SPARQLED

— ASSISTED SPARQL EDITOR THROUGH RDF GRAPH ANALYTICS

SparQLed is an interactive SPARQL editor that helps users in formulating complex SPARQL queries across multiple heterogeneous data sources. Whether or not the structure and vocabulary of data sources are known, users can easily formulate queries through various kinds of context-aware element recommendations.

This document presents the steps to setup the editor:

1. Compute the Data Graph Summary;
2. Configure the webapp;
3. Launch the SparQLed webapp.

The Data Graph Summary

SparQLed makes use of the entire query structure and of the Data Graph Summary [1] (DGS) in order to present to the user possible query elements.

The DGS has to be calculated beforehand and contains a statistical description of the underlying data. The DGS is depicted in Figure 1, where the bottom layer, i.e., the original RDF data, is condensed into the graph represented on the middle layer. For example, the resources of type Person are represented with the single resource labelled Person, where a cardinality indicates how many resources were aggregated.

The DGS is represented again in RDF using the *Dataset Analytics Vocabulary*[2] ontology. Usually considerably smaller than the original RDF Graph, it is accessed by the Sparqled via SPARQL. For querying and exploring the DGS outside of SparQLed see the Appendix.

[1] Stephane Campinas, Thomas E. Perry, Diego Ceccarelli, Renaud Delbru, and Giovanni Tummarello. Introducing RDF Graph Summary With Application to Assisted SPARQL Formulation. *WebS, DEXA workshop*, 2012

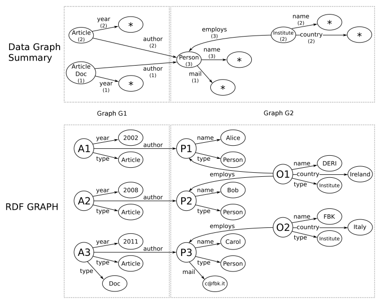


Figure 1: Layers of the Data Graph Summary Model

[2] `http://vocab.sindice.net/analytics`

0. Getting Started

The following needs to be installed:

- Tomcat6[3] or Tomcat7;
- Maven3;
- Java 1.6.0_*;
- a SPARQL endpoint with SPARQL1.1 support.

[3] on Ubuntu, sudo apt-get install tomcat6

The following command creates the file *sparql-summary-assembly.jar* in the directory *target/*. It is used to compute the Data Graph Summary and other operations on a SPARQL endpoint:

```
$ mvn --projects sparql-summary package assembly:assembly
--also-make
```

A detailed explanation of the `sparql-summary-assembly.jar` command line interface is found in the README file, in the `sparql-summary` module.

1. Computing the Data Graph Summary

The DGS implementation bundled with this documentation is based on SPARQL, and computes the summary over a single dataset. A high performance, hadoop-based, DGS implementation is also available, and allows to compute a summary across datasets. Therefore, links between datasets can also be recommended. However, this version is not shipped with this package. Please inquire[4].

[4] http://groups.google.com/group/sindice-dev

The following command will create the DGS over the given dataset which lies in a SPARQL endpoint located at `http://url/to/sparql/endpoint/`. The output is a gzip compressed NQuads file `/path/to/<data>-summary.nt.gz`. It is strongly advised to give enough memory to prevent OOME, e.g., `-Xmx2048m`.

```
 $ java -cp target/sparql-summary-assembly.jar
org.sindice.summary.Pipeline --type HTTP --repository
http://url/to/sparql/endpoint/ --outputfile
/path/to/<data>-summary.nt.gz
```

You can constrain the processing to a single named graph using the option `--domain`.

A SPARQL endpoint must then be loaded with the computed DGS NQuads, under the named graph `http://sindice.com/analytics`.

Remarks: using Virtuoso

If using a *Virtuoso* SPARQL endpoint, please write --type VIRTUOSO instead of --type HTTP in the previous command. Since the DGS computation is computationally demanding, you may want to increase the timeout parameter of Virtuoso by changing the following parameter in the *virtuoso.ini* file:

```
MaxQueryExecutionTime=X
```

where the X value is the time in seconds. The Virtuoso server needs then to be restarted after changing that file.

2. Configuring the Webapp

In this section and the next, we use by default tomcat6. If you are to use tomcat7, replace all occurrences of tomcat6 by tomcat7.

