RapidMiner Linked Open Data Extension

Manual

Version 1.5, 09/19/14

Heiko Paulheim

Petar Ristoski

Evgeny Mitichkin

Christian Bizer

University of Mannheim

Data and Web Science Group

Contact: use the Google Group
https://groups.google.com/forum/#!forum/rmlod
rmlod@googlegroups.com

Table of Contents

If you use the RapidMiner Linked Open Data Extension for scientific purposes, please kindly include a reference to the following paper:

H. Paulheim, P. Ristoski, E. Mitichkin, C. Bizer (2014): *Data Mining with Background Knowledge from the Web.* In: RapidMiner World, 2014.

1 Introduction

Linked Open Data[1] is a collection of freely accessible, machine interpretable data on the web. It uses semantic web standards like RDF for data representation (Bizer et al., 2009). As of today, there are several hundred datasets from various domains, encompassing general knowledge as well as specific domains such as government, geographic, and life science data, scientific publications, and media (Schmachtenberg et al., 2014). A detailed list of datasets that are available as Linked Open Data can be obtained at http://lod-cloud.net/.

Essentially, Linked Open Data contains data as graphs with labeled edges, where each entity is represented by a dereferencable URI. Each statement can be understood as a triple with subject, predicate, and object, such as

<http://dbpedia.org/resource/Germany> <http://dbpedia.org/ontology/capital> <http://dbpedia.org/resource/Berlin> .

The RapidMiner Linked Open Data extension leverages those data sources both to create useful datasets for Data Mining, as well as for adding background knowledge to existing datasets. For example, on a dataset for countries, data like the population, area, and GDP can be added from DBpedia[2] (Lehmann et al., 2013) in order to improve the results of the data mining process.

In particular, the extension works with datasets that provide a SPARQL endpoint, i.e., a web service which delivers data using the query language SPARQL.[3] A list of such datasets is available at datahub.io. Furthermore, datasets not offering SPARQL endpoints may also be used via dereferencing URIs.

Furthermore, data from Linked Open Data sources can be used as input to the data mining process, e.g., reading data from the Eurostat Linked Open Data endpoint and running analysis on the data with RapidMiner.

Important notice: The RapidMiner LOD Extension usually accesses live data from the web. This requires a live internet connection and can result in longer operator runtimes, depending on the dataset and task at hand.

For larger datasets, the runtimes of can become rather long, depending on the dataset and specific operators used. Since in a typical setting, you will first generate data and then experiment with a larger number of subsequent operators (e.g., classification or regression algorithms), we suggest that you store the output dataset of the LOD operator chain in an intermediate file (e.g., CSV) or database, and perform all subsequent analysis on that stored dataset. That way, you will have to run the time consuming access to Linked Open Data only once.

1http://lod-cloud.net/

2http://dbpedia.org/

3http://www.w3.org/TR/sparql11-overview/

2 Use Cases

The RapidMiner LOD extension, sometimes in conjunction with further RapidMiner extensions, can be used for various purposes. In the following, we give a list of use cases that the extension was already used for, each with a reference to a published paper in which further details can be found.

Using Linked Open Data in mining existing datasets:

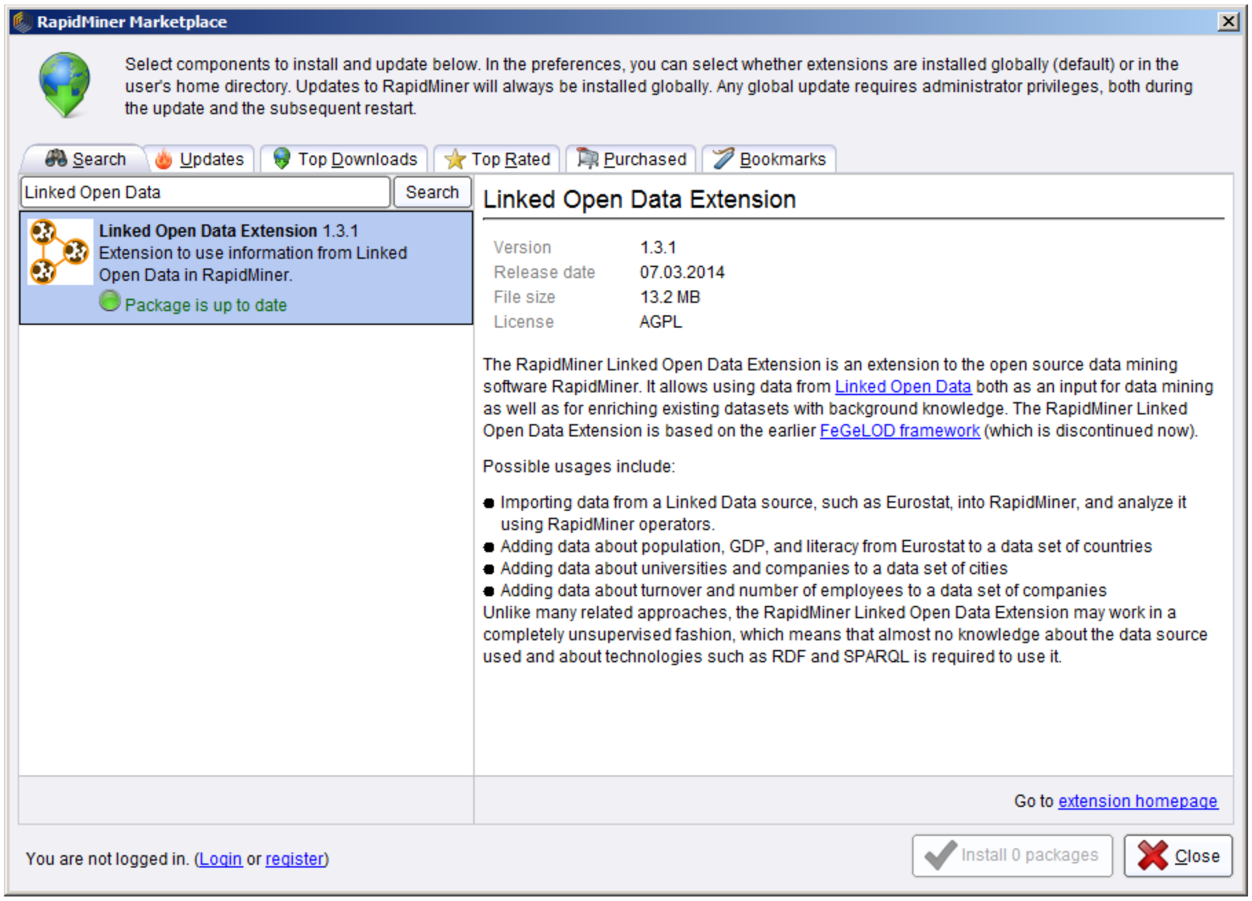
- *Generating explanations for statistics.* We use a statistic as an input, e.g., unemployment figures from different regions, and add new attributes about the regions from dataset in LOD. Based on that enhanced dataset, possible explanations can be found, e.g., by correlation analysis (Paulheim, 2012; Ristoski and Paulheim, 2013)
- *Classifying Tweets:* We process twitter messages with DBpedia Spotlight and extract further categories for the entities found. Based on those categories, we can tell whether a twitter message reports about a certain type of event (e.g., a traffic accident) or not (Schulz et al., 2013b)
- *Predicting the location of Twitter users:* We process twitter user profiles and messages with DBpedia Spotlight to extract locations. Those are used as an indicator to predict a twitter user's location (Schulz et al., 2013a)
- *Classifying event information extracted from Wikipedia:* In a dataset of events extracted from Wikipedia, we predict the category of events (such as politics, sports, etc.) based on category information of entities involved in the event (Hienert et al., 2012)

Using Data Mining for Improving and/or Debugging Linked Open Data:

- *Identifying wrong links in Linked Open Data.* Given a set of links between two datasets, we extract features for both linked entities. Assuming that the majority of links are correct, wrong links can be identified using outlier detection methods (Paulheim, 2014).
- *Schema Matching for Linked Open Data*. We extract types from a set of interlinked instances in two datasets, and use association rule mining to find mappings between the schemas used by the datasets (Janssen et al., 2012)

3 Enabling the RapidMiner LOD Extension

1. Searching for “Linked Open Data” in the RapidMiner Marketplace dialog will find the extension and allow you to install it:



Select components to install and update below. In the preferences, you can select whether extensions are installed globally (default) or in the user's home directory. Updates to RapidMiner will always be installed globally. Any global update requires administrator privileges, both during the update and the subsequent restart.

Linked Open Data Extension 1.3.1
Extension to use information from Linked Open Data in RapidMiner.
Package is up to date

Linked Open Data Extension

Version	1.3.1
Release date	07.03.2014
File size	13.2 MB
License	AGPL

The RapidMiner Linked Open Data Extension is an extension to the open source data mining software RapidMiner. It allows using data from Linked Open Data both as an input for data mining as well as for enriching existing datasets with background knowledge. The RapidMiner Linked Open Data Extension is based on the earlier FeGeLOD framework (which is discontinued now).

Possible usages include:

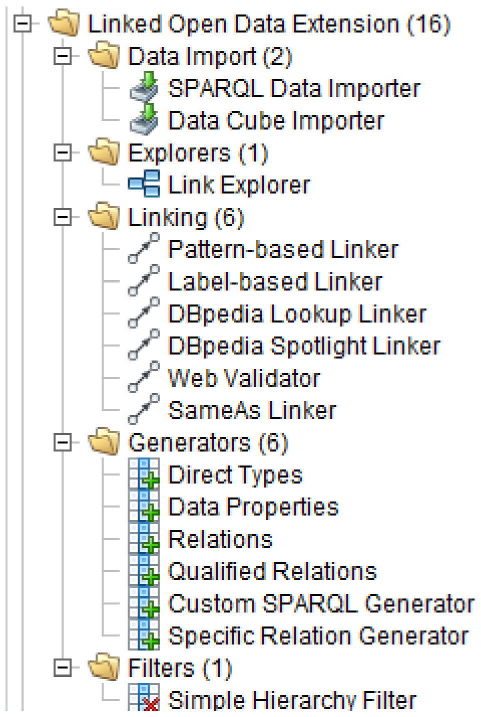
- Importing data from a Linked Data source, such as Eurostat, into RapidMiner, and analyze it using RapidMiner operators.
- Adding data about population, GDP, and literacy from Eurostat to a data set of countries
- Adding data about universities and companies to a data set of cities
- Adding data about turnover and number of employees to a data set of companies

Unlike many related approaches, the RapidMiner Linked Open Data Extension may work in a completely unsupervised fashion, which means that almost no knowledge about the data source used and about technologies such as RDF and SPARQL is required to use it.

Go to extension homepage

You are not logged in. (Login or register)

2. Create a new process. You will see a new entry (Linked Open Data Extension) in the catalog available, showing the operators from the extension:



- Linked Open Data Extension (16)
  - Data Import (2)
    - SPARQL Data Importer
    - Data Cube Importer
  - Explorers (1)
    - Link Explorer
  - Linking (6)
    - Pattern-based Linker
    - Label-based Linker
    - DBpedia Lookup Linker
    - DBpedia Spotlight Linker
    - Web Validator
    - SameAs Linker
  - Generators (6)
    - Direct Types
    - Data Properties
    - Relations
    - Qualified Relations
    - Custom SPARQL Generator
    - Specific Relation Generator
  - Filters (1)
    - Simple Hierarchy Filter

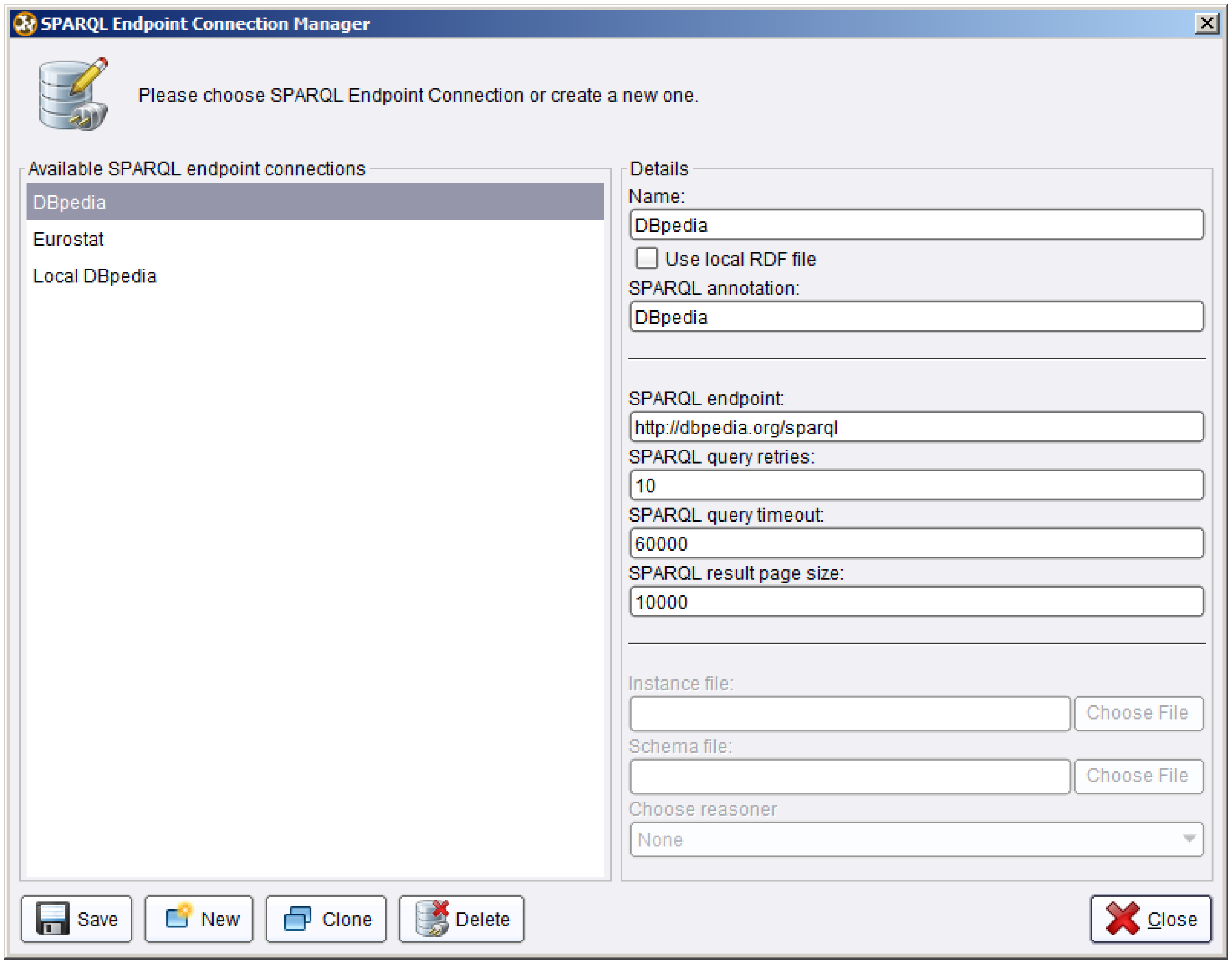
Note that most operators of the extension use an internet connection. In case you are behind a *proxy*, we use RapidMiner's proxy settings,

4 Configuring the Extension

4.1 Configuring SPARQL Endpoints

Most Linked Open Data sources provide a SPARQL endpoint. These endpoints allow for querying the underlying dataset using a standardized query language, i.e., SPARQL[4]. Most of the operators use this query language internally. Although they are also capable of using the datasets in a different mode, i.e., dereferencing URIs (see below), it is advised to use SPARQL where available, as this will reduce the runtime of most operators.

