



Semantic Web Service Matchmaker Evaluation Environment

SME²

Version 2.2

User Manual

01.12.2010

Matthias Klusch, Minko Dudev, Jozef Mišutka
Patrick Kapahnke, Martin Vasileski



Deutsches Forschungsinstitut für Künstliche Intelligenz DFKI GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken

Copyright ©: DFKI, 2010, All Rights Reserved

Table of Contents

1. Introduction	- 2 -
2. Using SME² Evaluation Environment.....	- 3 -
2.1. Adding Plugins	- 3 -
2.2. Starting SME ²	- 3 -
2.2.1. Requirements and Dependencies	- 3 -
2.3. User Interface	- 3 -
2.3.1. Configuration Tab	- 4 -
2.3.2. Results Tab	- 5 -
2.4. Configuration File	- 6 -
3. Plugin Development	- 7 -
3.1. Creating a Plugin	- 7 -
3.1.1. IMatchmakerPlugin Interface	- 7 -
3.2. Plugin Deployment.....	- 8 -
3.3. OWLS-MX Example Plugin	- 9 -
3.4. S3 2010 Plugins.....	- 10 -
4. Test Collections.....	- 11 -
4.1. Test Collection Setup	- 11 -
4.2. Provided Test Collections	- 12 -
5.0 Performance Evaluation Tests	- 13 -
5.1. Recall/Precision-based Measures	- 13 -
5.2. Query Response Time	- 15 -
5.3. Memory Consumption.....	- 15 -
5.7. Statistical Significance of Results	- 15 -
6. Contact and Copyright	- 17 -
References	- 18 -

1. Introduction

A semantic Web service matchmaker is a piece of software that stands between the user and the service repository providing functionality for locating services that are relevant to a users query. The matchmaker is at the core of more complex components, for instance, the service broker which takes a more active role in the service location tasks and can be seen as an extension of the semantic matchmaker on which it is built. Thus, it is crucial to be able to measure the performance of semantic matchmakers in terms of classical measures used in Information Retrieval along with resource consumption and responsiveness. A focus during the development was simplicity and user friendliness as far as the heterogeneity of the matchmakers allows it.

In this manual we present the features of the Semantic Service Matchmaker Evaluation Environment SME² which was designed as an integrated environment that provides an extensible framework for the testing of different semantic matchmakers in a consistent way. Different test collections and matchmakers can be added as plugins to the environment to conduct performance experiments. The framework can be easily extended to support various types of matchmakers.

Currently SME² is delivered with source code for an example plug-in for OWLS-MX ([1], available for download at <http://www.semwebcentral.org/projects/owls-mx/>, stand-alone version) which is a popular program supporting the matching of OWL-S services and queries. The authors of other matchmakers are encouraged to write a plugin following this manual and make it available to others for testing. SME² also allows for the configuration of different test collections. The test collection provided as additional packages with this release are OWLS-TC 4 (<http://www.semwebcentral.org/projects/owls-tc/>) and SAWSDL-TC 3 (<http://www.semwebcentral.org/projects/owls-tc/>). OWLS-TC 4 contains 42 queries and 1083 services in OWL-S 1.1, SAWSDL-TC 3 contains 42 queries and 1080 services in SAWSDL using WSDL 1.1. In future versions we plan to include also plugins and test collections for WSMO.

This tool has been used in the yearly Semantic Service Selection contest (S3). More detailed information on this can be found at <http://www-ags.dfki.uni-sb.de/~klusck/s3/>. Semantic Web service matchmaker developers are cordially invited to participate. This release of SME² includes all entries of the 2010 edition of S3 to reduce the effort of comparative performance analysis with respect to some of the most up-to-date matchmaker implementations available.

2. Using SME² Evaluation Environment

2.1. Adding Plugins

The SME² evaluation environment evaluates plugins. Plugins must be copied to a specific directory. This directory is specified in the `settings/settings.xml` directory of the SME² tool. The default directory is `plugin`. Each plugin must contain the interface implementation, an XML specification file and optionally directories with libraries the matchmaker depends on.

The best way is to create a new directory for each matchmaker and to copy all the files as well as the XML specification there: the JAR containing the interface implementation and directories containing libraries the matchmaker depends on. After copying the plugin to that directory, it should be automatically recognized by SME² at the next startup. The program searches the plugin directories for all plugin specification files and dynamically loads the libraries.

