Concept Lattices for Federated Learning Systems

SYLVIA ENCHEVA
Stord/Haugesund University College
Department Haugesund
Bjørnsonsg. 45, 5528 Haugesund
NORWAY

SHARIL TUMIN
University of Bergen
IT-Dept.
P. O. Box 7800, 5020 Bergen
NORWAY

Abstract: This paper focuses on a framework for building new courses or updating existing ones by choosing learning objects developed at universities that are members of a federated learning system. The aim of this work is twofold. First assisting a lecturer in collecting learning objects closest to the lecturer's vision on what a subject should contain and how the content should be presented. Secondly, present a student with content, tailored according to student's individual learning preferences.

Key–Words: Learning objects, concept lattices, federated systems

1 Introduction

A variety of learning technology systems and interoperability standards providing reuse of learning objects (LOs) and interoperability of content across delivery are developed ([21], [15], and [12]). While most efforts aim at providing a technology to access and share existing learning objects, there is yet no formal model of how to filter the most suitable LOs for a subject.

Let us consider a lecturer affiliated with an educational institution that is member of a federated learning system. Suppose the system is able to provide a large number of LOs upon the lecturer request. A lot of time and efforts can be spared if the system can first filter and rank those LOs according to the lecturer's preferences. Another important issue is how to build a course supporting student's individual learning preferences. The system can help both the lecturer and the students by presenting each student with a LO chosen from the related set of selected LOs but tailored according to the student's individual learning styles and preferences. Identifying a student's style and then providing instruction consistent with that style contributes to more effective learning [10].

Our goal is to develop a framework of how to enable a lecturer to build new courses and/or update existing ones by reusing LOs developed at universities that are members a federated learning system.

The paper is organized as follows. The system's framework is described in Section 4. Presenting students with LOs corresponding to learning preferences is discussed in Section 3. The paper ends with a conclusion in Section 6.

2 Related Work

SCORM [21] provides technical standards that enable web-based learning systems to find, import, share, reuse, and export learning content in a standardized way. However, SCORM is written for toolmakers who know what they need to do to their products to conform with SCORM technically. IEEE Learning Object Metadata [15] defines a set of resource description framework constructs that facilitates introduction of educational metadata into the semantic web. HarvestRoad Hive [12] is an independent, federated digital repository system. It enables the collection, management, discovery, sharing and reuse of LOs used in the delivery of online courses within higher education.

Top-down and bottom-up strategies for the development of tightly coupled, federated information systems are presented in [3]. Information integration, such as intention of treating different kinds of heterogeneity, preserving source autonomy and enabling change management while ensuring consistency is further discussed.

An agent-oriented method and a generic agent-based architecture for the development of Cooperative Information Systems (CISs) is presented in [5]. The proposed method allows mastering the complexity of the cooperation processes and the difficulty of setting up effective CISs, by the analysis and the modelling of the cooperation in two levels of abstraction.

A reference model and infrastructure for federated learning content repositories are developed in [18]. It is shown how to provide access to learning content, under the base assumption that good learning requires ubiquitous content, which in turn implies the need for an operational content infrastructure.

Uncertainties in the database integration process were analyzed in [1] and [2]. Engineering federated information systems are discussed by many authors, e.g. [9], [14], [14], [19], [20], and [23].

3 Concept Lattices, Unfolding Model, Learning Styles and Preferences

3.1 Concept Lattices

A *concept* is considered by its *extent* and its *intent*: the *extent* consists of all objects belonging to the concept while the *intent* is the collection of all attributes shared by the objects [6].

A *context* is a triple (G, M, I) where G and M are sets and $I \subset G \times M$. The elements of G and M are called *objects* and *attributes* respectively.

For $A \subseteq G$ and $B \subseteq M$, define

$$A' = \{m \in M \mid (\forall g \in A) \;\; gIm\},$$

$$B' = \{g \in G \mid (\forall m \in B) \;\; gIm\}$$

so A' is the set of attributes common to all the objects in A and B' is the set of objects possessing the attributes in B. Then a *concept* of the context (G, M, I) is defined to be a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The *extent* of the concept (A, B) is A while its intent is B. A subset A of G is the extent of some concept if and only if $A'' = A$ in which case the unique concept of the which A is an extent is (A, A'). The corresponding statement applies to those subsets B of M which are the intent of some concept.