To use SPARQL in the operators, you will first have to define SPARQL Endpoints for the datasets you are planning to use. To that end, go to the menu "Tools"->"Manage SPARQL Endpoint Connections…". The SPARQL Endpoint Configuration Dialog Opens:



SPARQL Endpoint Connection Manager

Please choose SPARQL Endpoint Connection or create a new one.

Available SPARQL endpoint connections
- DBpedia
- Eurostat
- Local DBpedia

Details

Name:	DBpedia
Use local RDF file	☐
SPARQL annotation:	DBpedia
SPARQL endpoint:	http://dbpedia.org/sparql
SPARQL query retries:	10
SPARQL query timeout:	60000
SPARQL result page size:	10000
Instance file:	
Schema file:	
Choose reasoner	None

4http://www.w3.org/TR/sparql11-overview/

You can create new SPARQL Endpoints, as well as change the settings for existing ones.

The configuration parameters for SPARQL Endpoints are:

- Name: the SPARQL Endpoint will be selectable later by that name
- SPARQL Endpoint: the URL of the service
- SPARQL Annotation: a string that will be used for naming attributes generated from this SPARQL endpoint

Three more expert parameters are available (i.e., they appear only if RapidMiner is executed in expert mode):

- Paging Size: many SPARQL endpoints have limitations w.r.t. to the amount of data they deliver. Paging ensures that all data is retrieved. For example, with a paging size value of 1000, a SELECT query is broken down into several queries like
  ```
  SELECT … LIMIT 1000 OFFSET 0
  SELECT … LIMIT 1000 OFFSET 1000
  SELECT … LIMIT 1000 OFFSET 2000
  ```
  etc.
  A Paging Size value of 0 disables paging.
- Query timeout defines the time which a SPARQL endpoint is given to respond (in milliseconds)
- Query retries defines the number of times a query is retried. If there are temporary errors, e.g., network problems, simply re-running the same query often resolves the issue.

Once the SPARQL Endpoint is defined, it may be used in subsequent operators. SPARQL Endpoint definitions for DBpedia and Eurostat are included as preconfigured endpoints.

The RapidMiner Linked Open Data extension also allows for local RDF files instead of remote SPARQL endpoints. To use a local RDF file, activate the corresponding checkbox. The options for local RDF files are:

- Specifying an instance and (optionally) a schema file
- Selecting a reasoner. The Jena built-in reasoners for RDFS as well as OWL Mini, OWL Micro and OWL Full are supported.[5]

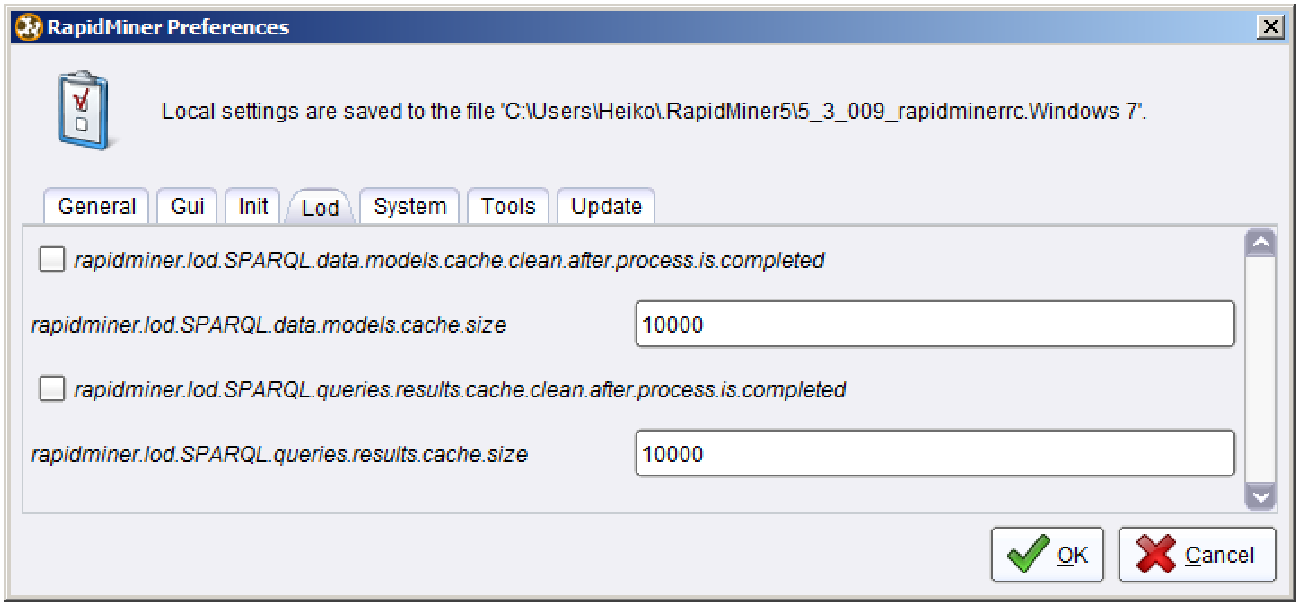
4.2 Caching Options

Since data from the web is used in the extension, some operators may run for a longer time. To mitigate that problem, the extension supports caching. Two different types of caches are used:

- *models* are cached when loading data from local files or from URLs
- *SPARQL results* are cached whenever loading data with a SPARQL query

For both caches, the size of the cache can be specified, as well as whether the cache should be emptied once a process has terminated, or persist as long as RapidMiner runs. The options are available in the Tools->Preferences dialog under the “Lod” tab:

5See http://jena.apache.org/documentation/inference/


RapidMiner Preferences

Local settings are saved to the file 'C:\Users\Heiko\.RapidMiner5\5_3_009_rapidminerrc.Windows 7'.

☐ *rapidminer.lod.SPARQL.data.models.cache.clean.after.process.is.completed*

rapidminer.lod.SPARQL.data.models.cache.size 10000

☐ *rapidminer.lod.SPARQL.queries.results.cache.clean.after.process.is.completed*

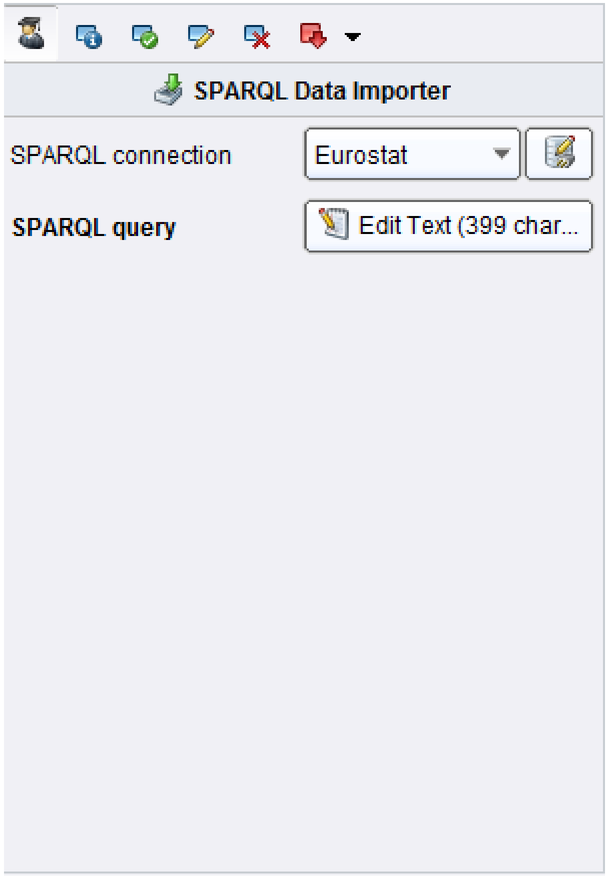
rapidminer.lod.SPARQL.queries.results.cache.size 10000

As a general recommendation, caches should be cleaned after the process finishes if subsequent processes use different datasets, since the memory consumption will drastically increase otherwise. However, if several processes are run on the same datasets, they will be significantly faster if keeping the caches over different processes.

In any case, caches will be cleared when RapidMiner is terminated.

5 Using Linked Open Data as Input for Data Mining

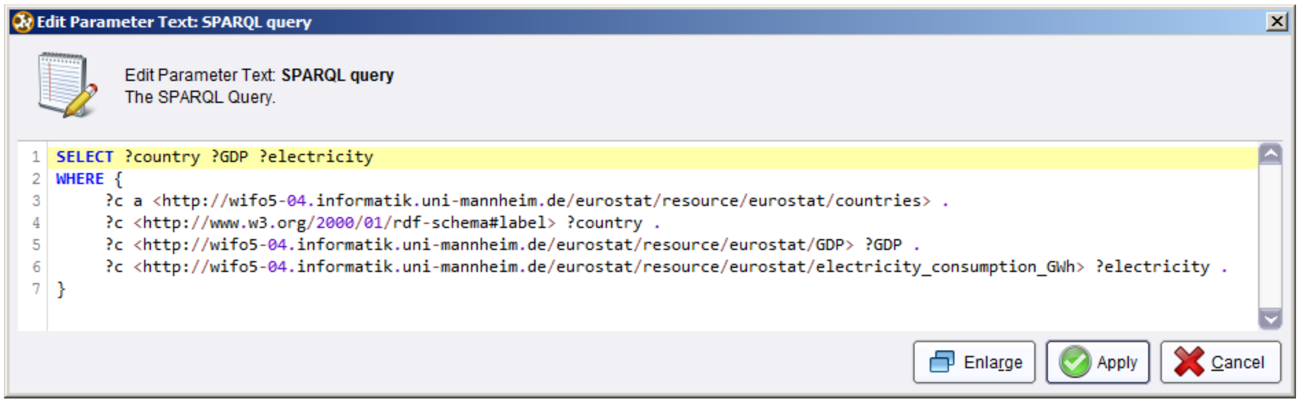
The RapidMiner Linked Open Data Extension provides an import operator to read data from a Linked Open Data source into a RapidMiner table. The SPARQL Data Importer uses a defined SPARQL Endpoint and a custom query to generate a RapidMiner table:



SPARQL connection: Eurostat

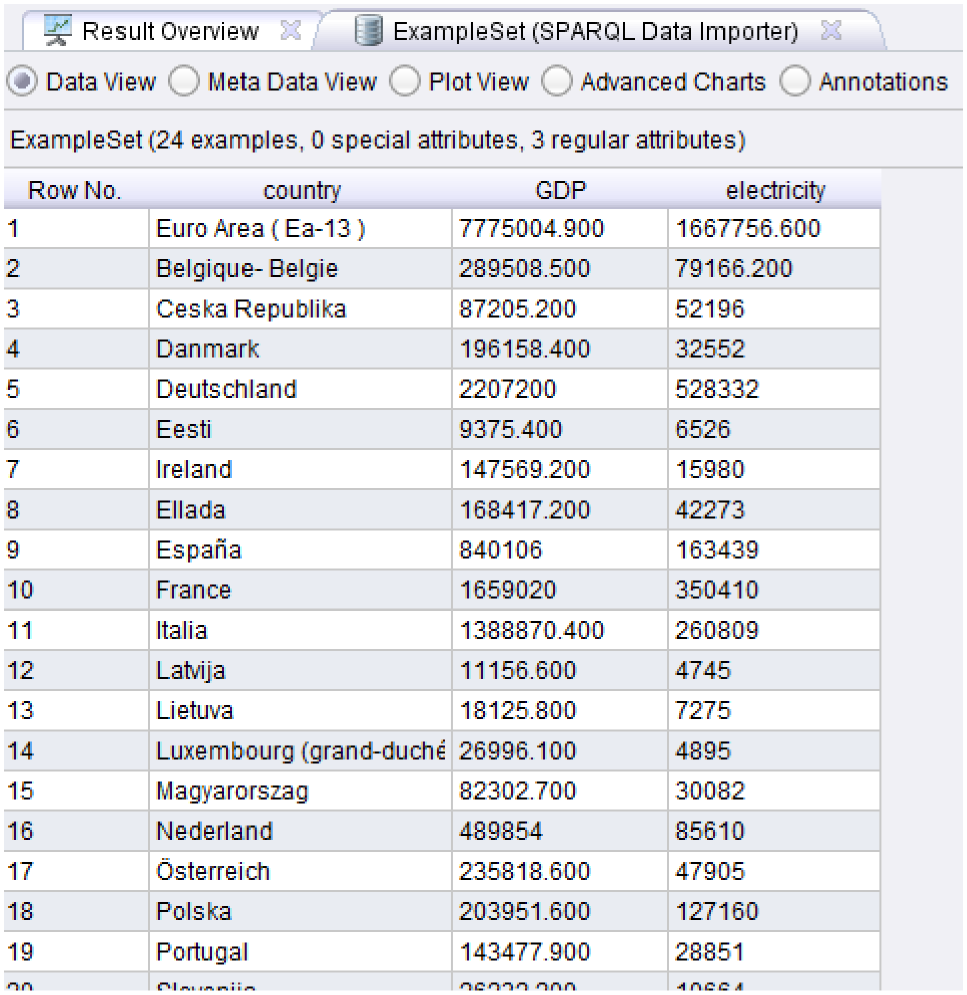
SPARQL query: Edit Text (399 char...