In order to properly configure the application, we need to stop tomcat:

```
$ sudo service tomcat6 stop
```

Edit Tomcat Properties

The application needs to set a pre-defined folder, where it can read configuration settings and also writes log output into a file. That folder is set using the Tomcat property `sindice.home` to the value `/path/to/sindice/home`.

As a super-user, edit the file `/etc/default/tomcat6` and update the `JAVA_OPTS` variable:

```
JAVA_OPTS="-Dsindice.home=/path/to/sindice/home"
```

Set tomcat6 permissions on that folder, so that the deployed webapp can write in that folder:

```
$ sudo chown -R tomcat6:tomcat6 /path/to/sindice/home
```

Create the Webapp .war File

Edit the default XML config file of SparQLed:

```
recommendation-servlet/src/main/resources/default-config.xml
```

Set the URL to the endpoints loaded with the original RDF Graph and the DGS. The endpoint with the RDF Graph is set under the `proxy` tag, the one with the DGS graph is set under the `recommender` tag.

recommender and **proxy** tags

We assume here that both graphs are loaded into the same, `HTTP` endpoint:

```
<backendArgs>http://path/to/sparql/endpoint</backendArgs>
```

The `sparqled.war` file is created using the following command, and is located in the directory `recommendation-servlet/target/`:

```
$ mvn --projects recommendation-servlet --also-make
package
```

3. Launching the SparQLed Webapp

With the SPARQL endpoint ready and the configuration files correctly setup, we are now ready to launch the webapp!

Clean Previously Installed Webapp

Remove a previously deployed SparQLed webapp, in order to avoid unexpected issues:

```
$ rm -rf /path/to/sindice/home/sparqled
$ rm -rf $CATALINA_BASE/webapps/sparqled*
```

On Ubuntu, $CATALINA_BASE=/var/lib/tomcat6/

Run Tomcat

Copy the `sparqled.war` file to the tomcat base directory and start tomcat:

```
$ sudo cp recommendation-servlet/target/sparqled.war
$CATALINA_BASE/webapps/
$ sudo service tomcat6 start
```

Using SparQLed

If no problem, the SparQLed (Figure 2) webapp is available at:

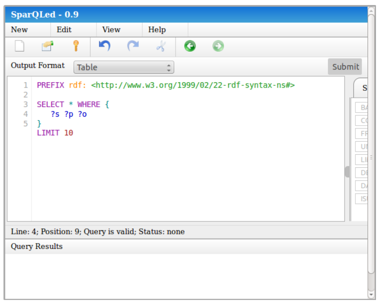
```
http://localhost:8080/sparqled/
```

In case the webapp does not work properly, please have a look at the logs:

```
$ tail -f $CATALINA_BASE/logs/catalina.out
$ tail -f /path/to/sindice/home/log/sparqled/sparqled.log
```

The SparQLed[5] editor gives you four types of recommendations, i.e., class, predicate, relationship between variables and named graphs.

In order to get recommendations you have to hit `Ctrl + Space` at the desired position in the query. The recommendations generated by the editor will be provided in the form of a drop down list. The user can then select one entry from the drop down box that matches his criteria. Keyword (resp., prefix) search is also possible by typing a word (resp., prefix), followed by `Ctrl + Space`.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT * WHERE {
   ?s ?p ?o
}
LIMIT 10
```

Line: 4; Position: 9; Query is valid; Status: none

Figure 2: The SparQLed editor

[5] A live version and a screencast are available at `http://hcls.sindicetech.com/sparql-editor/`

KNOWN ISSUES

The editor use the SPARQL1.0 grammar. Therefore, there is no syntax highlighting nor syntax recommendations[6] for SPARQL1.1 keywords. However, it is still possible to submit a 1.1 query to the endpoint. This issue will be resolved in the near future.

[6] not to be confused with the structural recommendations the SparQLed provides

ACKNOWLEDGEMENT

This software is based upon works supported by the European FP7 project LOD2 (257943).

CONTACT

If you have any questions or issues, please contact us at `http://groups.google.com/group/sindice-dev`.

Bibliography

[1] Stephane Campinas, Thomas E. Perry, Diego Ceccarelli, Renaud Delbru, and Giovanni Tummarello. Introducing RDF Graph Summary With Application to Assisted SPARQL Formulation. *WebS, DEXA workshop*, 2012.

Appendix

A. How to Query the Data Graph Summary ?

The Data Graph Summary is modeled using the *Dataset Analytics Vocabulary*[7] ontology. Below are some examples on how to use the Data Graph Summary.

[7] `http://vocab.sindice.net/analytics`

- The following query returns the properties used with the class `http://schema.org/Person` in the dataset `bbc.co.uk`:

```
PREFIX an: <http://vocab.sindice.net/analytics>
PREFIX any23: <http://vocab.sindice.net/>
PREFIX d: <http://sindice.com/dataspace/default/domain/>

SELECT DISTINCT ?Predicate
FROM <http://sindice.com/analytics>
WHERE {
    ?node an:label ?l .
    ?l an:label <http://schema.org/Person> .