The set of all concepts of the context (G, M, I) is denoted by $\mathcal{B}(\mathcal{G}, \mathcal{M}, \mathcal{I})$. $\langle \mathcal{B}(\mathcal{G}, \mathcal{M}, \mathcal{I}); \leq \rangle$ is a complete lattice and it is known as the *concept lattice* of the context (G, M, I).

For concepts (A_1, B_1) and (A_2, B_2) in $\mathcal{B}(\mathcal{G}, \mathcal{M}, \mathcal{I})$ we write $(A_1, B_1) \leq (A_2, B_2)$, and say that (A_1, B_1) is a *subconcept* of (A_2, B_2), or that (A_2, B_2) is a *superconcept* of (A_1, B_1), if $A_1 \subseteq A_2$ which is equivalent to $B_1 \supseteq B_2$ [6].

3.2 Unfolding Model

The unfolding model was devised for the analysis of ranking data based on preferential choice behavior [4].

According to the unidimensional unfolding model, preferential choice is made in the following manner: an individual evaluates m objects based on the objects' single common attribute. Each object is represented by a real number expressing the level of this attribute $x_i, i = 1, ...m$ or a point on the real line $\mathcal{R}$ (the 'unidimensional underlying continuum'). Each individual is also represented by a point $y \in \mathcal{R}$ on the same line. The point y is considered the individual's favorite and is called the individual's ideal point. The real line $\mathcal{R}$ containing both individual's and objects is thought of as the psychological space. Individuals and objects are identified with their corresponding points. The model assumes that individual y ranks the m objects $x_i, i = 1, ...m$ according to their distances from y, i.e., individual y prefers x_i to x_j if and only if $|y - x_i| < |y - x_j|$.

3.3 Learning Styles, Orientations and Preferences

A learning style is the general, habitual mode of processing information; it is a predisposition on the part of some students to adopt a particular learning strategy regardless of the specific demands of the learning task: that is, individuals' learning styles are simply the cognitive styles that they evidence when confronted with a learning task [22].

According to Kolb's model [13], there are four learners types - concrete, reflective; abstract, reflective; abstract, active; and concrete, active.

The three learning preferences are auditory (learning by hearing), visual (learning by seeing), and kinesthetic (learning by doing) [7].

3.4 Learning Orientations

Student learning orientations [16] are critical for individualizing the instructional process. The system provides different tests with respect to different learning orientations.

Each level has tutorial material generated for it; since it is important to target tutorial tasks at the student's ability, this is seen as being of more educational benefit than offering the same tutorials to all students and then assigning a student to a level based upon the grade the student achieves [11]. It is incorporated by including different help functions. Intelligent agents provide different students with different pages according to their needs. Additional explanations and examples helping to clear current difficulties and misconceptions are provided without use of human tutors.

There are four learning orientations [17]:

- `Transforming learners` - They place great importance on personal strengths, ability, persistent effort, strategies, high-standards, and positive expectations to self-direct intentional learning successfully. They prefer loosely structured learning environments.

- `Performing Learners` - They are non-risk, skilled learners that consciously, systematically, and capably use cognitive processes, strategies, preferences as they focus on grades and attaining normative achievement standards. They prefer semi-structured learning environments.

- `Comforming Learners` - They are compliant and more passively accept knowledge, store it, and reproduce it to conform, complete assigned tasks if they can, and please others. They prefer to have simple standards set for them and receive explicit guidance and feedback.

- `Resistant Learners` - They lack a fundamental belief that academic learning and achievement can help them achieve personal goals or initiate positive change. These learners do not believe that formal education or academic institutions can be positive or enjoyable influences in their life.

4 Framework

The system is based on a flexible framework that provides the instructors and students with possibilities in designing courses using shared, reusable LOs.

Course design costs a significant part of a WEB-based learning project budget. A big cost reduction is possible if as many LOs as possible are reusable and are shared among collaborative organizations. Instructors can just define which LOs should be included in their courses. This will make the course building more flexible and reduce costs for both students and organizations.

A subject is defined as a strictly sequential list of topics by a course builder. Each subject definition is saved in the database in XML. The subject's definition contains all the metadata needed to uniquely describe the subject in question. These metadata include server, domain, title, topics list, and other operational parameters.

A topic is composed of theoretical parts, exercises, examples, drills, and assessments, where some of them can be empty. Any of them is a LO designed by expert instructors. These LOs are WEB documents (HTML, PDF, Flash, Java-applet, WEB-form, etc.).

Each topic is defined by the course builder in XML and saved in the database. The topic's definition contains all the metadata needed to describe the topic content, drill, and assessment policies. These documents can be used to provide learners with dynamic HTML pages for each invocation of a topic depending on its LO's set.