2.2. Starting SME²

Running the program itself is simple: execute the `sme2.bat` provided in the distribution. This should launch a GUI window that will allow you to configure the experiment, view results, print reports and examine the result graphs. All necessary libraries for SME² are contained in the distribution in the `libs` directory.

2.2.1. Requirements and Dependencies

The evaluation environment is implemented for Java 2 (1.6 JRE or higher).

SME² has a minimum set of dependencies. Charting is employed using the *JFreeChart* library (<http://www.jfree.org/jfreechart/>). For working with XML the *Apache Xerces* library (<http://xerces.apache.org/>) is used. The PDF report generation is done using the *iText* library (<http://www.lowagie.com/iText/>). All these libraries are freely available on the internet and are included with the distribution in library folder.

2.3. User Interface

SME² has been designed to have a simple, straightforward user interface. Below, its features are explained grouped into two panels. The configuration panel allows setting up the experiment while the results panel provides data related to the outcome of the measurements. Please refer to Figure 1 and Figure 2 for the numbering of the controls on the configuration tab and the results tab respectively.

2.3.1. Configuration Tab

The following screenshot (Figure 1) shows the configuration part of SME², which will be automatically presented at startup. Here, the setup of experiments can be accomplished and evaluation can be started.

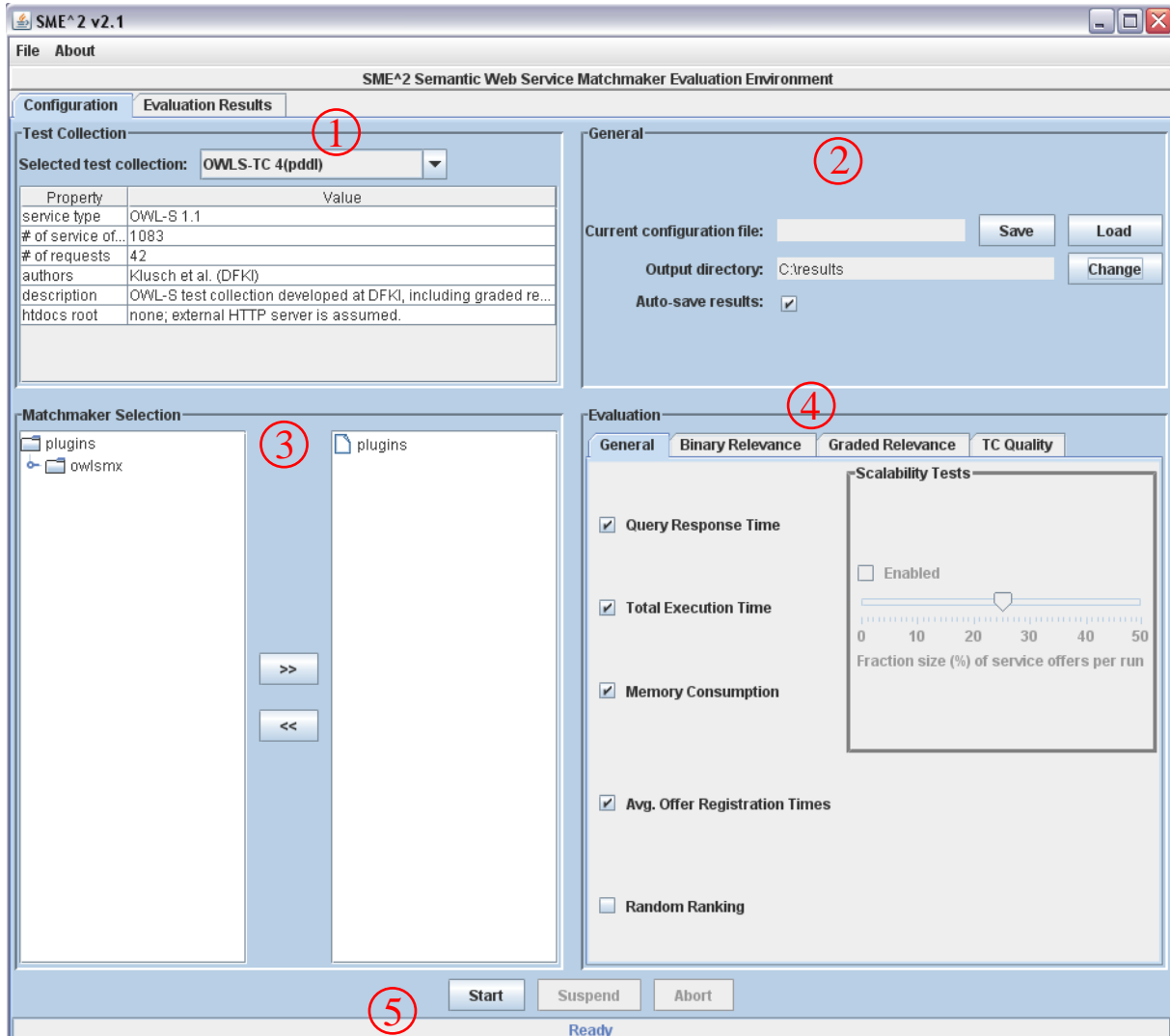


Figure 1 Configuration part of the GUI.

1. *Test Collection*: Provides a simple drop-down menu to select one of the available test collections. Some properties regarding the selected test collection are shown automatically.