    ?edge an:source ?node ;
          an:label ?Predicate ;
          an:publishedIn d:bbc.co.uk .
}
LIMIT 20
```

- The following query returns the most used classes in a dataset:

```
PREFIX an: <http://vocab.sindice.net/analytics>
PREFIX any23: <http://vocab.sindice.net/>

SELECT ?Dataset ?Class (SUM(?card) AS ?Cardinality)
FROM <http://sindice.com/analytics>
WHERE {
    ?node any23:dataset_uri ?Dataset ;
          an:cardinality ?card ;
          an:label ?l .
    ?l an:label ?Class .
}
GROUP BY ?Dataset ?Label
ORDER BY DESC(?Cardinality)
LIMIT 20
```

B. How to setup SparQLed from scratch ?

In this Section, we assume that you have no SPARQL endpoint and no RDF data available. We explain you how to setup a SparQLed instance from scratch.

Example Datasets

One may download one of the following datasets:

- the European Nature Information System[8] dataset (large, long setup):

```
$ wget http://eunis.eea.europa.eu/rdf/{sites,habitats,
  species,taxonomies}.rdf.gz --directory-prefix eunis
```

- the New York Times - Linked Open Data[9] dataset (small, fast setup):

```
$ wget http://data.nytimes.com/{locations,organizations,
  people,descriptors}.rdf --directory-prefix nytimes
```

[8] http://thedatahub.org/dataset/eunis

[9] http://thedatahub.org/dataset/nytimes-linked-open-data

We need to load the downloaded RDF data into a SPARQL endpoint. We create here a Sesame Native[10] endpoint:

[10] http://www.openrdf.org/doc/sesame2/users/ch07.html

```
 $ java -cp target/sparql-summary-assembly.jar
org.sindice.summary.Pipeline --feed --type NATIVE
--repository /path/to/<data>-native --add
/path/to/<data> --addformat=RDF/XML
```

The format of the ingested RDF data is in NTriples by default. A different format can be given through the option `--addformat`.

Compute the Data Graph Summary

To compute the DGS over the downloaded data, we will execute SPARQL queries against the previous Native repository. This creates the file `/path/to/<data>-summary.nt.gz` containing the summary in the NTriple format.

```
 $ java -cp target/sparql-summary-assembly.jar
org.sindice.summary.Pipeline --type NATIVE
--repository /path/to/<data>-native --outputfile
/path/to/<data>-summary.nt.gz
```

We load the DGS triples into a new Sesame Native repository `/path/to/<data>-summary-native` under the named graph `http://sindice.com/analytics`. The reason to create a new repository is that Sesame locks the Native repository when in use, preventing another connection to be made.

```
 $ java -cp target/sparql-summary-assembly.jar
org.sindice.summary.Pipeline --feed --type NATIVE
--repository /path/to/<data>-summary-native
--domain http://sindice.com/analytics --add
/path/to/<data>-summary.nt.gz
```

Configure the Webapp

Since we are now using a Native repository, the SparQLed configuration file needs to be updated:

- Edit the SparQLed XML configuration file:

```
recommendation-servlet/src/main/resources/default-config.xml
```

- Edit both endpoints, i.e., for the original RDF Graph and for the DGS, to point to the Native repositories:

```
<backend>NATIVE</backend>
<backendArgs>/path/to/<data>-native</backendArgs>
```

proxy tag

```
<backend>NATIVE</backend>
<backendArgs>/path/to/<data>-summary-native</backendArgs>
```

recommender tag

Launching the SparQLed Webapp

Before following the steps detailed in the Section 3, you have to grant tomcat6 (or, tomcat7) the ownership to the native repositories:

```
$ sudo chown -R tomcat6:tomcat6 /path/to/<data>-native
$ sudo chown -R tomcat6:tomcat6 /path/to/<data>-summary-native
```