The Hybrid Personalizer

The TENCompetence WP7 PDP Planning Tool’s Personalization Service

Software Documentation

Hendrik Drachsler, Eelco Herder, Marco Kalz, Philipp Kärger
Introduction

The Hybrid Personalizer Service is delivering personalized locations of Units Of Learning given a learner’s:

- preference,
- portfolio
- learning goal, and
- current learning state.

It also includes information about the learning object which location is to be computed and information about what other successful learners did. See the following graph to get an idea of how the available information is used.

This documentation contains general information about the single atomic personalization services which are incorporated into the hybrid personalization process. We provide details about how these atomic services are combined. We also include an architecture description of the Hybrid Personalizer. This document completes the java doc available at http://www.L3S.de/~kaeger/HybridPersonalizer/doc . More details about the Hybrid Personalizer are also described in Deliverable 7.2.
The Four Atomic Services

1. Positioning Service

The Positioning Service builds on the assumption that prior learning can be approximated through an analysis of the similarity between learner portfolios and (target) learning material. For this purpose Latent Semantic Analysis is used to calculate the similarity. The concept of the service is shown in the following figure (Kalz et al 2008):

The data layer is responsible for preprocessing the available learning content, eportfolio content and general language corpus documents. In the Similarity Measurement Layer a latent semantic space is build in which the query content, in this case the eportfolio documents, is projected in. The result of the service is a list of learning content units/ learning activities with cosine measures. The output layer produces this list with learning activities and cosine values, while specific thresholds can effect the output list.

The Positioning Service is located in the bottom up cluster of the Personalization Pipeline. In this pipeline the positioning service analyzes portfolio data and projects them into a latent semantic space build from general language data, domain data and the content related to learning activities in learning networks. The result of the positioning service is used by the personalization pipeline to recommend a set of learning activities which fit to the learner’s preferences and prior knowledge. In this case learning activities with a very high cosine similarity are omitted for this learner.
Implementation

The web service builds on the Php Lsa Engine, which is packed with an Easy-Php-Server including everything needed to run the webservice (Apache, MySQL, LSA Engine). The LSA uses the ALGLIB library (Bochkanov & Bystritsky, 2008) for the Singular Value Decomposition. The performance of the Positioning Web Service has been evaluated and optimized by external domain experts on a data set from psychology. We expect that the service needs to be optimized with training data for every implementation. From a technological perspective the web service has been tested regarding reliability and performance. The separation of the construction of the latent semantic space and the query process has improved performance for the service a lot. But the query process takes still around 30 seconds for a test dataset of 800 documents until the service is able to present results (total size 7 MB). This depends to a large extent on the corpus and query size. A demonstration for the Web Service is located at http://145.20.132.193/consum.php

Web Service API

| Calculated similarity between portfolio documents and learning network content |
|---|---|---|---|
| Name | Method | Description | Input (Parameter) | Output |
| getPositionValues | Get | Return a list of UoL annotated with cosine values | Iduser=xx | 2-dimensional array of floats. Each UoL with its calculated cosine values. |
| Frequency | | DATA Fields | Format |
| When visualization tool has to be re-ordered/reorganized. | | Iduser = Integer Learninggoal = Array (Strings) |

References

Bochkanov & Bystritsky, 2008. ALGLIB. Available at http://www.alglib.net/

2. Navigation Service

The Navigation Service is a bottom up service of the Integrated Personalization Service. Therefore, it is part of the Hybrid Personalizer’s bottom up technology cluster (together with the Positioning Service). These are bottom up technologies because they require no top down maintenance of metadata thus they rely on information from the learners. Beyond the specification of Deliverable 7.1, we added an additional function to the service API called getNextStepPropability. For every Unit of Learning (UoL) the Navigation Service calculates its relationship to other UoLs in the Learning Network in order to provide measures for the Visualisation Tool to place the UoLs on the screen. Therefore, it observes the behavior of the learners and their usage of the UoLs in the learning network. Every successful completion of a UoL is saved in a transition matrix. Based on this matrix the Navigation Service provides a
measure between UoLs that represents their relationships between each other according to the
learning behavior of the learners.

Implementation
The web service is created with PHP5 on an Apache 2 web server. The transition matrix is saved
in a MySQL 4 database. PHP5 has an integrated web service class that enables us to make the
Navigation Service’s results available for the Personalization Interface. A demo for the Web
Service is available at http://lnx-otecexp-004v.ou.nl/navigation/test/consum2.php
The performance of the Navigation service has been evaluated within an experiment with
students of a psychology course (Drachsler et al. 2008). From a technological perspective the web
service has been tested regarding reliability and performance with the WP partners.

Web Service API
The Navigation Service API still relies on the specification we made in the deliverable 7.1. For
the current prototypical Visualisation Tool we did not implement all functions into the service.
For a future integration of the service into the PCM minor adjustments have to be done in order to
feed the service with required data from the 10C infrastructure.
In the following we give a specification of the additional function called getNextStepProbability
we added to the Navigation Service.