2. *General*: Enables the user to *Save* and *Load* the settings for this page. Please note, that the selected matchmaker plugins are not saved and loaded in this version. This part also allows to enable the *Auto-save results* option. If the corresponding check box is activated, results produced during the evaluation step will be automatically saved for each processed matchmaker plugin. The files will be written to the specified *Output directory*.
3. *Matchmaker Selection*: In this control element, available matchmaker plugins are listed on the left side. The plugins, which should be evaluated can be selected (single

selection, multiple selection or selection of whole groups in the tree) and pushed to the right side, where all matchmakers that will be tested are listed.

4. *Evaluation*: This section allows the configuration of the performance evaluation tests. Various evaluation measures can be selected using the appropriate check boxes. For a detailed description of the different evaluation variants, we refer the reader to section 0. Please note, that not all measures are implemented in this version of SME². The features that are missing are deactivated and are subject to future work. This includes the measures for *Graded Relevance* assessments in a test collection as well as test collection *Quality* and matchmaker algorithm *Scalability* tests.
5. *Control Panel*: The lower part of this tab contains control buttons to *Start*, *Suspend (Resume)* and *Abort* the performance evaluation process. If the process is started, all configuration controls are inactive until the evaluation is finished or stopped manually. Using the *Suspend* button, it is possible to pause the evaluation process. All time-related measurements are stopped too during a suspension phase. The bar at the bottom provides status information in terms of text and visual progress display.

2.3.2. Results Tab

In the following, the graphical user interface part for result presentation is presented. After the performance evaluation process, it can be selected using the tab on the top. A typical situation is depicted in Figure 2.

