SciPlore Literature Recommendation Evaluator

1 Architectural Overview

The SciPlore Literature Recommendation Evaluator (LRE) is a web-based tool to conduct surveys about the relevance of literature recommendations. The recommendations can be generated using different similarity measures. Therefore, the LRE can serve to establish a gold standard dataset, e.g., for evaluations carried out using data of the CITREC framework.

The LRE comprises of six independent components implemented using the symfony2 framework. This architecture allows for easy integration of new components and quick substitution of existing components. The symfony2 framework uses the term “bundle” as a synonym for component. Hereafter, we briefly describe the six components and their interaction.

The *DataAccessBundle* uses the Object-Relational-Mapping (ORM) framework Doctrine for managing the persistent storage of data in the LRE. For mapping objects to database tables, Doctrine requires more information than PHP classdefinitions of components can provide. Special comments within the source code provide this additional information, e.g. “@ORM\Column(type="string", nullable=false)”.

The *UIBundle* controls the display of the website presented to the user.

The *UserManagementBundle* employs authorization and authentication components of the symfony2 framework for managing user logins and access protection. Chapter 13 [1] provides a detailed documentation for these components.

The *AnalysisBundle* serves to visualize results using the Google Chart API.

The *FlowManagementBundle* controls the sequence in which the LRE presents recommendations to users.

Components communicate and interact with each other through sending function calls to services. A service is a public accessible object. A component can provide several external interfaces through services. For example, the *DataAccessBundle* offers a service for retrieving document objects, which the *UIBundle*, the *AnalysisBundle* and the *ValidationBundle* use.

The *Service Container* is a special service registry in the symfony2 framework that globally links every service to a unique name and manages the existing instance of every service. Thanks to the Service Container, components can access a service through its name. Chapter 16 in [1] describes the service container in more detail.

Most services of the SciPlore LRE are stateless. Some services, such as the *FlowManagementBundle*, which controls the display order of recommendations, utilize the session service to store data during execution for later requests. The *FlowManagementBundle* uses the session service to store an array of recommendations and

their individual processing state for the current experiment instead of using the *DataAccessBundle* to retrieve full document objects for each request.

2 Database Schema

The SciPlore LRE uses as MySQL database. Figure 1 depicts the database schema.

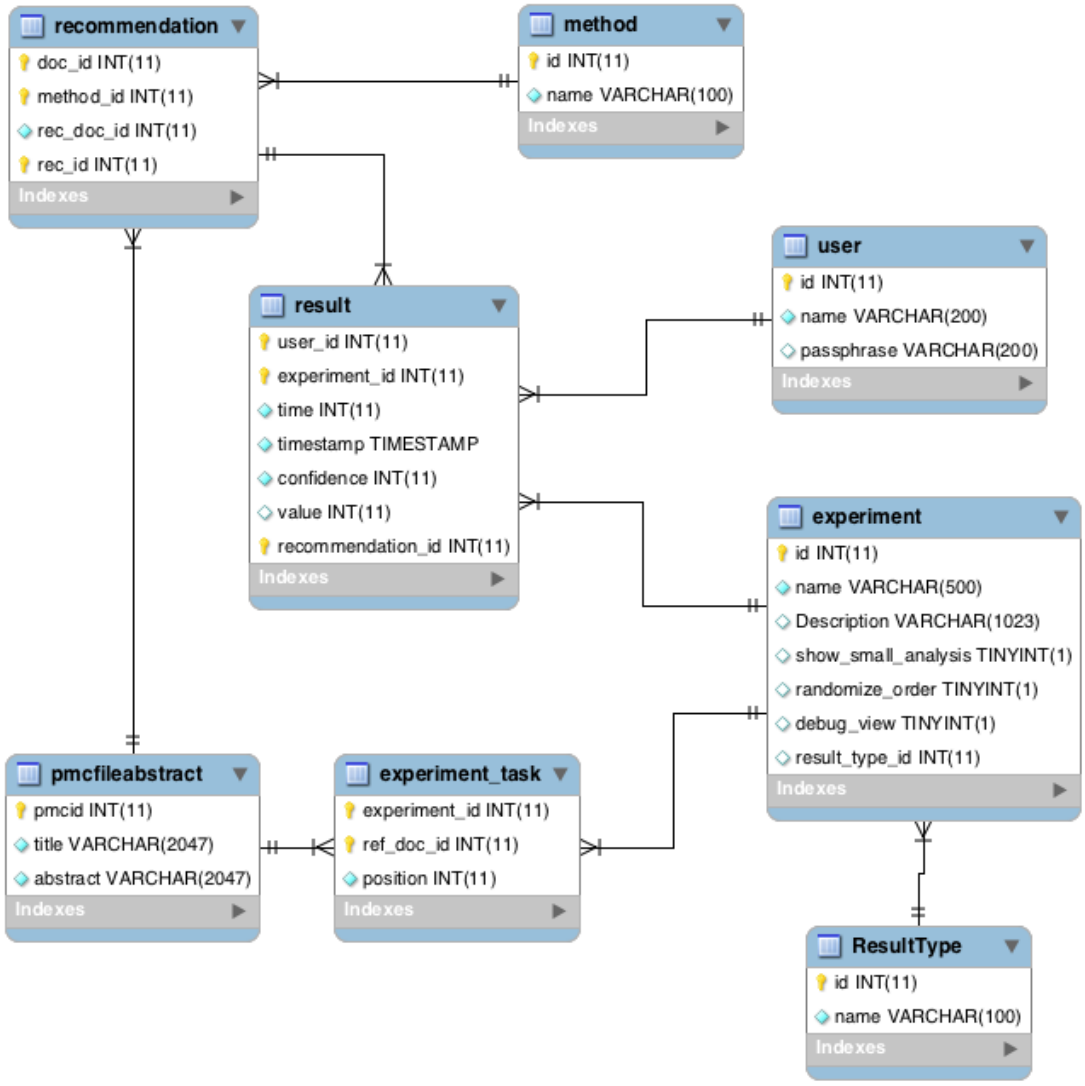

Figure 1: Database Schema

3 Setup

The SciPlore LRE requires a web server stack that includes PHP (minimum version 5.3.3) and MySQL.