The SPARQL query is defined as follows:


Edit Parameter Text: SPARQL query
Edit Parameter Text: SPARQL query
The SPARQL Query.
Enlarge
Apply
Cancel

```
SELECT ?country ?GDP ?electricity
WHERE {
     ?c a <http://wifo5-04.informatik.uni-mannheim.de/eurostat/resource/eurostat/countries> .
     ?c <http://www.w3.org/2000/01/rdf-schema#label> ?country .
     ?c <http://wifo5-04.informatik.uni-mannheim.de/eurostat/resource/eurostat/GDP> ?GDP .
     ?c <http://wifo5-04.informatik.uni-mannheim.de/eurostat/resource/eurostat/electricity_consumption_GWh> ?electricity .
}
```

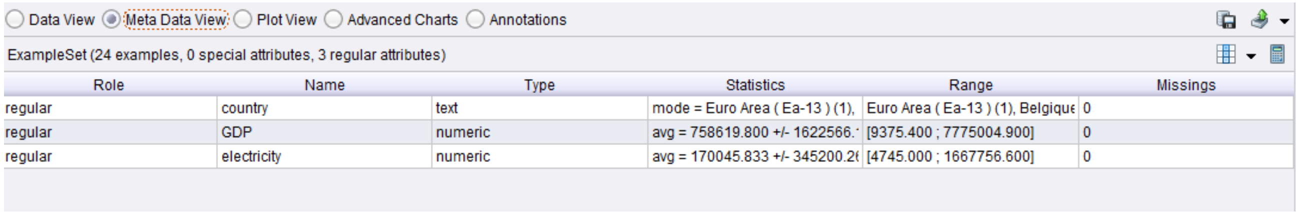
The table which is generated in the above example has three columns, named after the SPARQL query variables, i.e., country, GDP, and electricity. The output of the operator thus looks as follows:


Result Overview
ExampleSet (SPARQL Data Importer)
[✓ Data View] Meta Data View Plot View Advanced Charts Annotations

ExampleSet (24 examples, 0 special attributes, 3 regular attributes)

Row No.	country	GDP	electricity
1	Euro Area (Ea-13)	7775004.900	1667756.600
2	Belgique- Belgie	289508.500	79166.200
3	Ceska Republika	87205.200	52196
4	Danmark	196158.400	32552
5	Deutschland	2207200	528332
6	Eesti	9375.400	6526
7	Ireland	147569.200	15980
8	Ellada	168417.200	42273
9	España	840106	163439
10	France	1659020	350410
11	Italia	1388870.400	260809
12	Latvija	11156.600	4745
13	Lietuva	18125.800	7275
14	Luxembourg (grand-duché	26996.100	4895
15	Magyarorszag	82302.700	30082
16	Nederland	489854	85610
17	Österreich	235818.600	47905
18	Polska	203951.600	127160
19	Portugal	143477.900	28851

In the meta data view, you will observe that the operator automatically assigns meaningful data types to the values read from the SPARQL Endpoint. In that case, the country name is a text attribute, while GDP and electricity are numeric:


Data View [✓ Meta Data View] Plot View Advanced Charts Annotations

ExampleSet (24 examples, 0 special attributes, 3 regular attributes)

Role	Name	Type	Statistics	Range	Missings
regular	country	text	mode = Euro Area (Ea-13) (1),	Euro Area (Ea-13) (1), Belgique	0
regular	GDP	numeric	avg = 758619.800 +/- 1622566.	[9375.400 ; 7775004.900]	0
regular	electricity	numeric	avg = 170045.833 +/- 345200.2	[4745.000 ; 1667756.600]	0

When writing SPARQL statements, common prefixes are automatically resolved via a lookup in the prefix.cc catalogue.[6] Thus, it is possible to use shorthand notations such as `foaf:name` or `rdf:type` without explicitly declaring a prefix.

6http://prefix.cc/

6 Linking your Data to Existing Datasets

For using Linked Open Data as background knowledge for existing datasets, the first step is to link your data to a dataset from which you want to use additional data. For example, your dataset may contain a column "Country" with values "Germany", "France", etc. Linking your dataset to Linked Open Data means that an additional column is added, which contains URIs identifying those entities in Linked Open Data, such as http://dbpedia.org/resource/Germany, http://dbpedia.org/resource/France, etc.

The current version of the RapidMiner Linked Open Data extension comprises two linking methods:

1. The Pattern-based Linker builds links using a defined URI pattern. For example, the contents of the column "Country" are appended to a fixed string, e.g., "http://dbpedia.org/resource/".
2. The Label-based linker searches for the contents of a column (and optionally word n-grams) in a data source, and finds the most suitable resource based on their labels. For example, for a string "United States of America", the linker would search for word n-grams such as "United States" as well, and from all the results, use the best suited one.

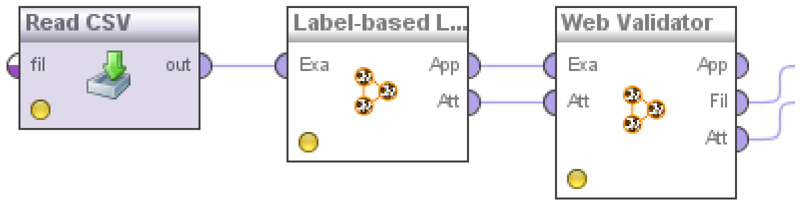
Each linker has two output ports:

1. App contains the data with the new column depicting the URI
2. Att contains the attribute name which holds the URIs

The column is named ORIGINAL_COLUMN_link_to_SPARQL_ENDPOINT_ANNOTATION, where ORIGINAL_COLUMN is the name of the column where the values come from (e.g., City), and SPARQL_ENDPOINT_ANNOTATION is the value set when configuring the SPARQL endpoint. For example, linking a column City to DBpedia will result in a new column City_link_to_DBpedia.

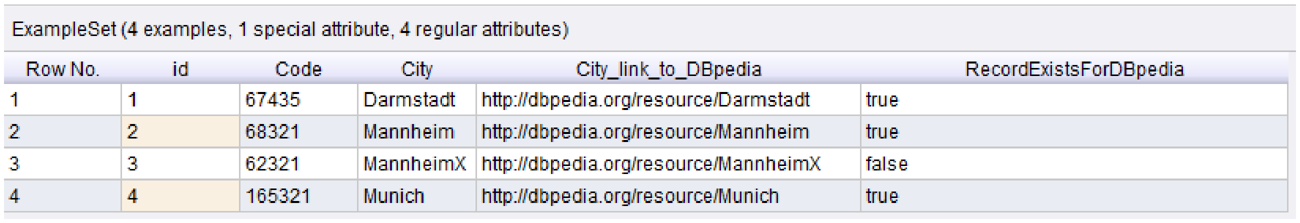
The Att port has to be connected to subsequent steps, e.g., generators, to inform them on which attribute to perform data generation.

Since some naïve linkers (such as the Label-based Linker) may generate links that actually do not exist, the Web Validator checks the existence of each link and removes instances for which the link does not exist. Furthermore, it removes all instances for which no link could be found. The full pipeline of linking looks as follows:



The Web Validator has three outputs: App is the original data with an additional attribute boolean flag indicating whether the link is valid or not, Fil is the filtered data with instances removed for which no link is found, and Att bypasses the original Att input port.

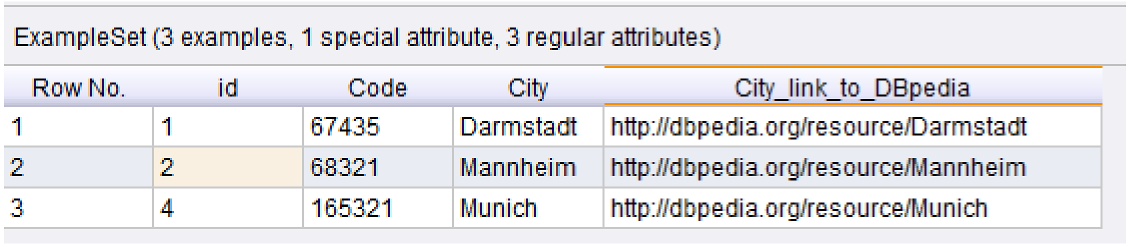
Assume that the original set contains an entity which cannot be resolved in DBpedia, i.e., "MannheimX". Linking that dataset to DBpedia with the simple linker results in the following table:



ExampleSet (4 examples, 1 special attribute, 4 regular attributes)

Row No.	id	Code	City	City_link_to_DBpedia	RecordExistsForDBpedia
1	1	67435	Darmstadt	http://dbpedia.org/resource/Darmstadt	true
2	2	68321	Mannheim	http://dbpedia.org/resource/Mannheim	true
3	3	62321	MannheimX	http://dbpedia.org/resource/MannheimX	false
4	4	165321	Munich	http://dbpedia.org/resource/Munich	true

The output produced by the Web Validator operator then filters the table as follows:



ExampleSet (3 examples, 1 special attribute, 3 regular attributes)

Row No.	id	Code	City	City_link_to_DBpedia
1	1	67435	Darmstadt	http://dbpedia.org/resource/Darmstadt
2	2	68321	Mannheim	http://dbpedia.org/resource/Mannheim
3	4	165321	Munich	http://dbpedia.org/resource/Munich

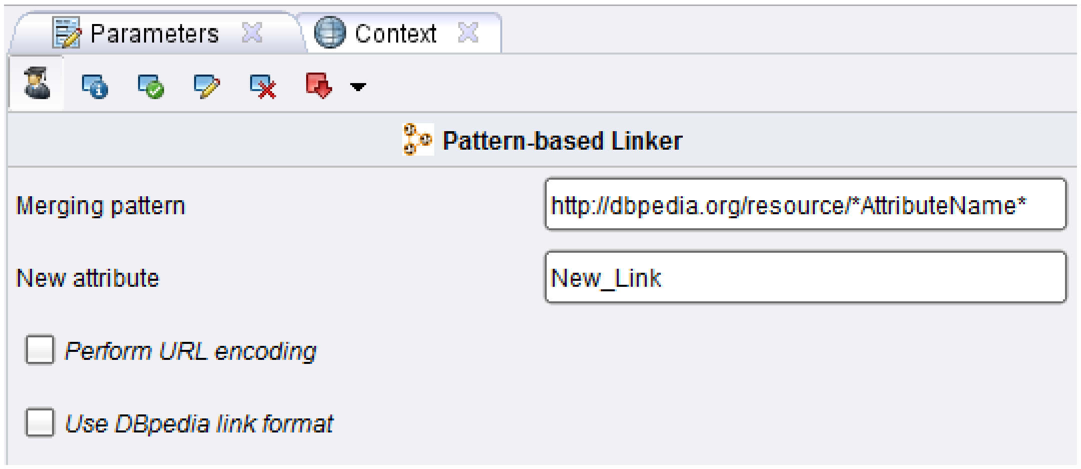
Observation: the entity MannheimX, for which the DBpedia link is wrong, is removed from the dataset.

6.1 The Pattern-based Linker

The Pattern-based linker provides a simple method for creating links, which simply guesses links by forming simple URI patterns. The linker takes three arguments:

1. The link pattern
2. The name of the new attribute

Links are created by replacing attribute values in the pattern. In the example below, the values of the attribute name replace the string *AttributeName*. Multiple placeholders may be used in one pattern.



Pattern-based Linker

Merging pattern	http://dbpedia.org/resource/*AttributeName*
New attribute	New_Link
☐ *Perform URL encoding*	
☐ *Use DBpedia link format*	

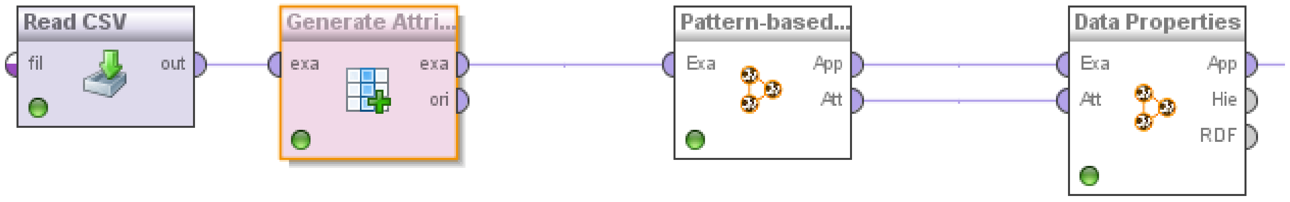
Further attributes that can be set are whether to perform URL Encoding (i.e., replacing special characters and creating proper UTF-8 links), and whether to use a specific link format for DBpedia (e.g., representing blanks as underscores).