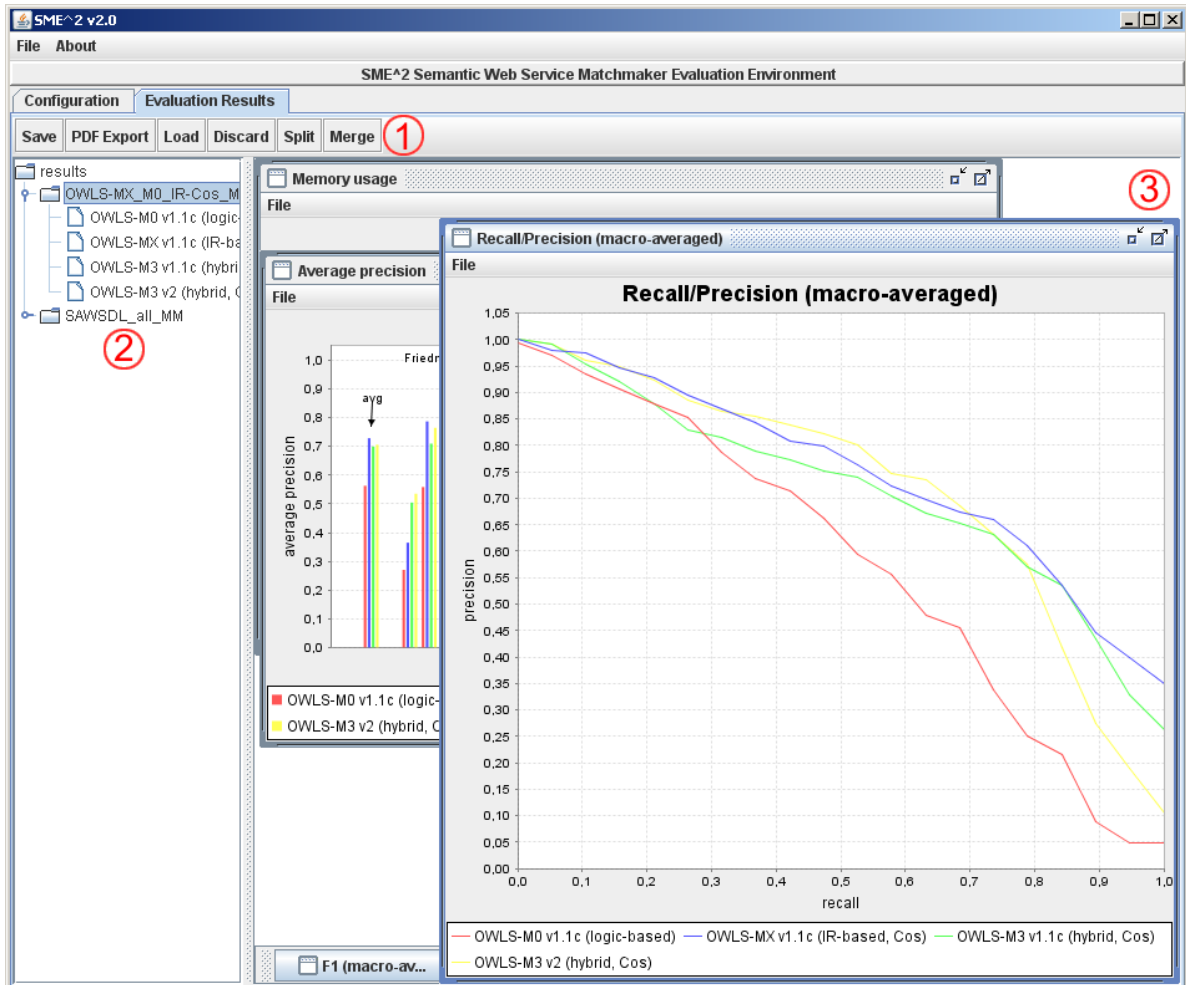


Figure 2 Result presentation part of the GUI.

1. *Toolbar*: The toolbar at the top of the results tab allows to *Save* a selected experiment (see 2) as XML document, *Load* XML documents produced with SME², export evaluation results to PDF using the iText library (*PDF Export*), *Discard*, *Merge* and *Split* experiments. To apply one of the latter three operations, one or more experiments have to be selected (see 2).
2. *Experiment Selection*: In this user interface part, several experiments produced with SME² (or loaded from previous sessions) are listed. Each experiment may consist of results for different matchmakers, which is represented by the leaf nodes of the tree view. To apply a *Discard* operation, a single experiment has to be selected. To *Merge* two experiments, two experiments have to be selected (not the leaf nodes listing the matchmakers). To *Split* an experiment, it has to be selected separately. New experiment entries will be created for every matchmaker contained in the original experiment. If an experiment is selected on this side, the performance evaluation results will be displayed on the desktop view on the right (see 3).
3. *Evaluation Result Desktop*: Depending on the selection made on the configuration tab, different windows regarding the selected experiment appear on the desktop view. They may be minimized, maximized and moved around as usual for window-like environments. For the graph windows, there exists an option to save images to file using the menu at the top of the appropriate element. A context menu for changing the display style of the graphs produced with *jFreeChart* is accessible by right-clicking on a window main view. The table-like content of the *Info* window provides functionality to save the selected information as text or to directly send it to a printer.

2.4. Configuration File

The directory `settings` contains an XML configuration file named `settings.xml`, where some global configuration information is stored. For the current version of SME², the following properties can be set:

- `plugins`: Directory, where plugins are located
- `testcollections`: The location of the test collections

It is planned to add all possible additional properties required in future versions of SME² to this file.

