Wang, T. I., Tsai, K. H., Lee, M. C., & Chiu, T. K. (2007). Personalized Learning Objects Recommendation based on the Semantic-Aware Discovery and the Learner Preference Pattern. *Educational Technology & Society*, 10 (3), 84-105.

Personalized Learning Objects Recommendation based on the Semantic-Aware Discovery and the Learner Preference Pattern

Tzone I Wang, Kun Hua Tsai, Ming Che Lee and Ti Kai Chiu
Laboratory of Intelligent Network Applications, Department of Engineering Science, National Chung Kung University, Tainan, Taiwan // wti535@mail.ncku.edu.tw // tsaikunhua@msn.com // apple@system.es.ncku.edu.tw

ABSTRACT

With vigorous development of the Internet, especially the web page interaction technology, distant E-learning has become more and more realistic and popular. Digital courses may consist of many learning units or learning objects and, currently, many learning objects are created according to SCORM standard. It can be seen that, in the near future, a vast amount of SCORM-compliant learning objects will be published and distributed cross the Internet. Facing huge volumes of learning objects, learners may be lost in selecting suitable and favorite learning objects. In this paper, an adaptive personalized recommendation model is proposed in order to help recommend SCORM-compliant learning objects from repositories in the Internet. This model adopts an ontological approach to perform semantic discovery as well as both preference-based and correlation-based approaches to rank the degree of relevance of learning objects to a learner's intension and preference. By implementing this model, a tutoring system is able to provide easily and efficiently suitable learning objects for active learners.

Keywords

Learning object, Personalized recommendation, Learner pattern, Semantic discovery, SCORM, LOM

1. Introduction

With energetic development of the Internet, especially on the web page interaction technology, distant e-learning systems have become more and more realistic and popular in the past ten years. Most of the currently running e-learning platforms are closed systems that have their own digital materials. These materials are hardly shared and re-used by different systems because they are made in proprietary formats. To solve the problems of sharing and reusing teaching materials in different e-learning systems, many international organizations, including IEEE Learning Technologies Standardization Committee (LTSC), Instruction Management System Global Learning Consortium (IMS), Aviation Industry CBT committee (AICC), Advanced Distributed Learning initiative (ADL) (ADL-1, 2001; ADL-2, 2001; ADL, 2003), and Alliance of Remote Instructional Authoring and Distribution Networks for Europe project (ARIADNE, 1998), have devoted to establishing e-learning standards. Among many proposed standards, the Sharable Content Object Reference Model (SCORM) (SCORM, 2003) is recognized as the most popular one and its Learning Object Metadata (LOM) (LOM, 2002) has been approved by the IEEE-Standards Association. The LOM aims to provide structured descriptions of digital contents, sometimes called "Learning Objects", a standard for encoding general learning object information including title, author, right, date, relations, and educational objectives. With the standard protocol, teaching materials made for different learning management systems can be shared, reused, and integrated; more and more learning objects will be authored and shared across the Internet in the near future. Before cooperative e-learning platforms with reusable and shareable learning objects are operational, issues below must be solved.

- **How to collect reusable learning objects on the Internet**. Generally, search engines, such as Google or Yahoo!, can help users search for learning objects. Keywords must be entered into these engines first but users are usually beset with what keywords to enter because they have not studied a specific domain yet. An ontology based query expansion and recommendation system can be used to help people use right keywords in finding suitable learning objects.
- **How to filter out unsuitable results as many as possible**. In general, the result of a query to a search engine is ranked by the degree of similarity to the keywords and uses a general basis for all the users. Search engines give no adaptive and personalized recommendations for each user. For a tutoring system, the ranking strategy should put the learning objects that are most conformed to a user's preference and intention at front places.
- **How to identify related learning objects for a specific course**. Because a learning object may be specifically made for a course, when reusing the object, it's necessary to know what the learning object is made of and what course it is suitable for. LOM is due to easy this problem. A course author has to define what learning objects

are needed for the course and, once this is established, these objects should be found from some repositories or the Internet.