Table 1: Context for different universities

	(T)	(E)	(I)	(D)	(A)
U1b	×	×		×	×
U1m	×	×			
U1p	×		×		×
U2b					
U2m	×		×		×
U2p	×				×
U3b					
U3m	×		×	×	
U3p	×	×			×

Topics are defined dynamically by diagnostic components of the system. The drills' design contains inference rules analyzing students' answers to carefully prepared tests. Students' test scores and wrong responses are used as input parameters to a diagnostic component in the drills, which provides recommendations to alternative personal topic trails containing relevant LOs in order to solve the current topic's drill.

Assessments are used for both formative and summative evaluations. Exercises give a list of unsolved problems for students to practice on, while examples can be a list of solved problems, Flash, or Java-applets for students to work with. Since each LO is self-contained, they can be developed independently at different organizations. Expert instructors at each organization maintain a repository of LOs. For students at different collaborative organizations, they will have the advantage that all LOs are accessible to them. For organizations owning the LOs, an audit trail for each LO can easily be gathered for billing purposes.

For the sake of simplicity we limit the amount of universities, study programs and related attributes to the ones included in Table 1.

The abbreviated objects are -
University 1 Bachelor level (U1b),
University 1 Master level (U1m),
University 1 PhD level (U1p),
University 2 Bachelor level (U2b),
University 2 Master level (U2m),
University 2 PhD level (U2p),
University 3 Bachelor level (U3b),
University 3 Master level (U3m),
University 3 PhD level (U3p).

The abbreviated attributes are -
Theory (T) - HTML, PDF, Flash, Java-applet, WEB-form
Exercises (E) - HTML, PDF, Flash

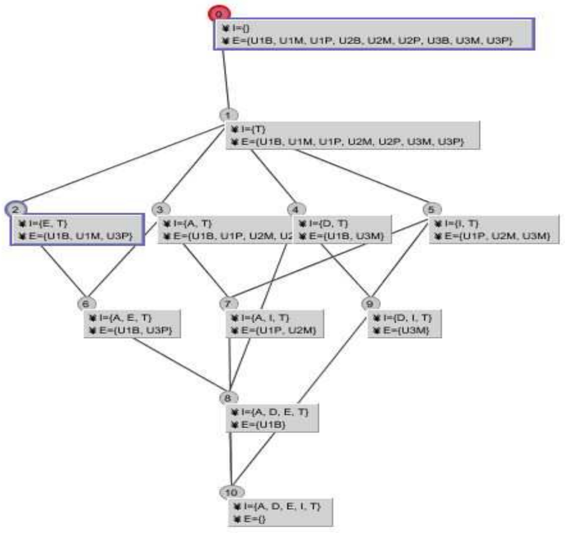

Figure 1: Hasse-diagram for different universities

Interactive examples (I) - Flash, Java-applet
Drills (D) - WEB-form
Assessment (A) - WEB-form.

The corresponding concept lattice is shown in Fig. 1.

5 System Architecture

The framework is composed of base system, runtime support, and agents.

`Base System`

Apache is used as a Web server with a Python interpreter extention using a mod_python module. PostgreSQL is used as a relational database that supports Structured Query Language (SQL).

Python is an object-oriented scripting language. Having a Python interpreter module in the Web server increases performance and reduces response time significantly. A standard Apache implementation with Common Gateway Interface (CGI) scripts can also be used. However, this will lead to a performance penalty in that the Web server needs to start a new external Python interpreter each time a CGI script is called.

A relational database management system is used to store tests and students' data. PostgreSQL provides us with database support for flexible implementation. All data is stored in a relational database and can be queried programmatically by Python scripts using SQL. Other relational databases such as MySQL and Oracle can be used instead. We have developed contents of learning materials on a Linux workstation.

`Runtime support`

Dynamic HTML pages are created by server-side scripts written in Python. Python programs are also used for database integration, diagnostics, and communication modules.

`Agents`

The system is supported by student profile, test, and diagnostic agents which we will discuss further in this paper.

The system structure is defined by pedagogical requirements. This structure defines dependencies among learning materials, levels and relationships between tests options, and inference rules used in a diagnostic agent. This structure is crucial in providing each student with a personalized learning work-flow for efficient learning. The system provides different students with different pages according to their needs. The responses from each student to the suggestions from the system provide the diagnostic agent with necessary data. The diagnostic agent analyzes these data using the programmed inference rules and provides the student with an immediate recommendation on how to proceed. The student status is saved in the database.