In the example shown above, a column “City” containing values such as “Mannheim”, “Darmstadt” etc. would result in the corresponding links “http://dbpedia.org/resource/Mannheim”, “http://dbpedia.org/resource/Darmstadt”, etc.

If additional preprocessing is needed, it can be done with other RapidMiner operators. For example, the RDF book mashup dataset has links based on book ISBNs, such as

http://wifo5-03.informatik.uni-mannheim.de/bookmashup/books/006251587X

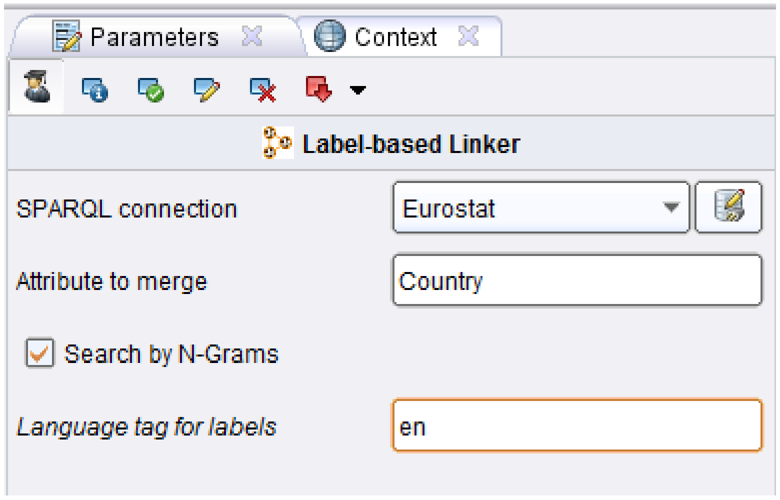
However, a dataset may contain ISBNs with dashes instead, such as 0-06-251587-X. The solution is to use RapidMiner's Generate Attribute operator first, creating a new attribute using the formula replace(isbn,"-",""), and then to use that new attribute in the linker:



6.2 The Label-based Linker

The Label-based Linker identifies entities by issuing SPARQL queries against an endpoint, trying to find entities whose labels contain a given string. It takes four parameters:

1. A SPARQL endpoint
2. The attribute to merge (see Pattern-based linker)
3. A flag indicating whether the search should include n-grams (see below)



Label-based Linker

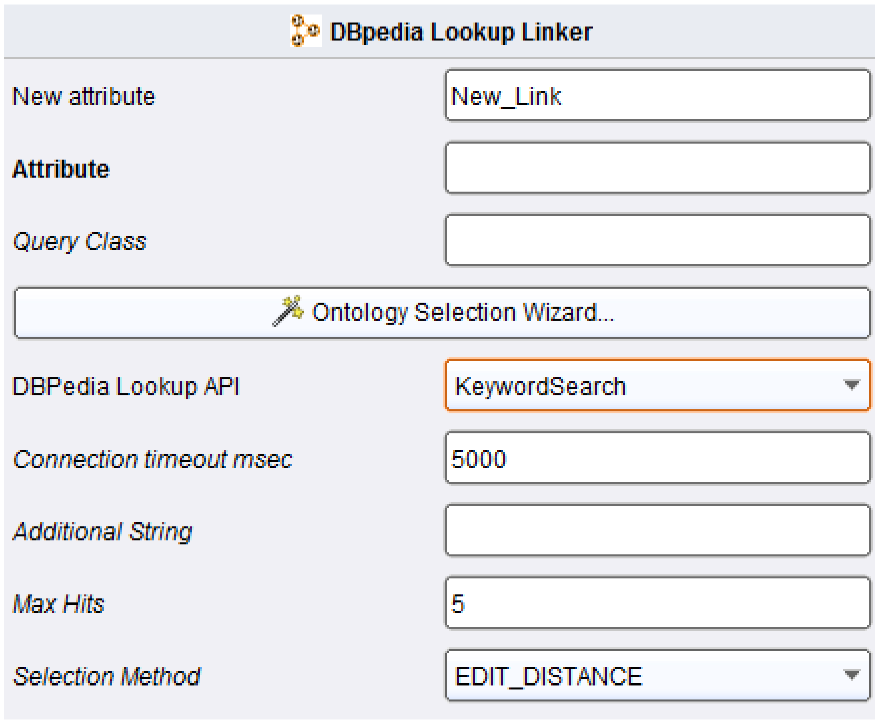
SPARQL connection	Eurostat
Attribute to merge	Country
☑ Search by N-Grams	
Language tag for labels	en

When including n-grams in the search, the linker will first try to identify entities whose labels contain the string in its entirety, e.g., “United States of America”. If this does not succeed, it will continue with sub tokens, eventually also finding entities whose label is only “United States”.

Note that the SPARQL Based linker, in particular when using the “Search by N-Grams” option, can take some time to link a dataset, since it performs full text search via SPARQL, which, depending on the endpoint used, may not be performed very fast.

In expert mode, you can also restrict the search to labels with a certain language tag, e.g., “en”. If the attribute is left empty (which is the default), labels in all languages, as well as without any language tag, are included in the search.

6.3 The DBpedia Lookup Linker


DBpedia Lookup Linker

New attribute	New_Link
Attribute	
Query Class	
Ontology Selection Wizard...	
DBPedia Lookup API	KeywordSearch
Connection timeout msec	5000
Additional String	
Max Hits	5
Selection Method	EDIT_DISTANCE

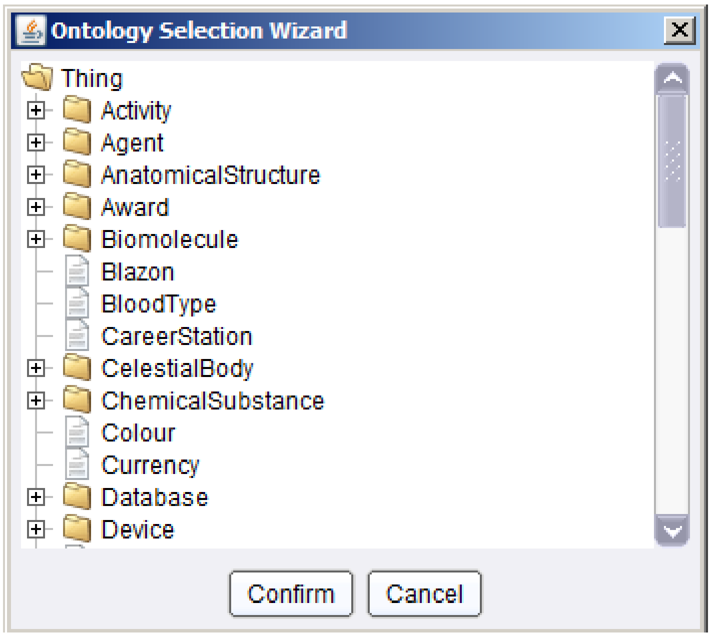
DBpedia Lookup[7] is a web service which allows search over DBpedia. It can be used as an operator for linking and is more versatile the pattern-based and the label-based linker.

There are various parameters for the lookup linker:

- *New Attribute* is the name of the attribute containing the link
- *Attribute* is the name of the attribute that contains the text value
- *QueryClass* is an optional parameter which can be used to narrow the search results. It is a class name from the DBpedia ontology[8], e.g., “City”. If the attribute to link contains city names, and QueryClass is not set, the linker will link the cities, e.g., http://dbpedia.org/resource/Mannheim. If the QueryClass is set to “University” instead, the link will be http://dbpedia.org/resource/University_of_Mannheim. For convenience, the class can be selected from a visualized type hierarchy of the DBpedia ontology:

7http://lookup.dbpedia.org/

8http://dbpedia.org/Ontology



- *DBpedia Lookup API* defines the URL of the lookup service. There are two predefined endpoints (the keyword and prefix lookup[9], as well as the possibility to define a user-specified service URL)
- *Connection timeout msec* defines the timeout interval of the service.
- *Additional string* can be used to define any postfix that is respected when using the EDIT_DISTANCE selection method (see below). For example, if "university" is added, results containing "university" in their name are preferred.
- *Max Hits* defines the maximum number of results used from spotlight (per query).
- *Selection Method* defines how to disambiguate between multiple results. EDIT_DISTANCE, JARO_WINKLER and JACCARD select the result whose name is closest to the original attribute using the respective string similarity function[10]. FIRST_RESULT uses the first result, which is usually the most prominent one.

9See http://lookup.dbpedia.org/ for details.

10See http://en.wikipedia.org/wiki/String_metric for details.

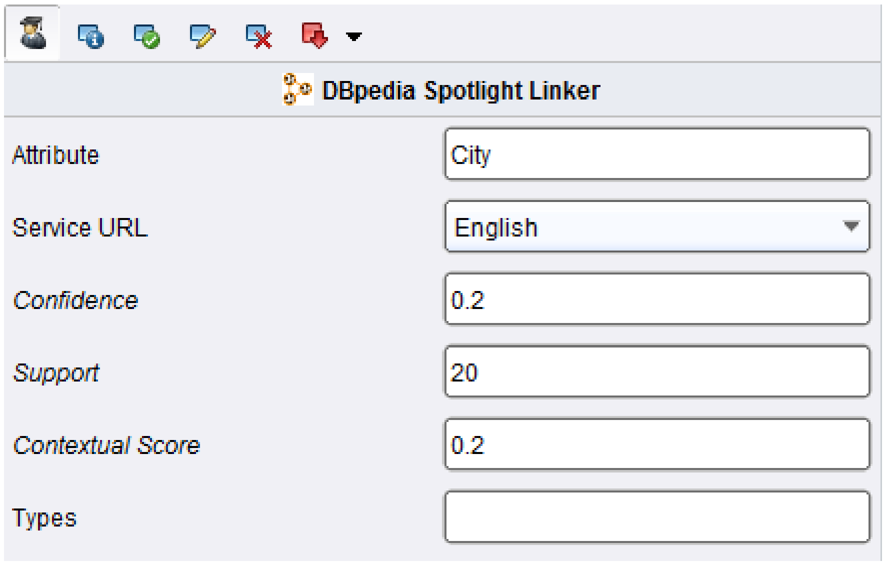
6.4 The DBpedia Spotlight Linker

The DBpedia spotlight linker is a linker which can be used for linking text. It produces a set of attributes, each containing one concept identified. For example, for an attribute "text", the attributes "text_Concept_1", "text_Concept_2", etc., are created.

In subsequent generators, the features generated for those attributes are automatically merged. For example, with a simple type feature generator, there would only be one feature called text_Concept_1_City, no matter if the city is identified as the first, second, third, etc. concept in the text.

The spotlight linker allows for configuring parameters of the service – confidence, support, and ccontextual score. Details can be found in the documentation of DBpedia Spotlight.[11]

The *Service URL* parameter allows for selecting the URL of the service. Some public installed services for common languages are predefined, "Custom" allows for specifying another URL.



DBpedia Spotlight Linker

Attribute	City
Service URL	English
Confidence	0.2
Support	20
Contextual Score	0.2
Types	

The *Types* parameter allows for finding only a list of types (e.g., Persons, Companies, etc.), as specified in the DBpedia ontology (see above) as a comma separated list.

6.5 The SameAs Linker

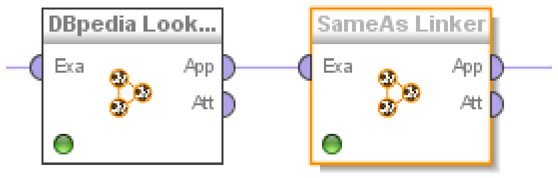
In Linked Open Data, a common way to establish links between datasets is the use of owl:sameAs links, indicating that a resource in dataset 1 is equal to a resource in dataset 2. The owl:sameAs linker follows such links in order to establish links to other data sources.

The typical use case for using the SameAs linker is to create links for a dataset that is difficult (or too slow) to link to with the other approaches, but which provides links to DBpedia, where links can be created,e.g., with the DBpedia Lookup Linker. Consider a setting where the Eurostat Linked Open Data endpoint contains triples such as

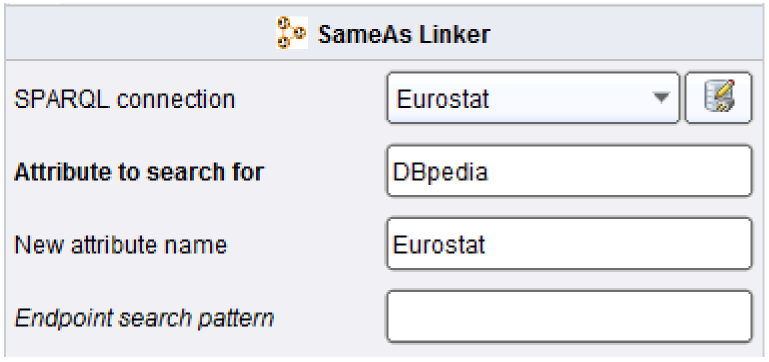
```
eurostat:France owl:sameAs dbpedia:France .
```

These links can be used using the following setup:

11http://spotlight.dbpedia.org/



For configuring the linker, the user has to specify the field containing the original link, the name for the new link, the SPARQL endpoint used to search for owl:sameAs links, and optionally a string pattern for filtering the results:



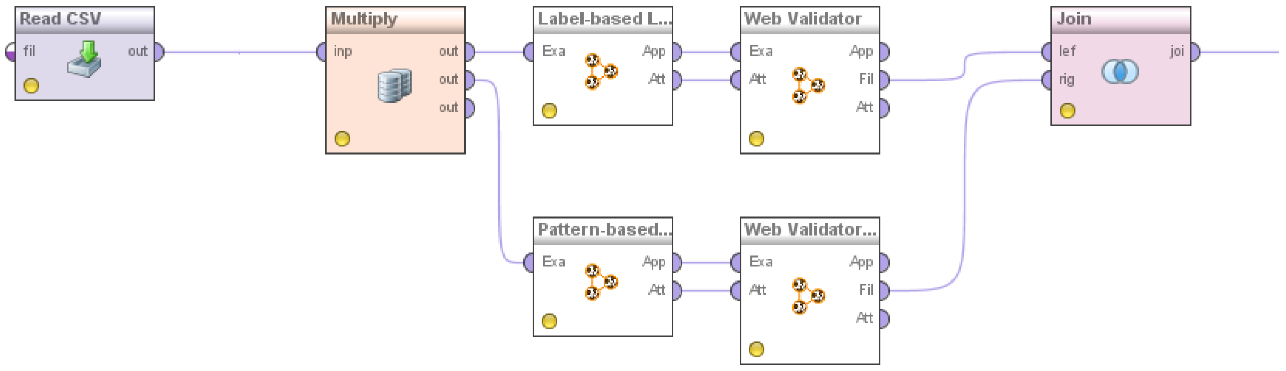
SameAs Linker

SPARQL connection	Eurostat
Attribute to search for	DBpedia
New attribute name	Eurostat
Endpoint search pattern	

With the pipeline and setting shown above, Eurostat URIs are found by first searching for DBpedia URIs, and then exploiting sameAs links in Eurostat.