This paper proposes a Personalized Learning Object Recommendation Model (LORM) that adopts ontological approach to guide a tutoring system in providing learning objects for a user according to the user's needs. Through the ontological approach, semantics of user entered query keywords can be conjectured and the expansion system may, according to the ontology inference, add additional keywords to the user's original ones to construct a more complete semantics. In this way, more suitable learning objects (i.e. learning objects a course may need according to a learner's intention) can be discovered and retrieved. A personalized recommendation mechanism is also proposed, which uses two algorithms, the *Preference-based algorithm* and the *Correlation-based algorithm*, to rank the recommended results to advise a learner with the most suitable learning objects. A system realizing this model can:

1. Use the specific ontology to infer what learning objects are needed for a course established for a specific learner requiring a specific subject.
2. Build each learner's personal preference pattern according to past studying histories.
3. Recommend suitable learning objects according to a learner's preference and intention.
4. Refer to the experiences of similar learners when looking for learning objects that should be helpful for each learner.
5. Provide adaptive, personalized learning objects and materials for each learner.

2. Background and Problems

A learning object is a basic component or unit of a course in a tutoring system. Some definitions for learning objects are summarized as follows:

- "Modular digital resources uniquely identified and meta-tagged, that can be used to support learning." -- *National Learning Infrastructure Initiative*
- "The main idea of learning objects is to break educational content down into small chunks that can be reused in various learning environments, in the spirit of object-oriented programming." -- *David A. Wiley*
- "Any entity, digital or non-digital, that may be used for learning, education or training." -- *IEEE 1484.12.1-2002, 15 July 2002, Draft Standard for Learning Object Metadata, IEEE Learning Technology Standards Committee (LTSC)*

2.1 SCORM

When a learning object is a self-contained component, it is natural to reuse the object in many courses. However that needs a common standard for tutoring systems to follow. **SCORM** is implemented by the Advanced Distributed Learning (ADL) Initiative created by the US department of defense. Its purpose is to foster creation of reusable learning content used as "instructional objects" within a common technical framework for both computer and web-based learning. SCORM describes this technical framework by providing a harmonized set of guidelines, specifications, and standards developed by several organizations. All of them are adapted and integrated with one another to form a more complete and easier-to-implement mode by ADL (ADL-1, 2001). Key contributors to SCORM include organizations AICC, ARIADNE, IEEE LTSC, and IMS.

2.2 Learning Object Metadata – LOM

LOM (LOM, 2002) is the set of attributes needed to allow Learning Objects to be managed, located, and evaluated. This standard uses a pre-defined and common vocabulary to describe the content of learning objects learning object content. There are nine categories in the vocabulary, including General, Lifecycle, Meta-metadata, Technical, Educational, Rights, Relation, Annotation, and Classification. They are shown as in Table 1 with short descriptions. The LOM fulfills the same role as "Dewey Decimal Classification" in library books catalog. If learning objects in a repository on the Internet are catalogued according to LOM, it would be easy to identify a particular learning object. In the libraries catalogs rapidly guide a user to locate books of interest, while on the Internet LOM should guide a learner to locate suitable learning objects. When to understand the preference of a learner, we can extract the field values of the learning object from these nine categories and regards them as the major features of the learning object.

2.3 Recommendation Model Survey

In traditional teaching scenarios, a teacher may usually decide what materials and in what order a class of students should learn. In this way, personalized learning for each learner can't be achieved easily. In the E-learning, using a recommendation mechanism to recommend materials (learning objects) to learners according to their actual needs might be a better strategy because the class hedge has been broken. But how to figure out a learner's actual needs without a comprehensive online interrogation becomes a new challenge.