<table>
<thead>
<tr>
<th>Navigation Web Service Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate relationship between UoLs based on behaviour of the learners</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Description</th>
<th>Input (Parameter)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNextStepProbability()</td>
<td>GET</td>
<td>Request a relationship description for a current UoL</td>
<td>UoL_ID = xx UoLs=xx</td>
<td>2 dimensional Array of Integers</td>
</tr>
<tr>
<td>Frequency</td>
<td>DATA Fields</td>
<td>Frequency of the UoLs from the Navigation Service</td>
<td>Format</td>
<td></td>
</tr>
<tr>
<td>Whenever the Visualisation Tool have to be reorganised according to decisions of the learner</td>
<td>UoL</td>
<td>Integer</td>
<td>UoLs</td>
<td>Array of Integer</td>
</tr>
</tbody>
</table>

References

3. Preference-based Personalization Service
The preference-based personalization service refers to the preferences specified by the learner (cf.
Deliverable 7.1) in order to provide meaningful personalized locations of learning objects:
preferred learning objects get location numbers close to zero while non-preferred ones get higher
values. The service can be configured according to two preference semantics: the qualitative Pareto semantics and a quantitative semantics.

**Qualitative semantics.** Learning objects that are among the first skyline (i.e., Pareto optimal) are given the location zero. The second skyline, i.e., the learning objects that or optimal among all learning objects except the first skyline, are given higher values. This approach corresponds to the qualitative semantics proposed by Pareto and exploited for so-called skyline querying. However, this semantics is very strict. For preferences over many dimensions it is known to include almost all objects, that is, all learning objects are optimal according to high-dimensional preferences. To overcome this problem, we introduced a second semantics in our implementation: a quantitative one.

**Quantitative semantics.** The quantitative semantics computes the location of a learning object by counting the amount of attribute values that the learner considers better (in terms of the defined preference order) than the actual value of the learning object. If all attributes of a learning object are optimal, it is given the location value zero. For each attribute that bears a non-optimal value, this number is increased dependent on the rank where this value appears in the preference order given by the user.

![Figure. How preference information is used to distribute the surrogates over the x-axis.](image)

The specification stating which semantics used by the preference-based personalization service is configurable by means of the configuration framework integrated into the Hybrid Personalizer.

### 4. Curriculum-based Personalization Service

Based on the pre- and post-conditions given for each learning object, the curriculum-based personalization service is able to induce information about which learning object is to be visited
by the learner at what state of the learning process. For this the service exploits information about
the learner’s current knowledge and information about the learner’s learning goal, i.e., the
knowledge the learner is aiming at with the current learning process. The information provided by
this service for the PDP Planning tool is twofold: first, the curriculum-based service is able to
compute a Learning Path (LPath) guiding the learner from her current state to her learning goal;
and second, the service is able to compute locations of the learning object’s surrogates given the
amount of learning objects in the learning space forming potential prerequisites for a learning
object.

**Learning Path generation.** Given a set of learning objects, the learner is enabled to request for
an appropriate LPath. The computation of the path follows the algorithmic curriculum planning as
it described in Deliverable 7.1. For the communication with the graphical user interface, a
Learning Path class has been implemented that is used for both, for the intelligent generation of
the Learning Path in the service as well as for the visualization in the PDP Planning Tool.

**Curriculum-based computation of locations.** Although the learner may not yet requested for a
Learning Path, the service can be used to locate learning object surrogates in a personalized way.
The learning objects that are close to the learner’s current knowledge according to any potential
Learning Path are given the location value zero. Based on these most appropriate next steps and
by following all potential paths the service is able to generate, the locations for potential follow-
up learning objects are computed.

![Diagram](image)

Figure: Computation of locations and generation of a Learning Path out of the
preconditions (below the icons) and the post-conditions (above) of the Units of Learning.
Hybrid Personalizer
An Integrated Personalization Service

The atomic Personalization Services as they are introduced up to here build up the basis for the advanced integrated Personalization Service called Hybrid Personalizer. Each of the atomic services provide the PDP planning tool with complementary information about which Unit of Learning suits a learner’s needs in her current situation. Combining these complementary input values is the challenge the Hybrid Personalizer is dealing with. The four atomic services can be divided into two bottom up approaches, namely the navigation and positioning Web Services, and into two top-down approaches, namely Preference-based and Curriculum-based personalization.

Figure: Architecture of the Hybrid Personalizer integrating the four Atomic Services (orange boxes).

The atomic services together with the integrated service on the one side and the graphical user interface of the PDP Planning Tool on the other are separated conceptually. That is, java interfaces as well as Web Service interfaces ease the communication between the components and allow for an ad-hoc adoption to new data or systems (e.g., the forthcoming integration into the PCM). Moreover, a configuration component allows a fine-grained tuning and adoption of
how the returned values of the atomic services are used to compute a single personalization value. By this means, the strategy of the hybrid personalization can be modified easily.