It is based on each individual's learning styles, orientations and preferences, and chooses the most appropriate LOs for the course, (see Fig.2). The selection process was discussed in Section 4. An agent is first checking whether all definitions and statements required are included in the suggested LOs. Another agent determines whether the level of difficulties assumed for the new LOs corresponds to the level of difficulties of the suggested LOs. The unfolding model (shortest distance) (see 3.2) is used while comparing different LOs. The level of difficulties is judged based on the included theory, examples and assessment tests.

A questionnaire is put to the students for determining their individual learning preferences (see 3.3). In the recommendation on how to proceed, a student can choose to subscribe to one or more suggested learning materials. The student's learning material subscriptions are placed in a stack-like structure in the student profile data. The student profile agent presents the student with the top most learning material in the profile stack for each new learning session.

Initially, the profile stack contains a sequential ordering of learning materials in a given subject. A student can choose to skip any presented learning material and go to the next one at any time. A student is considered to have completed a course when his/her profile stack is empty and he/she has passed all compulsory tests assigned to the course.

All learning material names taken by the student

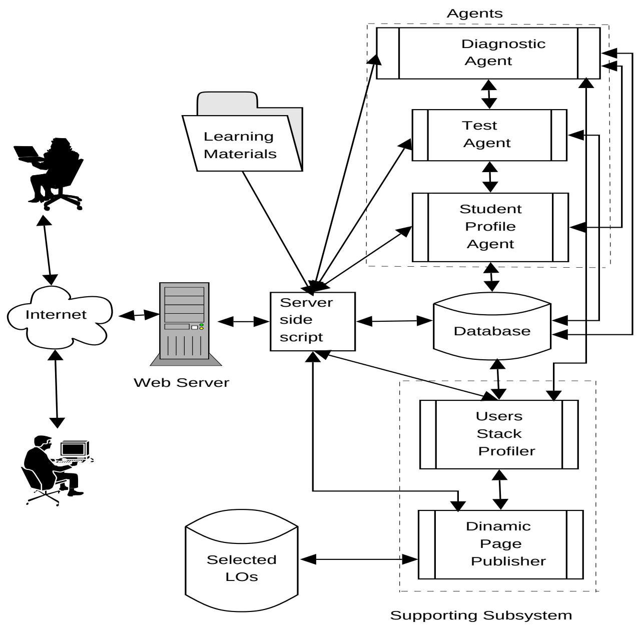

Figure 2: System architecture

during the course and scores of the tests are saved in the student audit-trail. Such audit-trail data is used for billing purposes while global analysis of the course and feedback data is used to improve learning materials for each subject.

The server-side script (SSS) contains student and teacher modules. The student module contains, among other things, student registration, student administration, student authentication, and authorization. The teacher module provides an interface for a teacher to define topics and subjects, students' status reports/diagnostic, and messaging.

SSS modules and agents communicate with each other by a request-respond mechanism in which remote procedure calls are done among different modules/agents providing students with a dynamic and personalized learning environment.

A pedagogically crafted scheme with a set of questions and answers provides a test agent with a programmed intelligence, in which wrong answers to a drill lead learners to appropriate hints and examples. The students can then subscribe to those learning materials suggested by the hints or try the drill again and continue with the current course. Students can jump back to the current course trail at any time while following trails suggested by the test agent. The agent calculates scores, shows result status, and keeps track of assessments taken by each student. After each assessment the test agent sends summarized information to the diagnostic agent.

5.1 Supporting Subsystem

Stack Profiler

In the recommendation on how to proceed, a student can choose to subscribe to one or more suggested LOs. The student's LO subscriptions are placed in a stack-like structure in the student profile data. The system presents the student with the top most LO in each new learning session.

Initially, the profile stack contains a sequential ordering of LOs in a given subject. A student can choose to skip any presented LOs and go to the next one at any time. A student is considered to have completed a course when her/his profile stack is empty and she/he has passed all compulsory tests assigned to the course.

All element names taken by the user during the course and scores of the tests are saved in the user audit-trail. Such audit-trail data is used for billing purposes while global analysis of the course and feedback data is used to improve contents and tests for each subject.

Policy

The curriculum of each subject at every university should be described using a set of agreed upon metadata presented in a standard structure in a database, for example a name space of a LO in a database.

```
University:Faculty:Level:Course-name:
Learning-object:Expire-data
```

LO Caching

- If LOs in a course are connected with hyperlinks, the course builder risks to end up with some dead links during the semester.