3.1 Installation

1. Clone the current version of the SciPlore LRE git repository:
   ```
   $ git clone https://bitbucket.org/sciplore/lre.git {dir}
   $cd {dir}
   ```
   Hereafter, we state all paths relative to the chosen installation directory {dir}.

2. Initialize the LRE database , setup a database user
   ```
   $mysql –u root –p < {dir}/database/db_init.sql
   ```
3. Copy the template for the parameter init file within the config directory:
   ```
   $cp {dir}/app/config/parameters.ini.template
   {dir}/app/config/parameters.ini
   ```
4. Change the parameters by editing parameters.ini.
   ```
   $nano {dir}/app/config/parameters.ini
   ```
5. Install external dependencies:
   ```
   $php {dir}/bin/vendors install
   ```
6. Activate the production mode by changing the 4th line in the file {dir}/web/.htaccess
   From: `RewriteRule ^(.*)$ app_dev.php [QSA,L]`
   To: `RewriteRule ^(.*)$ app.php [QSA,L]`
7. Generate the cache and the assets (css and js files of the bundles):
   ```
   $php {dir}/app/console assets:install web
   $php {dir}/app/console --env=prod --no-debug cache:warmup
   ```

3.2 Configuration

The LRE offers two levels of configuration settings: global, i.e. application-wide, settings and experiment-specific settings.

Global Settings

Global settings can be changed by editing parameters in the configuration file *{dir}/app/config/config.yml*. After changing any global parameter, all files in *{dir}/app/cache* (but not the directory itself) have to be deleted for the changes to take effect. The following table explains all parameters in *{dir}/app/config/config.yml*

Option	Description
sciplore.measurement.levels	An associative array of levels using the yml syntax (comparable to a hash-table)[1]. The key is the numerical value of the level and the value its name. The levels are ordered by their value. The sequence of numerical values for the levels may contain gaps, e.g. one can define the levels 1,2,5,6. However, the scale of the evaluation graph will show all integer values in the interval [min value; max value]
sciplore.validation.confidence.min	The minimum value users can choose to rate their confidence for a document judgment.
sciplore.validation.confidence.max	The maximum value users can choose to rate their confidence for a document judgment.
sciplore.autologin.enable	By default, the system allows automatic user logon without password authentication and redirection to an experiment by using URLs in the form: http://lre.sciplore.org/auto_login?username=test123&experiment=10 If the supplied username does not exist, the system creates a user with the name entered. Automatic user logon can be disabled by setting *sciplore.autologin.enable* to *0*. Setting *sciplore.autologin.create_user* to *0* prevents the system from creating new users, yet allows automatic logon of existing users. Setting *sciplore.authentification.requirePassword* to *1* enables password authentication for existing users Other changes to the logon behavior require editing the file *{dir}/src/sciplore/UserManagementBundle/Controller/AutoLoginController.php*
sciplore.autologin.create_user	
sciplore.authentification.requirePassword	

Experiment-specific Settings

Experiment-specific settings can be changed by adjusting attribute values in the database. Most attributes are contained in the table *experiment* and explained in the following table.

[1] See http://en.wikipedia.org/wiki/YAML#Syntax for a specification of the syntax

Attribute Name	Description
show_small_analysis	The attribute enables (set to 1) or disables (set to 0) a continuously updated bar chart that shows how many points users attributed to recommendations generated with different similarity measures.
randomize_order	The attribute determines if the order of reference documents shown to users is randomized (set to 1) or fixed (set to 0).
debug_view	The attribute determines if management options are hidden (set to 0) or visible to **every user** (set to 1).
result_type_id	The attribute determines the scale used for ratings. The attribute must correspond to an entry in the table *ResultType*. Currently, there are two defined scales available, a nominal scale and an ordinal scale. **Changing the scale only effects new ratings. Existing ratings remain unaltered and should not be analyzed in combination with new ratings.**
visible	The attribute determines if an experiment is shown in the experiment selection screen (set to 1) or not (set to 0).

Adding Additional Similarity Measure to the Evaluation

The database table *method* defines similarity measures available for experiments. Further measures can be defined by inserting rows in this table.

The results analysis of an experiment shows similarity measures only if at least one recommendation in the selected experiment was generated based on that measure.

Changing the Configuration of a Service

Editing the file */Resources/config/services.xml* configures the services of all components. For each service, this file lists the service's unique identifier, e.g. *sciplore.flow_manager*, the full name of the class that implements the service, e.g. *{dir}/sciplore/FlowManagementBundle/Services/FlowManagementService* and a list of all other services that the respective service uses.

Updating a SciPlore LRE Installation

1. Go to the installation directory
   ```
   $cd {dir}
   ```

2. Pull a newer version of the git repository:
   ```
   $git pull origin
   ```
3. Install assets (css and js files of the bundles) and reset the caches:
   ```
   $php {dir}/app/console assets:install web
   $php {dir}/app/console --env=prod --no-debug cache:warmup
   ```

4 References

[1] SensioLab. *Symfony - the Book for Symfony 2.0*. SensioLab, 2012. Available at: http://-symfony.com/doc/current/book/index.html.

Authors: Christian Olms, Norman Meuschke, Corinna Breitinger