3. Plugin Development

3.1. Creating a Plugin

Developing a new matchmaker plugin for the SME² environment requires several simple steps:

- Implement the provided interface `de.dfki.sme2.IMatchmakerPlugin` and create a Java library.
- Create an XML plugin description file.
- Copy the plugin library, the XML file and additional required libraries to the `plugin` directory of SME² (or the appropriate directory if global configuration was changed).

It is a good idea to create a new subdirectory for each matchmaker to add, because SME² automatically structures the user interface depending on the found directory structure. However, different implementations or configurations of one matchmaker type can be set up by adding more than one XML description file to one plugin directory. The result in the user interface will be a tree-like structure derived from the directory structure and the discovered XML descriptions.

After all files are prepared and copied, the new plugins should appear after starting the tool using the provided `sme2.bat`.

3.1.1. IMatchmakerPlugin Interface

This version of SME² has new plugin interface which defines five methods. However, compatibility with the older version of the plugin interface is retained and SME² can handle both versions simultaneously. The user can choose which version of the plugin interface he wants to implement, but it is required from the user to include the plugin interface version number in the plugin XML description file.

```
void parseOffer(URI serviceURI)
```

Initializes the service registry of a matchmaker. An advertised service identified by `serviceURI` should be parsed in this method.

`serviceURI`: A URI that points to a service description file.

Please note, that the functionality of passing a whole directory to this method is not used in the current version of SME² anymore. The tool will call this method for every single service offer of the selected test collection.

```
void processOffer(URI serviceURI)
```

An advertised service identified by `serviceURI` should be processed in this method. After this method has been called for every service offer in the test collection, the matchmaker should be ready to accept queries.

`serviceURI`: A URI that points to a service description file.

Please note, that the functionality of passing a whole directory to this method is not used in the current version of SME² anymore. The tool will call this method for every single service offer of the selected test collection.

```
void parseQuery(Uri queryUri)
```

A query should be parsed in this method by a matchmaker.

queryUri: A URI that points to service description file used as a request.

```
void processQuery(Uri queryUri)
```

A query should be processed in this method by a matchmaker.

queryUri: A URI that points to a service description file used as a request.

```
Vector<Uri> match(Uri queryUri)
```

This method performs a matching, returning a ranked result list (Vector) for the query specified by queryUri.

queryUri: A URI that points to a service description file used as request.

return A vector with ranked list of matching results.

The results should be sorted in decreasing order, i.e. the best matching result should be at position 0 followed by the second result at position 1 and so on.