- If all LOs in a course are cached on a local server, the course builder is sure that all LOs are going to be available to the students through the entire semester. The owners of the LOs have no control over the amount of students and number of times those LOs are used. However, the owners of the LOs can include f. ex. 1×1 pixel gif picture in every LO. Thus the owners will get information from log files for the number of times a LO has been used and by how many different users.

6 Conclusion

This paper describes a framework for building new courses or updating existing ones by choosing learning objects developed at universities that are members a federated learning system. The proposed system has been exploited by four high level educational institutions. Both course builders and students expressed satisfaction using it. Eventhough LOs are indexed with metadata which simplifies the process of identifying, locating and retrieving them, additional work is needed for further improvement of the automated selection of LOs.

References:

[1] E. Altareva, E., S. Conrad, The Problem of Uncertainty and Database Integration Engineering Federated Information Systems, *Proceedings of the 4th Workshop EFIS 2001*, Berlin, Germany 2001

[2] E. Altareva, E., S. Conrad, Analyzing Uncertainties in the Database Integration Process by Means of Latent Class Analysis, Engineering Federated Information Systems, *Proceedings of the 5th Workshop EFIS 2003*, Coventry, UK 2003

[3] S. Busse S., R-D. Kutsche and U. Leser, Strategies for conceptual design of federated information systems, *Lecture Notes in Computer Science* 1626, 1999, pp. 255–269.

[4] C. H. Combs, A theory of data. New York, Wiley 1964

[5] B. Djamel, B. Zizette and B. Mahmoud, From the Analysis of Cooperation within Organizational Environments to the Design of Cooperative Information Systems: an Agent-Based Approach, *Proceedings of International Workshop on Modeling Inter-Organizational Systems (MIOS)*, Cyprus 2004

[6] B.A. Davey and H.A. Priestley, Introduction to lattices and order, *Cambridge University Press*, Cambridge 2005

[7] R. Dunn, K. Dunn and G. Price, Manual: Learning style inventory, *Lawrence, KS: Price Systems* 1985

[8] B. Ganter, G. Stumme and R. Wille, Formal Concept Analysis - Foundations and Applications, *Lecture Notes in Computer Science* 3626, 2005

[9] A. James, S. Conrad and W. Hasselbring, Engineering Federated Information Systems, *Proceedings of the 5th Workshop EFIS 2003*, Coventry (UK), Akad. Verlagsgem. / IOS Press 2003

[10] D.H. Jonassen and B.L. Grabowski, Handbook of Individual Differences, Learning, and Instruction. *Mahwah, N.J.*: Erlbaum. 1993

[11] J.R. Hartley, Interacting with multimedia, *University computing* 15, 1993, pp. 129–136

[12] http://www.harvestroad.com/

[13] D.A. Kolb, Experiential Learning: Experience as the Source of Learning and Development, Englewood Cliffs, NJ: *Prentice-Hall* 1984

[14] R.-D. Kutsche, S. Conrad, W. Hasselbring (eds.), Engineering Federated Information Systems,*Proceedings of the 4th Workshop EFIS*, Berlin, 2001

[15] http://kmr.nada.kth.se/el/ims/md-lomrdf.html

[16] M. Martinez and C. V. Bunderson, Building interactive Web learning environments to match and support individual learning differences, *Journal of Interactive Learning Research* 11(2), 2000, pp. 163–195.

[17] M. Martinez, Key design considerations for personalized learning on the Web, *Educational Technology & Society* 4(1), 2001, pp. 26–40.

[18] D. Rehak, P. Dodds and L. Lannom, A Model and Infrastructure for Federated Learning Content Repositories, *14th International World Wide Web Conference (WWW2005)*, Chiba, Japan, 2005

[19] M. Roantree, W. Hasselbring and S. Conrad (eds.), Engineering Federated Information Systems, *Proceedings of the 3rd Workshop EFIS 2000*, Dublin, Ireland, Akad. Verl.-Ges. Aka, Berlin / IOS Press, Amsterdam, 2000

[20] M. Roantree, J.B. Kennedy and J. Barclay, Using a metadata software layer in information systems integration, *CAiSE 2001*, LNVS 2068, 2001, pp. 299–341.

[21] http://www.adlnet.org/index.cfm?fuseaction=scormabt

[22] R. Schmeck, Learning Strategies and Learning Styles. New York: *Plenum Press*, 1988

[23] K. Taylor and J. Murty, Implementing Role Based Access Control for Federated Information Systems on the Web, *AISW 2003* 21, 2003