compiler->Scala Compiler
and set up compiler library



- b) go to Project Structure->Facets->ScalaFacet
and select "Use project SFC" Do it for all modules



6) After all these steps you should be able to compile the project

How to prepare the workbench.war for Tomcat container

Currently the build is optimize to work with Jetty 7. Below is a list of modification which has to be done to make it work with Tomcat.

- Edit web.xml

```
edit silk-workbench/silk-workbench-webapp/src/main/webapp/WEB-INF/web.xml
comment out line for jetty uncomment line for tomcat

<pre><code>
...
<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class><!-- Tomcat -->
<!--<servlet-class>org.eclipse.jetty.servlet.DefaultServlet</servlet-class>--><!-- Jetty -->
...
</code></pre>
```

- Edit Main.scala

```
edit silk-workbench/silk-workbench-webapp/src/main/scala/de/fuberlin/wiwiss/silk/workbench/lift/Main.scala
comment out everything jetty related

<pre><code>
package de.fuberlin.wiwiss.silk.workbench.lift

//import org.eclipse.jetty.server.Server
//import org.eclipse.jetty.webapp.WebAppContext
```



```

/**
 * Starts the Workbench.
 */
object Main {
  def main(args : Array[String]) {
    /*
    val server = new Server(8080)
    val webapp = new WebApplicationContext();
    webapp.setContextPath("/");
    val protectionDomain = Main.getClass.getProtectionDomain
    val location = protectionDomain.getCodeSource.getLocation.toExternalForm
    webapp.setWar(location);
    server.setHandler(webapp);

    server.start()
    */
  }
}

```

- edit pom.xml find and comment out jetty dependencies

```

<pre><code>
...
<!--
  <dependency>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-server</artifactId>
    <version>7.4.5.v20110725</version>
  </dependency>
  <dependency>
    <groupId>org.eclipse.s</groupId>
    <artifactId>jetty-webapp</artifactId>
    <version>7.4.5.v20110725</version>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-continuation</artifactId>
    <version>7.4.5.v20110725</version>
  </dependency>
  -->
....
</code></pre>

```

- Now you can build the war file the usual way

```
mvn war:war
```

- prepare Tomcat as the silk all over the place use "user.home" system variable overwrite the user.home by modifying /etc/default/tomcat6

```

JAVA_OPTS="${JAVA_OPTS} -Duser.home=/home/sindice/silk"
where /home/sindice/silk is the directory where tomcat user can write

```

- set also `SILK_WORKBENCH_CONFIG_PATH` if you want to use multiuser mode

```
JAVA_OPTS="${JAVA_OPTS} -DSILK_WORKBENCH_CONFIG_PATH=/home/sindice/silk/config"
```

- copy generated war into tomcat container

```
cp target/silk-workbench-webapp-2.5.war $CATALINA_HOME/webapps/workbench.war
```

- Done. Now you should see workbench running at `http://example.com:8080/workbench`

Publications

- Robert Isele, Christian Bizer: [Learning Linkage Rules using Genetic Programming](#). 6th International Workshop on Ontology Matching. Bonn, Germany, October 2011.
- Robert Isele, Anja Jentzsch, Christian Bizer: [Efficient Multidimensional Blocking for Link Discovery without losing Recall](#). 14th International Workshop on the Web and Databases (WebDB 2011), SIGMOD2011, Athens, Greece, June 2011.
- Robert Isele, Anja Jentzsch, Christian Bizer: [Silk Server – Adding missing Links while consuming Linked Data](#). 1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai, China, November 2010.
- Anja Jentzsch, Robert Isele, Christian Bizer: [Silk – Generating RDF Links while publishing or consuming Linked Data](#). Poster at the International Semantic Web Conference (ISWC2010), Shanghai, China, November 2010.
- Julius Volz, Christian Bizer, Martin Gaedke, Georgi Kobilarov: [Discovering and Maintaining Links on the Web of Data](#). International Semantic Web Conference (ISWC2009), Westfields, USA, October 2009.
- Julius Volz, Christian Bizer, Martin Gaedke, Georgi Kobilarov: [Silk – A Link Discovery Framework for the Web of Data](#). 2nd Workshop about Linked Data on the Web (LDOW2009), Madrid, Spain, April 2009.
- Christian Bizer, Tom Heath, Tim Berners-Lee: [Linked Data – The Story So Far](#). In: International Journal on Semantic Web & Information Systems, Vol. 5, Issue 3, Pages 1-22, 2009.