3.2. Plugin Deployment

In addition to implementing the plugin interface, a XML plugin description file must be created. In the following, an example document is provided:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <jar>OWLSMX.jar</jar>
  <class>de.dfki.owlsmx.sme2.OWLSM0</class>
  <directory recursive="true">lib</directory>
  <name>OWLS-M0 v1.1c (logic-based)</name>
  <version>2.0</version>
</plugin>
```

As can be seen, the document element is named `plugin`. It does not allow any attributes, the following child elements are allowed in any order:

- `jar`: Specifies the path to the JAR file containing the interface implementation. If it is given as relative path, the location of the XML file should be used as base. This element is mandatory and may occur exactly once.

- **class:** Points to the class that should be used as plugin implementation. The full package name must be included. This element is mandatory and may occur exactly once.
- **name:** Using this tag, a user-friendly name can be specified. It is optional and may occur at most once. If a name is specified, it will be used in the user interface of SME². If it is missing, the file name of the XML description will be used as name.
- **directory:** Specifies a directory, from which all libraries should be dynamically loaded in the context of this plugin. This element is optional and may occur more than once. Classes are loaded in order of occurrence inside the directory.
 - **recursive:** Is a boolean attribute that can be used to direct the plugin loader to recurse into subdirectories or not.
- **lib:** Specifies a single library, which should be loaded for the plugin. This element is optional and may occur more than once and in arbitrary combination with **directory**.
- **version:** Specifies a version number of the plugin interface. It can have two values: 1.0 (old version), or 2.0 (new version). If this tag is not defined, SME² by default assumes that older version of the plugin interface is implemented.

Please note, that all libraries specified in the XML description are loaded in exactly the same order as the corresponding directives appear in the XML file. Using this mechanism, a plugin developer can control the order of libraries in the internal class loader (which may be important if e.g. different versions of libraries are used inside a single plugin).

3.3. OWLS-MX Example Plugin

The SME² project contains an additional package containing an example plugin for the hybrid semantic Web service matchmaker OWLS-MX ([1]). All relevant files explained above can be found at <http://www.semwebcentral.org/projects/sme2/>. Additionally, the source code implementing the `IMatchmakerPlugin` interface can be found in the `src` subfolder of the plugin download.

To test the SME² application and the OWLS-MX plugin, simply start SME² as explained in section 2.2 after copying the plugin to the appropriate directory, select one or more of the OWLS-MX variants to perform experimental evaluation, set up the tests as described in section 2.3.1 and start the evaluation. Please note, that a test collection must be installed too. OWLS-TC 4 is provided as additional package. In the previous version of SME², the test collection required a local Web server that provides the ontologies used to describe the services, but in this version a Jetty Http Server is implemented that acts as local Web Server. Details on this can be found in section 4.2 and are specific to the test collections and not to OWLS-MX itself.

3.4. S3 2010 Plugins

For the version 2.2 release of SME², all plugins that participated in the 2010 edition of the S3 contest have been included to reduce the effort of comparative performance analysis with respect to some of the most up-to-date matchmaker implementations available. The packages are located in the `plugin` directory.

In the following, an exhaustive list of the contained entries is given subdivided by the track they were submitted for (i.e. they are known to work with the corresponding test collections):

- **Track 1:** OWL-S matchmakers, test collection OWLS-TC 4.0
 - o iSeM 1.0 (DFKI, Germany)
 - o OWLS-MX3 (DFKI, Germany)
 - o SeMa2 (TU Berlin, Germany)
 - o OWLS-iMatcher2 (U Zurich, Switzerland)
 - o SPARQLent (HP, Italy)
 - o OWLS-SLR (Aristotle U of Thessaloniki, Greece)
 - o XSSD (Beihang U, China)
 - o EMMA (U Seville, Spain)

- **Track 2:** SAWSDL matchmakers, test collection SAWSDL-TC 3.0
 - o LOG4SWS.KOM (TU Darmstadt, Germany)
 - o COV4SWS.KOM (TU Darmstadt, Germany)
 - o iSeM 1.0 (DFKI, Germany)
 - o SAWSDL-MX1 (DFKI, Germany)
 - o URBE (Politecnico di Milano, Italy)
 - o SAWSDL-iMatcher3 (U Zurich, Switzerland)

More details including contact information and an overview of the functioning of each plugin can be found in the S3 summary report for 2010: <http://www-ags.dfki.uni-sb.de/~klusck/s3/s3c-2010-summary-report-v2.pdf>.

Please note: Some plugins have specific requirements that must be met to ensure that they work without problems and produce correct results. Please have a look at the `doc` folder of the SME² distribution for details.

4. Test Collections

SME² provides a simple way to add test collections for evaluation of semantic Web service matchmakers. However, we intend to improve this functionality in future versions of SME² and the procedure described in section 4.1 will be more tightly integrated with other tools supporting an XML format for describing test collections (e.g. SWSRAT, a tool for supporting binary as well as graded relevance assessments in large test collections, which is still under development).

4.1. Test Collection Setup

A test collection (TC) for the current version of SME² consists of a set of service offers, a set of service requests and a relevance set for each of these requests structured in a fixed file hierarchy, which is as follows:

- `<TC root folder>`: usually `testcollections` if not changed in the settings file.
 - `<TC folder>`: may have an arbitrary name reflecting the test collection and contains exactly one test collection. There may exist several folders for different test collections at this level.
 - `queries/<type>`: contains the request documents. Please note that an additional folder specifying the type of the descriptions is needed (this is a relict of other tools developed at our group such as OWLS-MX, however it is ignored for SME²). The folder `queries` should be empty except this subfolder.
 - `relevance_sets`: contains a subfolder for each query.
 - `<domain>-<query name>`: contains copies of all service offers relevant for the specified query.
 - `services/<type>`: contains all service offers. Analogous to the `queries` folder, there has to be a type subfolder.

The elements contained in sharp parentheses (`<>`) are placeholders for user-defined or setting-dependent folder or file names. Please refer to the explanations given at these elements.

Additionally, the `<TC folder>` must contain an XML description file for the test collection. A self-explaining example is given in the following. Every new test collection has to provide such an XML description with exactly the same structure.