In the following we provide more detailed but still conceptual information about the Hybrid Personalizer’s implementation. For more details we refer to the TENCompetence evs WP7 Module and to the Java-doc for the implementation to be found at http://www.l3s.de/~kaerger/HybridPersonalizer/doc/.

The **PersonalizationService interface**

The task of the Hybrid Personalizer is to combine the complementary personalization services in a way that information is generated that is exploitable to place the learning object’s surrogates in a personalized way. Therefore, we offer the java interface PersonalizationService providing the following methods:

- **List<Double> getLocation(String UoLID, String learnerID)** returns a pair of double values. Each coordinate is a value between 0 and 1. This pair represents the location of the learning object’s surrogate in the Graphical User Interface of the PDP Planning Tool.
- **List<List<Double>> getLocations(List<String> unitOfLearningIDs, String learnerID)** returns a list of pairs. The order of the elements in the returned list corresponds to the order of the provided list of Unit Of Learning IDs.

By means of these two methods any component is able to retrieve locations of surrogates that correspond to their appropriateness for the learner is her current situation. In order to call the atomic services in a unique way, we introduced an interface that is implemented by the atomic services or their wrappers, namely the **AtomicPersonalizationService interface**.

The **AtomicPersonalizationService interface**

Each of the four atomic services deliver their data via arbitrary technology. In order to easy their integration and to allow to simply plug-in potential new atomic services, we provide the AtomicPersonalization interface. It is implemented by the wrapper classes for each of the four services and provides to following methods.

- **Double getLocation(String UoLID, String learnerID)** returns a single double value between 0 and 1. This value represents how appropriate is the given Unit of Learning for the given learner.
- **List<Double> getLocations(List<String> unitOfLearningIDs, String learnerID)** returns a list of double values. The order of the double values in the returned list corresponds to the order of the provided list of Unit of Learning IDs.
- **void init(Set<String> unitOfLearningIDs)** initializes the atomic service. This method is also used to update the service if some metadata of the learner or of a Unit of Learning changed. For some services that implement a cache, calling this method may be needed if a new Unit of Learning is introduced.
The Wrappers for the atomic services

Each Personalization Service is wrapped by a class implementing the AtomicPersonalizationService interface.

The class NavigationBasedService forms the wrapper for the Navigation Web Services. It is implemented as an Axis client and communicates with the actual Navigation Web Service via SOAP. This class scales the values returned by the Navigation Web Service to a value between 0 and 1.

The class PositioningBasedService is the wrapper for the Position Web Service. Similar to the NavigationBasedService class it consists of an axis Web Service client able to communicate via SOAP. Again, this wrapper scales the output values of the Positioning Web Service to values between 0 and 1.

The class PreferenceBasedService is responsible for computing locations based on preference information. It implements the AtomicPersonalization-Interface. The getLocation-method returns a Double value ranging from 0 to 1. This number represents how appropriate the given unit of learning is according to the preferences of the given learner.

The class CurriculumBasedService wraps the implementation of the algorithmic curriculum planner. It exploits the information delivered by the planner to compute numeric values for learning objects. These values correspond to the position of the given Unit of Learning in any learning path that starts at the learner’s current knowledge.

The class FairXValueDistributor. We provided a fifth implementation of the AtomicPersonalizationService, namely the FairXValueDistributor class. This implementation of an atomic personalization service does not take any information about learners and Units of Learning into account. For the computation of a double value location on the x-axis only the amount of Units of Learning with the same y-axis values are considered. The x-values are computed in a way that all these Units of Learning are distributed over the x-axis equally. This is of particular usefulness if no information about learner preferences or about other learner’s steps (as needed for the navigation service) is exploitable for computing an x-axis value.

The Learning Path Provider interface

The LearningPathProvider interface offers two methods that deliver Learning Paths. One method provides a learning path given information about the learner (her current knowledge and learning goal). The other receives a learning path ID and returns the corresponding learning path.

- LearningPath computeLearningPath(Set<String> UoLs, Set<String> knowledge, Set<String> goal) computes a learning path for achieving the competences in the given goal (a set of competences) by assuming a learner’s knowledge as it is defined in the set knowledge (another set of competences).
- LearningPath getDefinedLearningPath(String learningPathID) returns a predefined learning path, for example a previously stored learning path from the TENCompetence server.
Currently, we provide one implementation of this interface called CurrBasedLearningPathProvider. The Curriculum-based learning path provider makes use of the algorithmic curriculum planning tool to compute a learning path.

**The configuration framework**

Merging the output of the four atomic services in order to provide a single personalized view on the learning space is a challenging task. There are arbitrary many ways of combining the output of the atomic services. This may depend on the metadata available for computing the personalized locations. Dependent on the data available one may also want to put different weights to the results of the atomic services, leave some of them out, or even let the user decide how to configure the Hybrid Personalizer.