Table 1. LOM categories

Top Level	Description
General	The *General* category groups the general information that describes the resource as a whole.
Lifecycle	The *Lifecycle* category groups the features related to the history and current state of this resource and those who have affected this resource during its evolution.
Meta-Metadata	The *Meta-metadata* category groups information about the meta-data record itself (rather than the resource that the record describes).
Technical	The *Technical* category groups the technical requirements and characteristics of the resource.
Educational	The *Educational* category groups the educational and pedagogic characteristics of the resource.
Rights	The *Rights* category groups the intellectual property rights and conditions of use for the resource.
Relation	The *Relation* category groups features that define the relationship between this resource and other targeted resources.
Annotation	The *Annotation* category provides comments on the educational use of the resource and information on when and by whom the comments were created.
Classification	The *Classification* category describes where this resource falls within a particular classification system.

Currently, recommendation strategies can be divided roughly into three types, i.e. the content-based (Belkin. N. et al., 1992; Mooney, R. J. et al., 1999), the collaborative-based (Konstan, J. A. et al., 1997; Jin, R. et al, 2003(UAI); Jin, R. et al, 2003 *(*CIKM*)*; Hofmann, T., 2003; Hofmann, T., 2004) and the hybrid (Soboroff, I. Et al., 1999; Tran, T. et al., 2000; Popescul, A., 2001; Melville, P. et al., 2002) methods. The idea behind the content-based method is that if a user liked an object in the past, he/she would probably like other similar objects in the future. This method obtains learning objects' features and compares them with a users' profiles to predict their preferences (Jin, R. et al, 2003(UAI); Jin, R. et al, 2003 (CIKM)). Typically a collaborative-based method recommends learning objects to a user based on the preferences of his/her neighbors by using a nearest-neighbor algorithm. On the other hand, a hybrid method, attempting to combine the former two techniques to eliminate drawbacks of each, usually applies a user's profile and descriptions of learning objects to find users who have similar interests, and then uses collaborative filtering to make better predictions. For the time being, to combine the content-based and the collaborative-based method is usually considered to be a better methodology for the recommendation mechanisms. The mechanism in this paper is a hybrid method that recommends the learning objects. First, the preference-based algorithm will calculate a learner's preference score and the second correlation-based algorithm will provide similar learners' experience to calculate the helpfulness score. Finally, the two scores will be aggregated to one recommendation score. The detailed procedures will be introduced in a later section.

2.4 Problems on Recommendation Models

In traditional teaching strategies, learning objects of a course are assigned unique orders. Each learner can only follow this pre-defined sequence to explorer all the learning objects. E-learning systems have potentials of providing an adaptive and personalized learning style easily. Each learner may be allowed to select his/her desired learning objects to study. To do this, a system has to try inferring the semantics of a query a learner types in and recommends learning objects according to the learner's desire. While a beginner generally lacks basic knowledge of a specific domain, the query keywords input to a tutoring system for a specific subject may merely a surmise of the learner and the learning objects retrieved consequently may be incorrect or even misleading. In our observation, most

information retrieval (IR) systems that can perform well in keyword matching are deficient in semantic-aware searches or semantic discoveries, as well as inferring queries. Tutoring systems in the E-learning paradigm are subject to the same problem. In IR systems, users know what they meant to is a basic assumption. However in the E-learning paradigm, novice learners may have no idea about what they need and what the metadata of learning objects is. On the other hand, it is unreasonable to require learners to be familiar with the LOM standard, novices and professionals alike. Providing a group of fields for users to fill in is one possible solution. However this practice not only produces inflexible systems and but also requires learners to know various types of metadata. Even though learners are familiar with LOM, a system may be still not semantic aware. That is to say, the field descriptions and titles may be well defined, but the relationships of learning objects, especially those retrieved from other repositories and reused, in the system may not be well organized, and such a system may be incapable of organizing a suitable course for a specific learner if the semantics of input keywords is not properly recognized to reflex the real intention and preference of the user. And users' intentions and preferences awareness is what's called the *Personalization*.