```
<testcollection>
  <proprietary/>
  <name>OWLS-TC 2.2 revision 2</name>
  <authors>Klusck et al. (DFKI)</authors>
  <type>OWL-S 1.1</type>
  <description>OWL-S test collection developed at DFKI</description>
  <htdocs>testcollections/owls-tc4(pddl)/htdocs</htdocs>
</testcollection>
```

For example test collections, we refer to the additional packages provided for SME², which are explained in more detail in the following section.

4.2. Provided Test Collections

In addition to the SME² tool itself, there exist two test collection packages, which can be installed by simply copying the extracted content to the test collection root folder. There exists a test collection for OWL-S, namely OWLS-TC 4, as well as a test collection for SAWSDL, namely SAWSDL-TC 3. They can be found in the same project as SME² for the latest release at <http://www.semwebcentral.org/projects/sme2/>.

The services provided with these test collections refer to ontologies located at the test collection folder or local HTTP host. These ontology files are provided with the package and the relative path from their location should be included in the <htdocs> tag from the TC XML description file, in order to activate the Jetty Web Server as a local host. Otherwise they can also be made accessible by installing an HTTP Web server such as the *Apache* Web server.

5.0 Performance Evaluation Tests

SME² supports various performance evaluation tests based on well-known measures such as *Recall* and *Precision*, as well as tests regarding the run time, number of HTTP request per query, HTTP access time per query and memory consumption of matchmaker plugins. Currently, only tests for test collections based on binary relevance assessments are implemented. However, it is planned to also support graded relevance assessments. Please note, that the user interface elements for selecting evaluation strategies based on these graded relevances can already be found in SME² version 2.0 but are inactive.

5.1 Recall/Precision-based Measures

SME² produces graphs for measures well-known from information retrieval based on the following definitions:

$$\begin{aligned} Recall &= \frac{|A \cap B|}{|A|}, \\ Precision &= \frac{|A \cap B|}{|B|}, \\ Fallout &= \frac{|\bar{A} \cap B|}{|\bar{A}|}, \end{aligned}$$

where A is the set of all relevant documents for a request, B the set of all retrieved documents for a request and \bar{A} the set of all non-relevant retrieved documents for a request.

To produce results for a complete series of tests (using all available request documents defined in a test collection), macro-averaging as well as micro-averaging is adopted.

The *macro-averaged* precision computes the mean of precision values for answer sets returned by a matchmaker for all queries in the test collection at equidistant standard recall levels $Recall_i, 0 \leq i < \lambda$ and is defined as follows:

$$Precision_{macro}(i) = \frac{1}{|Q|} \cdot \sum_{q \in Q} \max \{ P_o | R_o \geq Recall_i \wedge (R_o, P_o) \in O_q \},$$

where Q is the set of request documents, O_q denotes the set of observed pairs of recall and precision values for query q when scanning the ranked services in the answer set for the query stepwise for *true positives*.

Micro-averaging is defined as follows:

$$Recall_{micro}(i) = \sum_{q \in Q} \frac{|A_q \cap B_{i,q}|}{\tilde{A}},$$

$$Precision_{micro}(i) = \sum_{q \in Q} \frac{|A_q \cap B_{i,q}|}{\tilde{B}_i},$$

where i is the evaluation level with $0 \leq i < \lambda$, A_q the set of all retrieved relevant documents for request $q \in Q$, $B_{i,q}$ the set of all retrieved documents up to evaluation level i for the request, \tilde{A} the sum of the magnitudes of all relevance sets in the test collection and \tilde{B}_i the number of retrieved documents for all queries at i . The computation of recall/fallout values for the graphs is analogous considering false positives and the number of irrelevant documents.

In addition to these measures, the *F1* measure is provided. It combines recall and precision to a single value as follows:

$$F1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}.$$

The graphs generated from the F1 measure computes values for different recall levels $0 \leq i < \lambda$ using the macro-averaging or micro-averaging approach.

The well-known *Average Precision* measure, which produces a single-valued rating of a matchmaker for a single query result ranking is also implemented in SME². It is defined as follows:

$$AP = \frac{1}{|R|} \sum_{r=1}^{|L|} isrel(r) \frac{count(r)}{r},$$

where R is the set of relevant documents according to the predefined relevance sets, L the ranking for the query in question, $isrel(r) = 1$ if the item at rank r is relevant and 0 otherwise $count(r) = \sum_{i=1}^r isrel(i)$. The Average Precision measure enables performance evaluation invulnerable to varying sizes of returned rankings. SME² produces a bar chart for *AP* values, where each bar represents the result for a single request document and matchmaker. The mean Average Precision is also presented.