6.6 Combining Linkers

In some cases, it might be useful to combine linkers – e.g., to add information from multiple sources, or to add information about different entities in a dataset, e.g., both the film as well as its director. In that case, the data can be multiplied with the Multiply operator (in “Process control”), and the output of both linkers is joined with the Join operator (in “Data Transformation”). Note that for the latter step, the original input data needs to define at least one ID column.



7 Using generators

The RapidMiner Linked Open Data extension supports different generators, which perform different knowledge extraction and aggregation steps. A general introduction to generation strategies can be found in (Paulheim and Fürnkranz, 2012).

A generator needs to know which attribute(s) contain the link(s). There are two ways of passing that information to the operator:

- by wiring the Att output of a linker to the corresponding input of the generator (this is the standard way when using generators).
- by entering the attribute name(s) in the Attribute to extend parameter (this is the way to go if the dataset at hand already contains links).

Apart from the attribute generation, most generators also generate a hierarchy of the attributes generated, which can be used for later filtering (see below). Furthermore, the output of the generator is made available in the form of RDF triples.

As a specific configuration option, most generators allow for specifying regular expressions for filtering

Each generator requires either the selection of a SPARQL endpoint *or* the selection of the "Use URI data model", which generates attributes by URI derefencing.

All generators allow for setting regular expression filters. For example, to create features only from properties/classes in the FOAF namespace, the regular expression filter `^http://xmlns.com/foaf/` would be used.

7.1 Value Creation Strategies

Most generators support different value creation strategies:

- *Binary* creates a binary feature being true if a certain characteristic is fulfilled (e.g., a property exists), and false if not
- *Count* creates a numeric feature counting the occurences of a certain characteristc (e.g., the number of times a property occurs with an instance)
- *Relative count* creates a numeric feature, counting the occurences of a certain characteristics in relation to the total number of possible occurences (e.g., the percentage of properties of an instance being of a certain kind)
- *TF-IDF* creates a similar feature as relative count, but weighs the percentage with the overall occurrence of a characteristic in the dataset. Thereby, the influence of generic, most often useless characteristics (such as an instance being of type `owl:Thing`) can be reduced.

In general, it is hard to tell in advance which characteristic works best for a given use case. A detailed comparison of the strategies is given in (Paulheim and Ristoski, 2014a).

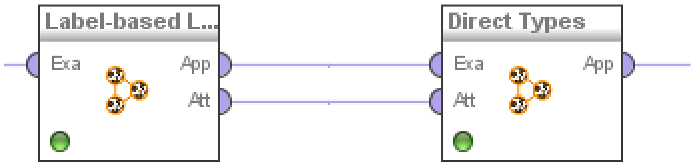
7.2 Direct Types Generator

The Direct Types Generator extracts all statements with predicate `rdf:type` and creates a boolean attribute for each type. For example, from a statement like

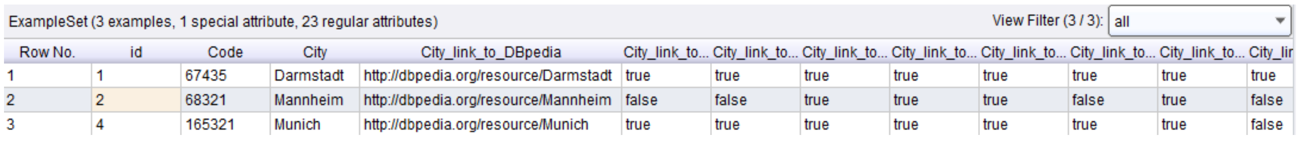
```
<http://dbpedia.org/resource/Mannheim> rdf:type
<http://dbpedia.org/ontology/City> .
```

an attribute with name City_link_to_DBpedia would be created, and the value for the instance linked to <http://dbpedia.org/resource/Mannheim> would be set to true.

The process using the Simple Types Generator looks as follows:



Note that both outputs of the linker – App and Att – need to wired to the respective input ports of the generator. The output looks as follows:


ExampleSet (3 examples, 1 special attribute, 23 regular attributes) View Filter (3 / 3): all

Row No.	id	Code	City	City_link_to_DBpedia	City_link_to...	City_link_to...	City_link_to...	City_link_to...	City_link_to...	City_link_to...	City_link_to...	City_lir
1	1	67435	Darmstadt	http://dbpedia.org/resource/Darmstadt	true	true	true	true	true	true	true	true
2	2	68321	Mannheim	http://dbpedia.org/resource/Mannheim	false	false	true	true	true	false	true	false
3	4	165321	Munich	http://dbpedia.org/resource/Munich	true	true	true	true	true	true	true	false

7.3 Data Properties Generator

The Data Properties Generator creates an attribute for each literal value that the linked instances has. For example, from the property

```
<http://dbpedia.org/resource/Mannheim>
<http://dbpedia.org/ontology/population> "123873"^^xsd:integer .
```

the generator would create an attribute City_link_to_DBpedia_data_http://dbpedia.org/ontology/population and set the value for the instance linked to Mannheim to 123873. Some basic guessing of data types is performed, e.g., numeric values are marked as such, and numeric literals without an explicit type are parsed into numbers, if possible.

7.4 Unqualified and Qualified Relation Generators

The RapidMiner Linked Open Data extension supports generators exploiting relations to other resources. The *unqualified* relations generators create attributes from the existence of relations, while the *qualified* relations generators also take the types of the related resources into account. Both generators exist in two variants, one generating boolean, one generating numeric features.

Consider the following triples:

```
<http://dbpedia.org/resource/Berlin>
<http://dbpedia.org/ontology/capitalOf>
<http://dbpedia.org/resource/Germany> .

<http://dbpedia.org/resource/Germany> rdf:type
<http://dbpedia.org/ontology/Country> .
```

The Relation Boolean generator would create an attribute City_link_to_DBpedia_in_boolean_http://dbpedia.org/ontology/capitalOf and set its value to *true* for an instance linked to <http://dbpedia.org/resource/Berlin> . The Relation Numeric generator would create an attribute City_link_to_DBpedia_in_numeric_http://dbpedia.org/ontology/capitalOf and set its value to *1* for an instance linked to <http://dbpedia.org/resource/Berlin>.

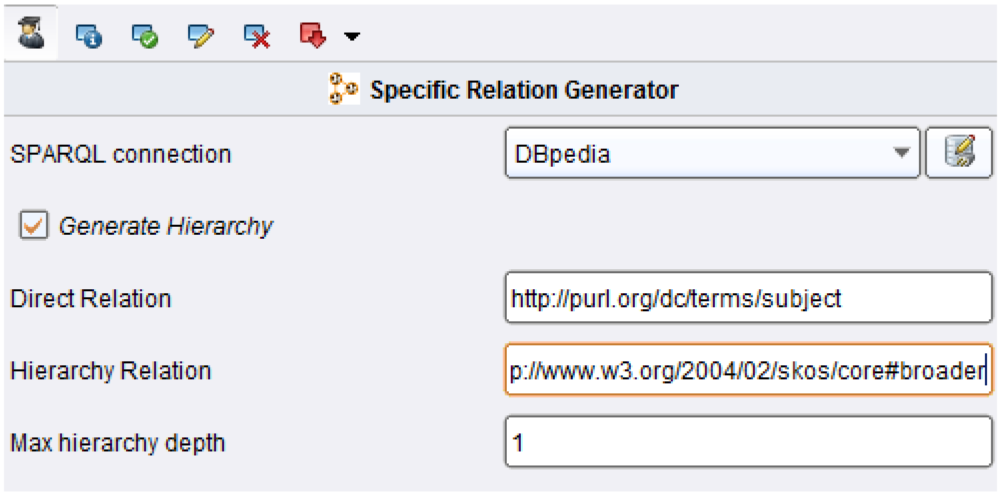
The Qualified Relation Boolean generator would create an attribute City_link_to_DBpedia_in_boolean_http://dbpedia.org/ontology/capitalOf_type_http://dbpedia.org/ontology/Country and set its value to *true* for an instance linked to <http://dbpedia.org/resource/Berlin> . The Qualified Relation Numeric generator would create an attribute City_link_to_DBpedia_in_numeric_http://dbpedia.org/ontology/capitalOf_type_http://dbpedia.org/ontology/Country and set its value to *1* for an instance linked to <http://dbpedia.org/resource/Berlin>.

Note that in especially the qualified relation generators, while very powerful, may create a very large number of attributes, which can cause problems for subsequent processing steps.

7.5 The Specific Relation Generator

In order to pursue only a specific predicate, the specific relation linker can be used. Optionally, a predicate to use for generating the hierarchy can be specified.

For example, the configuration below extracts all categories from DBpedia, and uses the skos:broader predicate (which is used for category hierarchies in DBpedia) for generating the hierarchy:

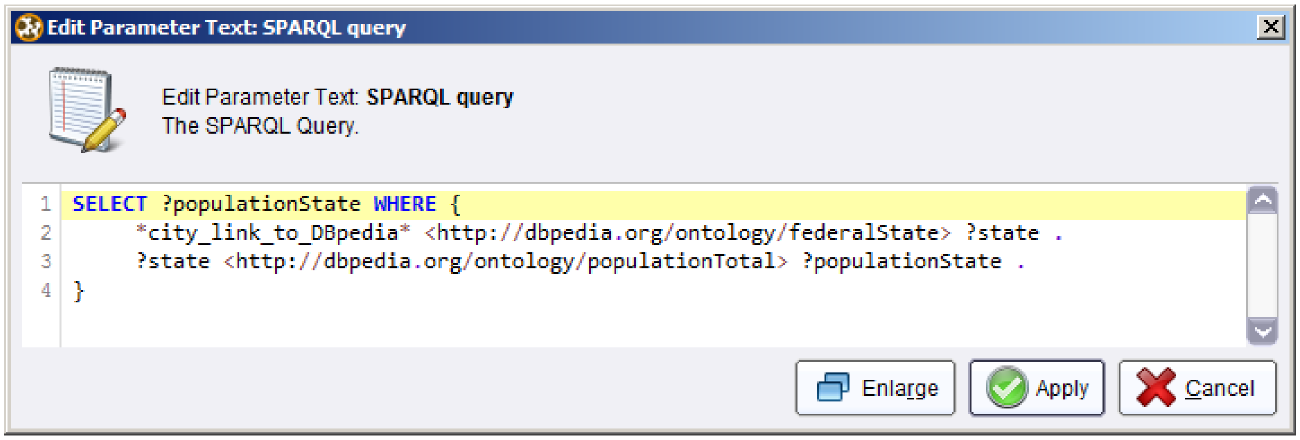


Specific Relation Generator

SPARQL connection	DBpedia
✓ Generate Hierarchy	
Direct Relation	http://purl.org/dc/terms/subject
Hierarchy Relation	p://www.w3.org/2004/02/skos/core#broader
Max hierarchy depth	1

7.6 The Custom SPARQL Generator

In cases where you know the dataset you are using and want to add specific data not covered by any of the default generators, you can also define your own SPARQL statements. For example, for a dataset of cities, you might want to add the population of the state that the city is in. This can be done with the custom SPARQL generator, which allows to create a custom statement:



Edit Parameter Text: SPARQL query

Edit Parameter Text: **SPARQL query**
The SPARQL Query.

```
SELECT ?populationState WHERE {
     *city_link_to_DBpedia* <http://dbpedia.org/ontology/federalState> ?state .
     ?state <http://dbpedia.org/ontology/populationTotal> ?populationState .
}
```

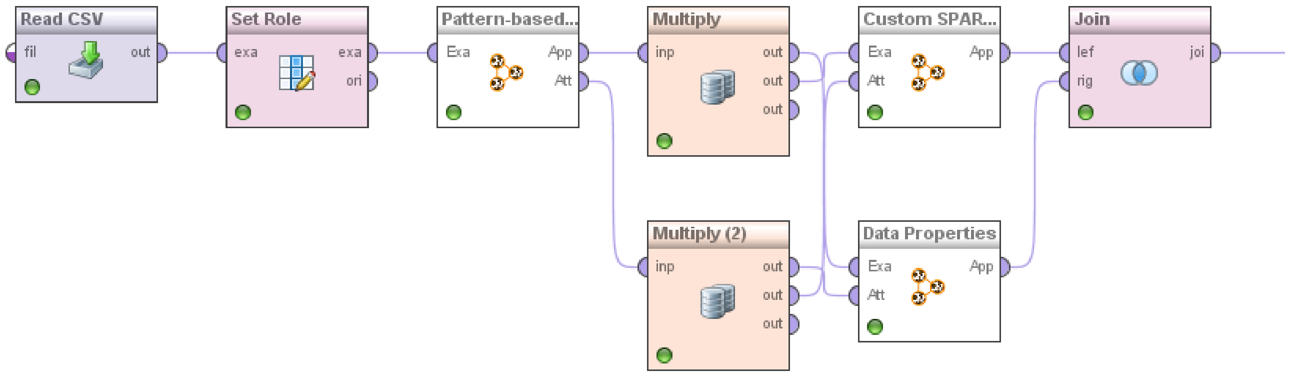
Within the statement, you can use links generated by linkers as placeholders, enclosed in asterisks. In the above example, a linker has generated the link attribute city_link_to_DBpedia, which is used in the query. The generator then generates a new attribute called "populationState" (using the variable name for naming the attribute) and adds it to the dataset.

The reserved term `*ATT_INPUT_PORT*` can be used in the query. It is filled with the attributes passed at the att input port.

As for the SPARQL data importer (see above), custom prefixes can be used, which are resolved using the prefix.cc service.

7.7 Combining Generators

For combining generators on the same link, e.g., extracting both data properties and types, the following process setup has to be used:



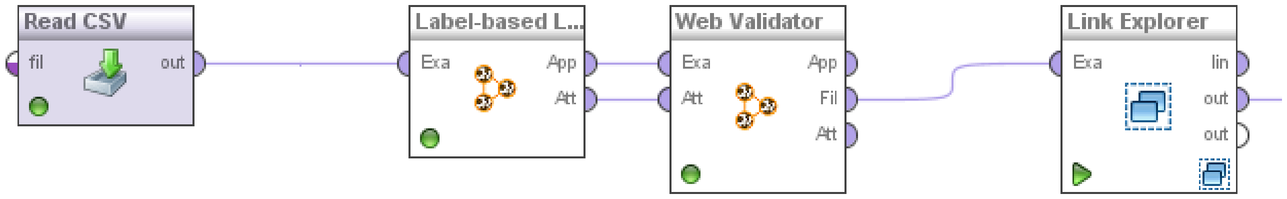
Two Multiply operators are used, one for multiplying the data, one for multiplying the attribute information. Like for combining linkers, the input data needs to define at least one ID attribute to allow the final Join. The Set Role operator can be used for defining IDs.

Note that you can also combine generators with data from different data sources. To that end, you have to run a linker and one or more generators for each data source.

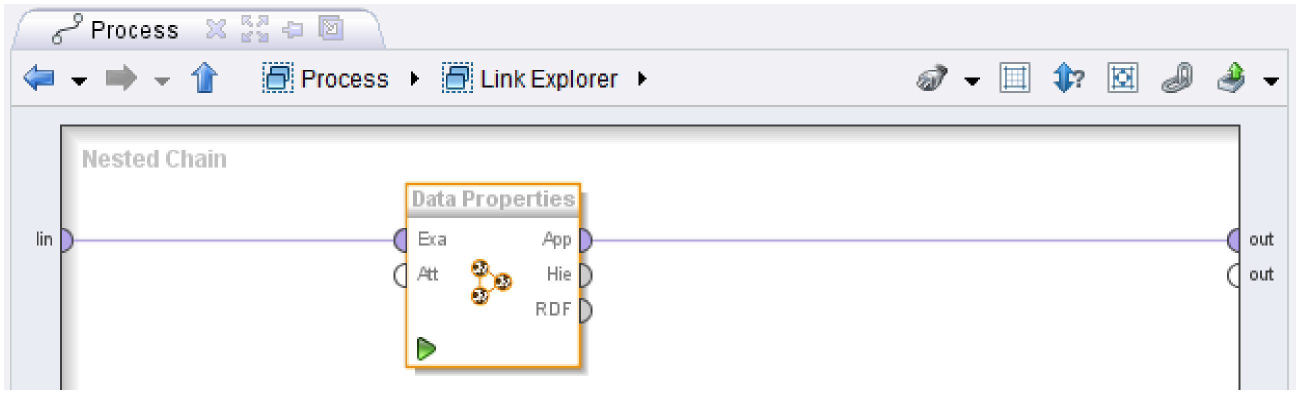
8 Exploring the Linked Open Data Cloud

One key characteristic of Linked Open Data is that agents can navigate the cloud of interlinked datasets by following links, discovering new information on the fly. While in all example so far, we have looked at datasets that are known in advance, the LOD extension also supports an explorative way of dealing with data.

Starting from a link found by a linker (see above), the extension is capable of following certain links to a number of user specified hops, and collect all information found with one or more generators. An example process looks as follows:

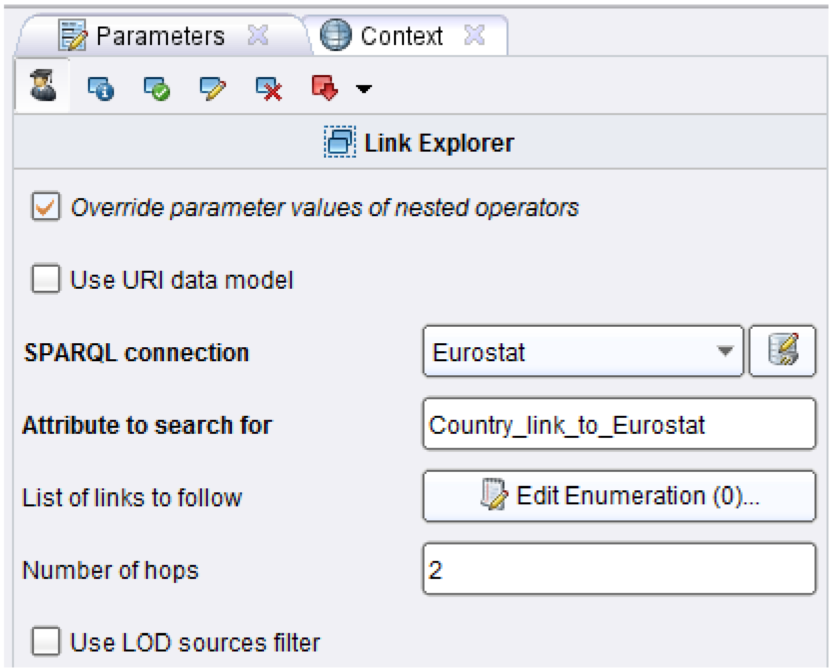


The link explorer has two outputs: "lin" delivers all links discovered by the explorer, while "out" delivers the output collected from the generator(s) used by the explorer. That generator(s) is defined in a nested process inside the operator, e.g., collecting all data properties found at each discovered link:



Note that for the nested generator, "Use URI data model" has to be selected instead of a SPARQL endpoint, since the SPARQL endpoint for a discovered resource is usually not known (see Paulheim and Hertling, 2013, for a discussion of the problem).

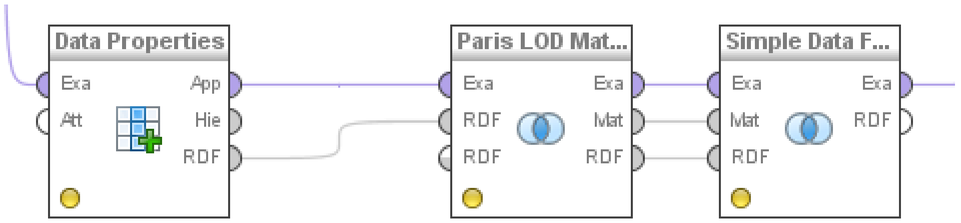
The link explorer allows for different configuration options:


Parameters
Context
Link Explorer
☑ *Override parameter values of nested operators*
☐ Use URI data model
SPARQL connection: Eurostat
Attribute to search for: Country_link_to_Eurostat
List of links to follow: Edit Enumeration (0)...
Number of hops: 2
☐ Use LOD sources filter

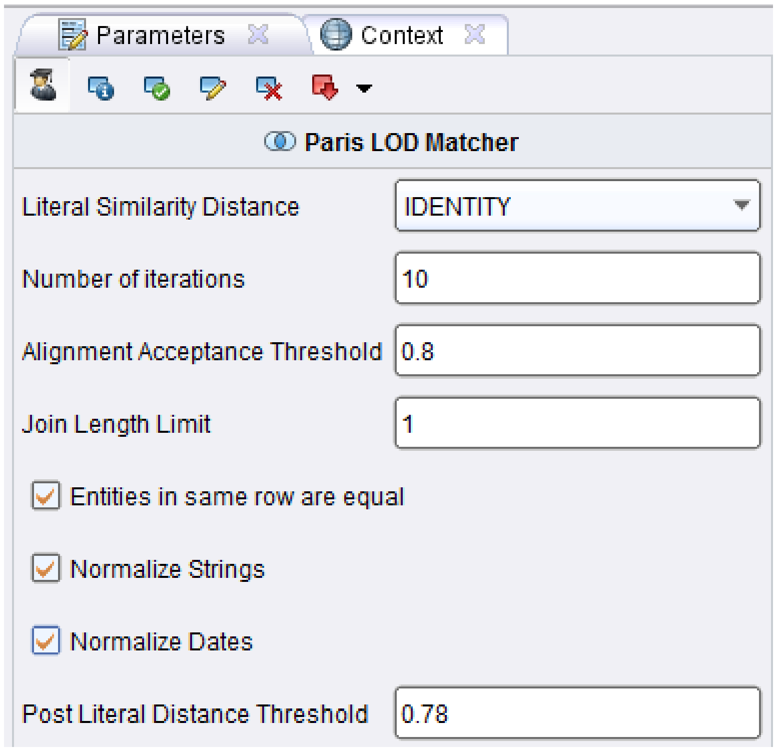
- SPARQL connection/Use URI data model are only relevant for the initial link to find the links from there. Once those are followed, only URI data models may be used (see above)
- List of links to follow: by default, the link explorer follows all owl:sameAs and rdfs:seeAlso links, but a custom list of links may also be used
- Number of hops: the depth of exploration of the LOD cloud
- Use LOD sources filter: a set of datasets may be defined here to restrict the exploration.

9 Schema Matching and Data Fusion

If data from different sources is used, it may be beneficial to match and fuse that data. The extension provides two operators: a Linked Open Data matching operator using PARIS (Suchanek et al., 2011), and a Data Fusion operator. They have to be wired as follows:

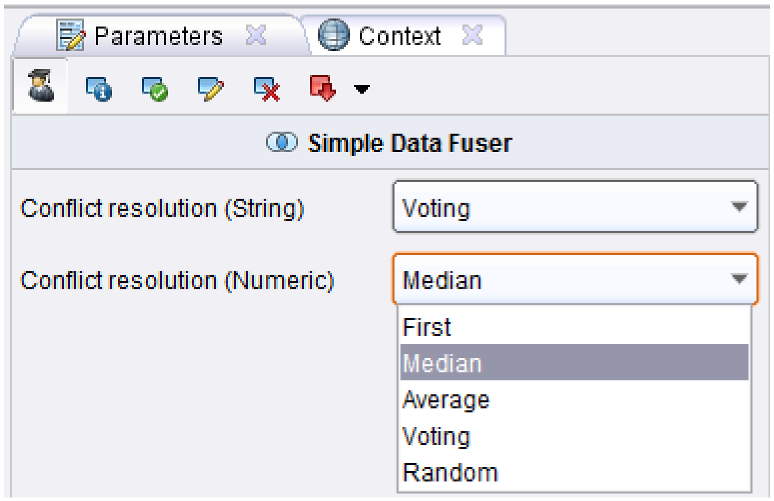


The matcher takes as input the instances and the RDF created by one or more generators, and outputs a list of matches. Different parameters can be set for the PARIS matcher; see Suchanek et al. (2011) for details:



Literal Similarity Distance	IDENTITY
Number of iterations	10
Alignment Acceptance Threshold	0.8
Join Length Limit	1
✓ Entities in same row are equal	
✓ Normalize Strings	
✓ Normalize Dates	
Post Literal Distance Threshold	0.78

The mappings created by the matcher can be used in the Simple Data Fusion operator using different fusion strategies, to be selected separately for nominal/string and numerical values:



Simple Data Fuser

Conflict resolution (String): Voting

Conflict resolution (Numeric): Median

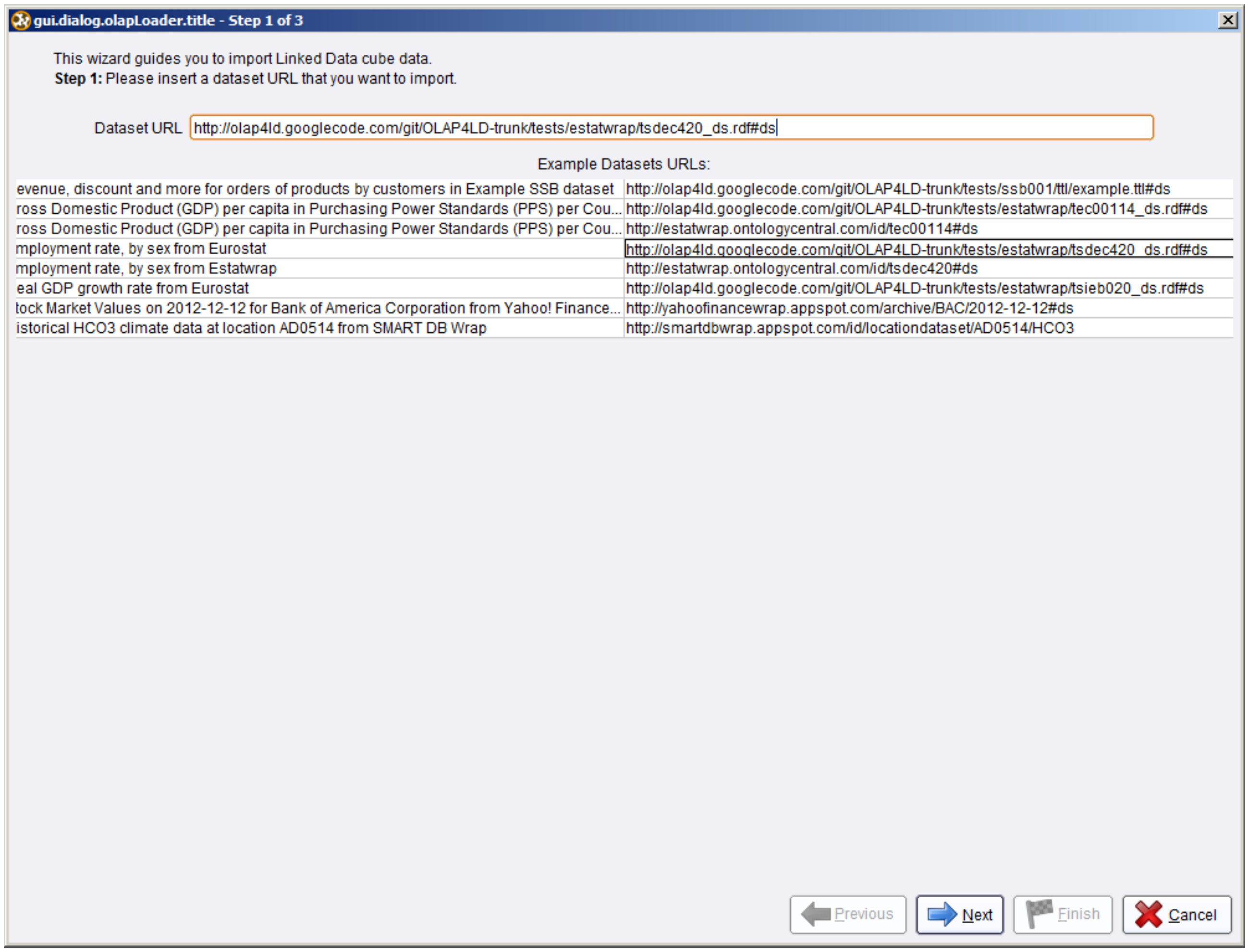
10 Accessing RDF Data Cubes

When publishing multi-dimensional datasets as Linked Open Data (e.g., statistical observations of different phenomena across several years), the RDF data cube vocabulary[12] is often used. The LOD extension provides a data importer for such data cubes, which is based on the OLAP4LD library (Kämpgen and Harth, 2014).

For example, the dataset

http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/estatwrap/tsdec420_ds.rdf#ds

contains unemployment rates for different states by age and sex. It can be accessed by using the "Data Cube Importer" and starting the corresponding wizard:


gui.dialog.olapLoader.title - Step 1 of 3

This wizard guides you to import Linked Data cube data.
Step 1: Please insert a dataset URL that you want to import.

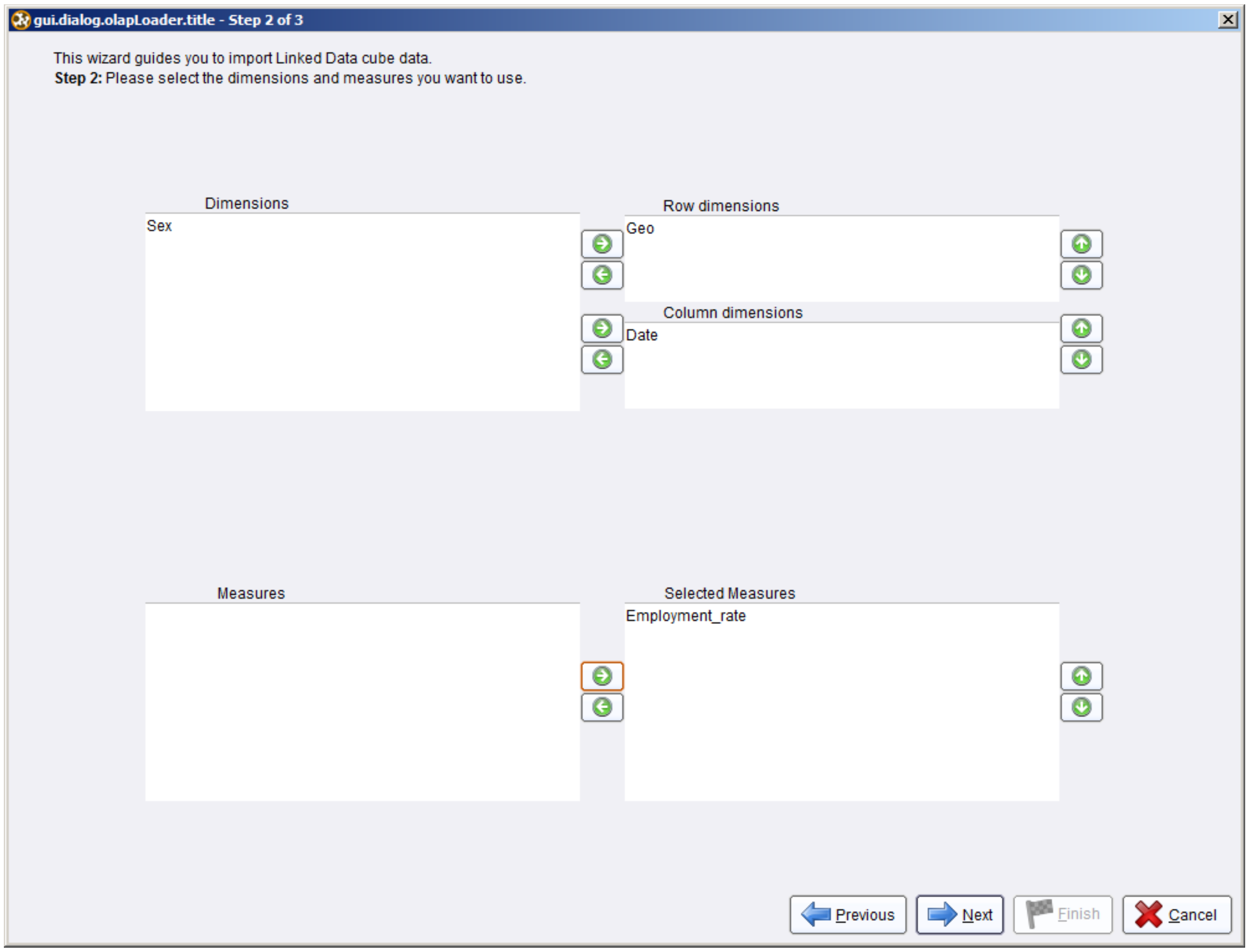
Dataset URL: http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/estatwrap/tsdec420_ds.rdf#ds

Example Datasets URLs:

evenue, discount and more for orders of products by customers in Example SSB dataset	http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/ssb001/ttl/example.ttl#ds
ross Domestic Product (GDP) per capita in Purchasing Power Standards (PPS) per Cou...	http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/estatwrap/tec00114_ds.rdf#ds
ross Domestic Product (GDP) per capita in Purchasing Power Standards (PPS) per Cou...	http://estatwrap.ontologycentral.com/id/tec00114#ds
mployment rate, by sex from Eurostat	http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/estatwrap/tsdec420_ds.rdf#ds
mployment rate, by sex from Estatwrap	http://estatwrap.ontologycentral.com/id/tsdec420#ds
eal GDP growth rate from Eurostat	http://olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/estatwrap/tsieb020_ds.rdf#ds
tock Market Values on 2012-12-12 for Bank of America Corporation from Yahoo! Finance...	http://yahoofinancewrap.appspot.com/archive/BAC/2012-12-12#ds
istorical HCO3 climate data at location AD0514 from SMART DB Wrap	http://smartdbwrap.appspot.com/id/locationdataset/AD0514/HCO3

Upon clicking "Next", the dataset found at the URI is analyzed, and the dimensions are extracted. In the next dialogue, the dimensions for the rows and columns, as well as the variable under inspection, are selected:

12http://www.w3.org/TR/vocab-data-cube/



gui.dialog.olapLoader.title - Step 2 of 3

This wizard guides you to import Linked Data cube data.
Step 2: Please select the dimensions and measures you want to use.

Dimensions: Sex

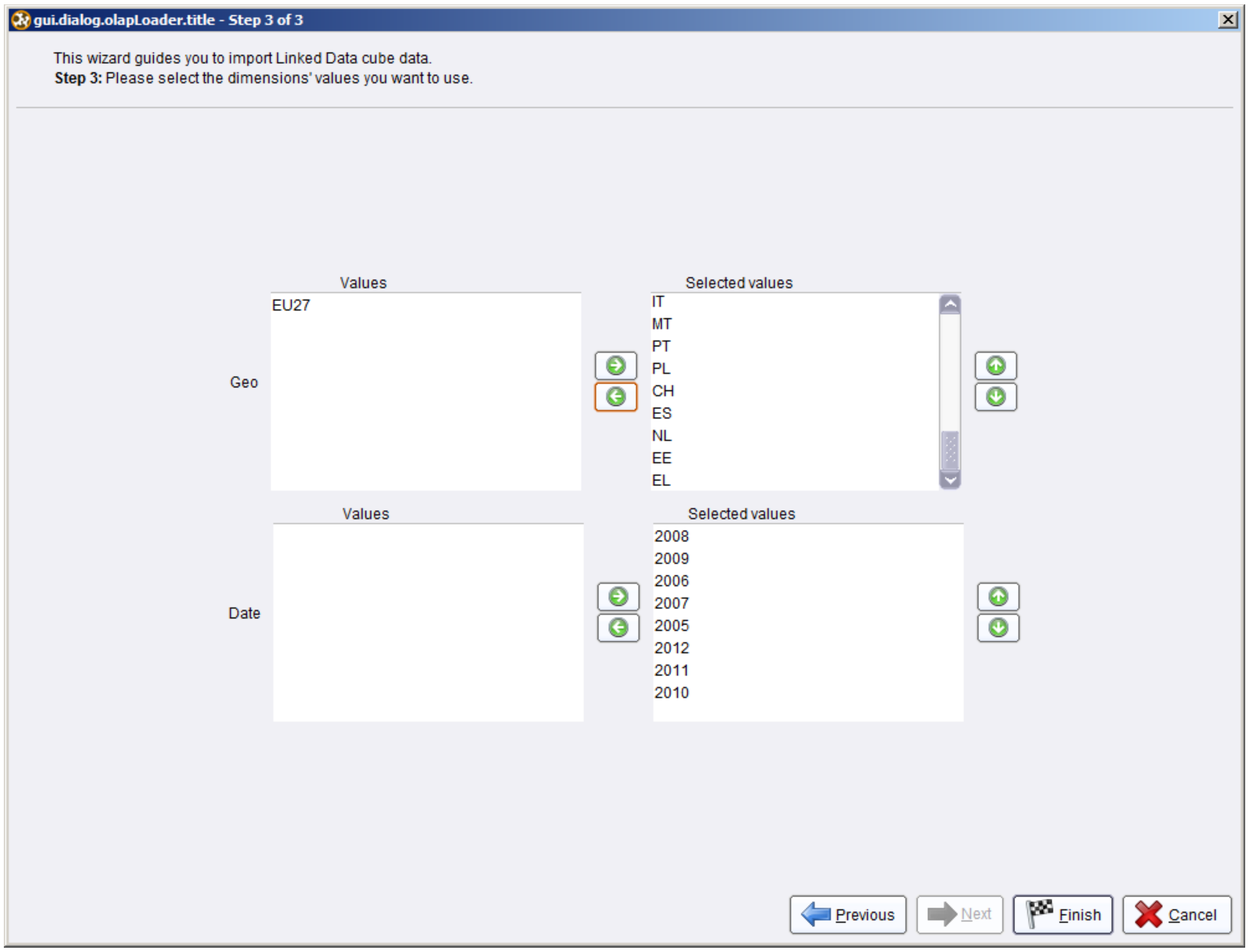
Row dimensions: Geo

Column dimensions: Date

Measures:

Selected Measures: Employment_rate

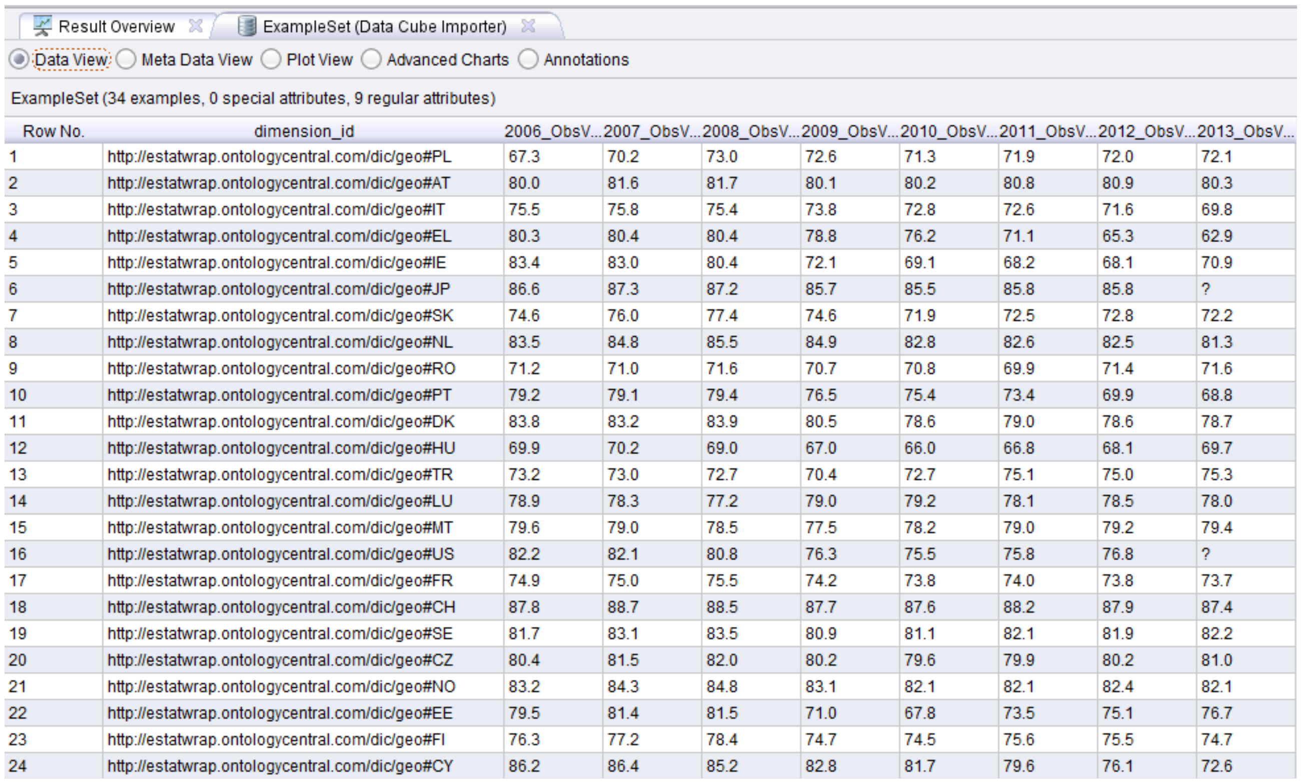
Finally, the instances of each dimension are selected:


gui.dialog.olapLoader.title - Step 3 of 3

This wizard guides you to import Linked Data cube data.
Step 3: Please select the dimensions' values you want to use.