In this version of SME² two new precision measures are implemented, Precision at k and R-precision. Precision at k evaluates the standard precision of returned results at a given cut-off rank, considering only the topmost k results returned by a matchmaker for each query. It has the advantage of not requiring any estimation of the number of returned service offers per query but the disadvantage is that it does not average well, since the total number of relevant service offers for a query has a strong influence on precision at k .

On contrary, R-precision averages well. It requires having a set of known relevant service offers R , from which the precision is calculated of the topmost R service offers returned by a matchmaker. The average value of R-precision is also computed and presented.

5.2. Query Response Time

The query response time measurement in this version is evaluated into more details. It is represented with three complementary time measurements: query parsing time, query processing time and actual matching time, i.e. the time taken by a matchmaker to compute the service ranking for a single query. It is displayed as a stacked bar chart in SME², including the average detailed query response time.

Additionally, the overall computation time of a matchmaker (i.e. the time a matchmaker takes to register all service offers of a test collection and to match all request documents) is also given.

5.3. Memory Consumption

The memory usage is measured for the whole evaluation process including the service registration phase as well as the query processing. It is displayed as graph using a time axis. Please note, that *Java garbage collection* artefacts may be included in this test.

5.4. Average Offer Registration Times

The average offer registration times are represented by the average offer parsing time, i.e. the time needed for a matchmaker to parse a service offer, and average offer processing time, i.e. the time needed for a matchmaker to actually process and register service offer.

5.5. Random Ranking

In this version of SME² is provided a feature called Random Ranking, which represents a pseudo matchmaker that receives service offers as an input, and returns randomly ranked results for each given query. All evaluation measures that can be calculated for a standard matchmaker, excluding the time measures, can also be used to evaluate the performance of this pseudo matchmaker.

5.6. Http Request Statistics

With the implementation of Jetty Web Server in this version of SME², detailed Http request statistics is collected per query. It includes number of Http requests made per query and Http access time per query.

5.7. Statistical Significance of Results

The evaluation tool also supports a statistical significance test checking whether matching results of different matchmakers in terms of *Average Precision* are significantly different at a level α ($\alpha = 0.05$ is usually adopted), namely the *Friedman Test*. This is a non-parametric test for simultaneously analyzing ranked result sets of at least two different (service matching) methods and has been shown in [2] to be a vital explanatory component of a comparative retrieval performance evaluation.

SME² uses the Friedman Test variant proposed in [3]. The resulting *p-value* indicates, if there is a significant difference between the variants which one can not interpret as being an implication of the *null hypothesis*, i.e. that variations of the matchmaker rankings per query are insignificant.

Please note, that in case of significant differences, additional tests have to be performed, because the *Friedman Test* does not determine the significant outlier for a given set of matchmakers.

6. Contact and Copyright

The Semantic Web Service Matchmaker Evaluation Environment SME² was developed at the German Research Center for Artificial Intelligence DFKI GmbH (<http://www.dfki.de>) in Saarbrücken, Germany. This work was sponsored in part by the project SCALLOPS (BMBF, 3/2004 - 6/2007)).

Copyright ©: DFKI, 2010, All Rights Reserved.

For bug reports, other technical problems and feature requests please contact Patrick Kapahnke: patrick.kapahnke@dfki.de.

For general scientific inquiries please contact Dr. Matthias Klusch: klusch@dfki.de.

References

- [1] Klusch, M.; Fries, B.; Sycara, K. (2009): *OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services*. International Journal of Web Semantics, 7(2), Elsevier
- [2] Hull, D.: *Using statistical testing in the evaluation of retrieval experiments*. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, pages 329-338, 1993
- [3] Iman, R.L., Davenport, J.M.: *Approximations of the critical region of the friedman statistic*. Communications in Statistics, A9:571-xxx, 1980