	Values	Selected values
Geo	EU27	IT, MT, PT, PL, CH, ES, NL, EE, EL
Date		2008, 2009, 2006, 2007, 2005, 2012, 2011, 2010

When running the process, a data table with the specified rows and columns is created, which can then be further analyzed:



ExampleSet (34 examples, 0 special attributes, 9 regular attributes)

Row No.	dimension_id	2006_ObsV...	2007_ObsV...	2008_ObsV...	2009_ObsV...	2010_ObsV...	2011_ObsV...	2012_ObsV...	2013_ObsV...
1	http://estatwrap.ontologycentral.com/dic/geo#PL	67.3	70.2	73.0	72.6	71.3	71.9	72.0	72.1
2	http://estatwrap.ontologycentral.com/dic/geo#AT	80.0	81.6	81.7	80.1	80.2	80.8	80.9	80.3
3	http://estatwrap.ontologycentral.com/dic/geo#IT	75.5	75.8	75.4	73.8	72.8	72.6	71.6	69.8
4	http://estatwrap.ontologycentral.com/dic/geo#EL	80.3	80.4	80.4	78.8	76.2	71.1	65.3	62.9
5	http://estatwrap.ontologycentral.com/dic/geo#IE	83.4	83.0	80.4	72.1	69.1	68.2	68.1	70.9
6	http://estatwrap.ontologycentral.com/dic/geo#JP	86.6	87.3	87.2	85.7	85.5	85.8	85.8	?
7	http://estatwrap.ontologycentral.com/dic/geo#SK	74.6	76.0	77.4	74.6	71.9	72.5	72.8	72.2
8	http://estatwrap.ontologycentral.com/dic/geo#NL	83.5	84.8	85.5	84.9	82.8	82.6	82.5	81.3
9	http://estatwrap.ontologycentral.com/dic/geo#RO	71.2	71.0	71.6	70.7	70.8	69.9	71.4	71.6
10	http://estatwrap.ontologycentral.com/dic/geo#PT	79.2	79.1	79.4	76.5	75.4	73.4	69.9	68.8
11	http://estatwrap.ontologycentral.com/dic/geo#DK	83.8	83.2	83.9	80.5	78.6	79.0	78.6	78.7
12	http://estatwrap.ontologycentral.com/dic/geo#HU	69.9	70.2	69.0	67.0	66.0	66.8	68.1	69.7
13	http://estatwrap.ontologycentral.com/dic/geo#TR	73.2	73.0	72.7	70.4	72.7	75.1	75.0	75.3
14	http://estatwrap.ontologycentral.com/dic/geo#LU	78.9	78.3	77.2	79.0	79.2	78.1	78.5	78.0
15	http://estatwrap.ontologycentral.com/dic/geo#MT	79.6	79.0	78.5	77.5	78.2	79.0	79.2	79.4
16	http://estatwrap.ontologycentral.com/dic/geo#US	82.2	82.1	80.8	76.3	75.5	75.8	76.8	?
17	http://estatwrap.ontologycentral.com/dic/geo#FR	74.9	75.0	75.5	74.2	73.8	74.0	73.8	73.7
18	http://estatwrap.ontologycentral.com/dic/geo#CH	87.8	88.7	88.5	87.7	87.6	88.2	87.9	87.4
19	http://estatwrap.ontologycentral.com/dic/geo#SE	81.7	83.1	83.5	80.9	81.1	82.1	81.9	82.2
20	http://estatwrap.ontologycentral.com/dic/geo#CZ	80.4	81.5	82.0	80.2	79.6	79.9	80.2	81.0
21	http://estatwrap.ontologycentral.com/dic/geo#NO	83.2	84.3	84.8	83.1	82.1	82.1	82.4	82.1
22	http://estatwrap.ontologycentral.com/dic/geo#EE	79.5	81.4	81.5	71.0	67.8	73.5	75.1	76.7
23	http://estatwrap.ontologycentral.com/dic/geo#FI	76.3	77.2	78.4	74.7	74.5	75.6	75.5	74.7
24	http://estatwrap.ontologycentral.com/dic/geo#CY	86.2	86.4	85.2	82.8	81.7	79.6	76.1	72.6

11 Filtering Attributes

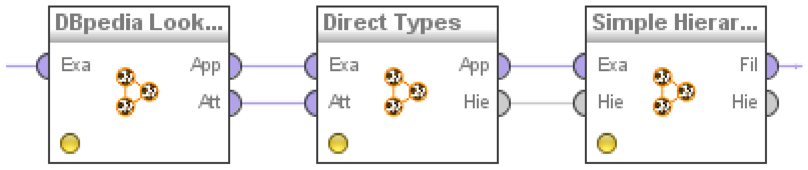
The RapidMiner LOD extension may create a lot of attributes. Thus, it is advisable to filter the attributes. Besides standard attribute selection operators that exist in RapidMiner (e.g. Remove Correlated Attributes) or the Feature Selection Extension[13], the extension provides means to exploit the semantics of Linked Open Data sources used to filter attributes.

Some generators, such as the direct types generator, also create a hierarchy of attributes. As for direct types, rdfs:subClassOf links are exploited to build the hierarchies of attributes generated.

The Simple Hierarchy Filter operator uses that hierarchy to filter attributes. Different algorithms are available:

- *SHSEL* eliminates redundant attributes along hierarchy paths, while *pruneSHSEL* further reduces the feature set by selecting only the most relevant attributes of the reduced hierarchy (Ristoski and Paulheim, 2014). For both algorithms, the underlying relevance measure can be correlation (C) or information gain (IC).
- *TSEL* (Jeong and Myaeng, 2013) selects the most valuable attributes from each branch in the hiearchy, based on information gain or lift.
- *Hill Climbing* is a bottom up approach which uses the purity of nearest neighbors of all instances in a dataset to assign scores to attributes (Wang et al., 2003).
- *Greedy top down* selects attributes that have a high information gain ratio, while pruning those attributes that are adjacent to the selected ones in the hierarchy (Lu et al., 2013)

All algorithms aim at removing redundant attributes, and avoiding to learn overfitting models. The usage is shown below.



A detailed comparison and evaluation on the filtering algorithms can be found in (Ristoski and Paulheim, 2014b).

13http://rm-featselext.sourceforge.net/

12 Migration from Earlier Versions of the Extension

Migrating from earlier versions of the extension may cause problems in the SPARQL configurations dialog. If you experience such problems (which manifest themselves by repeated questions whether you want to store a connection), locate and delete the file `configurable-sparqlconfig.xml` on your harddisk, and restart RapidMiner.

13 References

1. Christian Bizer, Tom Heath, and Tim Berners-Lee: Linked Data-the Story so Far. International Journal on Semantic Web and Information Systems 5(3), 1-22 (2009)
2. Daniel Hienert, Dennis Wegener and Heiko Paulheim. Automatic Classification and Relationship Extraction for Multi-Lingual and Multi-Granular Events from Wikipedia. In: Proceedings of the Workhop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2012); 1-10. RWTH, Aachen, 2012.
3. Frederik Janssen, Faraz Fallahi, Jan Nößner and Heiko Paulheim. Towards Rule Learning Approaches to Instance-based Ontology Matching. In: Proceedings of the First International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data; 13-18. RWTH, Aachen, 2012.
4. Yoonjae Jeong and Sung-Hyon Myaeng. Feature selection using a semantic hierarchy for event recognition and type classiffication. In: International Joint Conference on Natural Language Processing, 2013.
5. Benedikt Kämpgen, Andreas Harth OLAP4LD - A Framework for Building Analysis Applications over Governmental Statistics. ESWC 2014 Posters & Demos.
6. Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web Journal, 2013.
7. Sisi Lu, Ye Ye, Rich Tsui, Howard Su, Ruhsary Rexit, Sahawut Wesaratchakit, Xiaochu Liu, and Rebecca Hwa. Domain ontology-based feature reduction for high dimensional drug data and its application to 30-day heart failure readmission prediction. In International Conference Conference on Collaborative Computing (Collaboratecom), 2013.
8. Heiko Paulheim. Generating Possible Interpretations for Statistics from Linked Open Data. In: 9th Extended Semantic Web Conference, ESWC 2012; 560-574. Springer, Berlin [u.a.], 2012.
9. Heiko Paulheim. Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection. In: Third International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2014).
10. Heiko Paulheim and Johannes Fürnkranz. Unsupervised Generation of Data Mining Features from Linked Open Data. In International Conference on Web Intelligence, Mining, and Semantics (WIMS'12), 2012.
11. Heiko Paulheim and Sven Hertling. Discoverability of SPARQL Endpoints in Linked Open Data. In: ISWC 2013 Posters & Demonstrations Track, 2013.
12. Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, Christian Bizer : Data Mining with Background Knowledge from the Web. In: RapidMiner World, 2014.
13. Petar Ristoski and Heiko Paulheim. Analyzing Statistics with Background Knowledge from Linked Open Data. In: First International Workshop on Semantic Statistics (SemStats 2013).
14. Petar Ristoski and Heiko Paulheim: A Comparison of Propositionalization Strategies for Creating Features from Linked Open Data. In: Workshop on Linked Data for Knowledge Discovery, 2014a
15. Petar Ristoski and Heiko Paulheim: Feature selection in hierarchical feature spaces. In: Discovery Science, 2014b.

16. Max Schmachtenberg, Christian Bizer, and Heiko Paulheim: Adoption of the Linked Data Best Practices in Different Topical Domains. In: International Semantic Web Conference, 2014

17. Axel Schulz, Aristotelis Hadjakos, Heiko Paulheim, Johannes Nachtwey, Max Mühlhäuser. A Multi-Indicator Approach for Geolocalization of Tweets. In: International AAAI Conference on Weblogs and Social Media (ICWSM), 2013a.

18. Axel Schulz, Petar Ristoski and Heiko Paulheim. I See a Car Crash: Real-time Detection of Small Scale Incidents in Microblogs. In: Workshop on Social Media and Linked Data for Emergency Response (SMILE), 2013b.

19. Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. PVLDB, 5(3):157–168, 2011.

20. Bill B. Wang, R. I. Bob Mckay, Hussein A. Abbass, and Michael Barlow. A comparative study for domain ontology guided feature extraction. In Australasian Computer Science Conference, 2003.

14 Release Notes

Version 1.5.0, released on September 19th, 2014

- Schema Matching
- Improved Data Cube Importer
- Bug fixes

Version 1.4.0, released on August 11th, 2014

- Link Explorer
- Data Cube Importer
- RDF Output for generators
- Regex filters in generators
- Enhanced selection options for datatype property generator
- Easier handling of URI derefencing
- Web Validator also works with URI derefencing
- RapidMiner proxy settings are used
- Advanced caching mechanism
- Bug fixes

Version 1.3.3, released on May 15th, 2014

- Additional configuration options for DBpedia Lookup and DBpedia Spotlight
- Performance optimizations
- Bug fixes

Version 1.3.2, released on March 14th, 2014

- The SPARQL importer and generator now support common prefixes (e.g., rdf, owl, foaf, etc.) through prefix.cc
- Bugfix in SPARQL importer

Version 1.3.1, released on March 7th, 2014

- Support for feature extraction by resolving URIs
- Entity class detection during linking
- Visual selection of the query class for the DBpedia Lookup linker
- Performance improvements through multi threading
- Bugfixes

Version 1.2.116, released on November 29th, 2013

- Specific relation generator
- Sameas linker
- Simple hierarchy filter
- Support for pre-defined links
- Performance improvements
- Bugfixes

Version 1.1.024, released on October 15th, 2013

- Support for local RDF files, including reasoning
- DBpedia Lookup linker
- DBpedia Spotlight linker
- Support for binary features in Custom SPARQL generator
- Bugfixes

Version 1.0.071, released on September 13th, 2013.