Another challenge related to preferences is when a retrieve returns a result with numerous similar learning objects it is very difficult to decide the recommendation priority of these learning objects. Traditional content-based methods usually assume that a user should prefer something that has contents in accordance with her/his experiences. For the learning object case, this assumption is inadequate because a learner should expect to study something new and desirable. Thus traditional content-based methods should be more adequately changed to preference-based methods in E-learning systems. In this paper, both a learner's preference and other similar learners' experiences are used in ranking and recommending learning objects within top N learning objects to the learner.

3. An Java Tutoring System

A tutoring testbed has been setup to verify and evaluate the perspectives of proposed model. A primary element, the Java Learning Object Ontology (JLOO), of the system will be briefed here. Steps to organize learning objects into an ontology- based repository are also described briefly.

3.1 Java Learning Object Ontology

In 2001, The Joint IEEE Computer Society/ACM Task Force on the Model Curricula for Computing published the Computing Curricula 2001 (CC2001) (CC2001, 2001) that contained curriculum recommendations for undergraduate programming courses in computer science. The report also called for additional discipline-specific volumes for each of computer engineering, information systems, and software engineering. In the curriculum, six top-level concepts are suggested for the programming fundamentals; they are *Data model*, *Control structures*, *Order of execution*, *Encapsulation*, *Relationships among encapsulated components*, and *Testing and debugging* respectively. CC2001 also defined the "Description" and "Associated activities" of these concepts. "Description" describes the inner meaning of these concepts while "Associated activities" describes related activities in programming briefly. For example, the most obvious inner meaning of "Encapsulation" is information-hiding, and the related activities of "Control structures" are reading and explaining the effects of all the operations, implementing and describing these operations, and etc.

In one of our previous researches, an experimental Java programming course ontology -- the "Java Learning Object Ontology – JLOO" was built and introduced in (Lee, M., 2005). JLOO serves as a guideline in developing learning objects of introductory Java course and in organizing these learning objects in an adaptive learning environment. The work is based on CC2001 and is designed as an infrastructure of a Java learning object knowledge base. This ontology is focused on the atomic knowledge units of introductory Java programming, therefore the concept hierarchy has "learning concepts" only, without any Java APIs included. For the same reason, other advanced programming entities (such as swing, networking, multi-thread) are also not included. JLOO covers the six subjects defined in the fields of the introductory curriculum in CC2001. In the case of JLOO, the ontology terminology "concept" represents a learning unit in Java, "instance" represents a learning object that belongs to a concept, and "slot" represents properties and relations between two concepts.

The top level of JLOO is the six programming fundamental categories suggested by CC2001. In Figure 1 is a portion of the concept hierarchy of the category "*Control structure*".

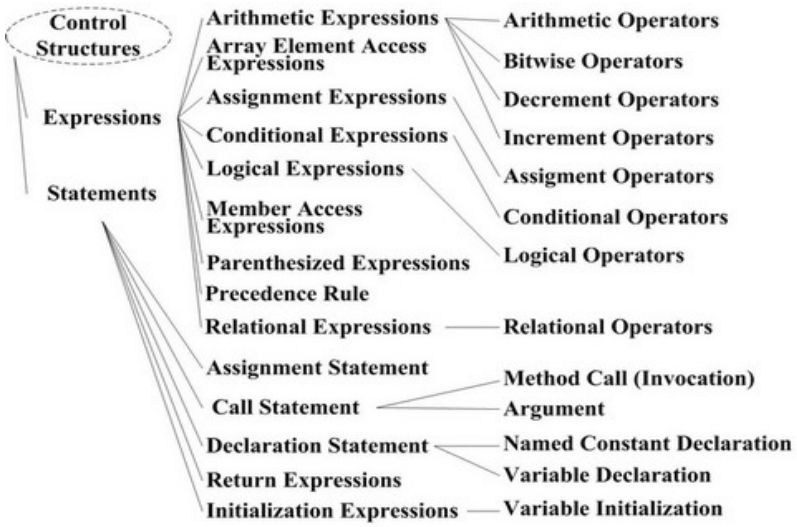


Figure 1. The portion of the concept hierarchy of the category "*Control structure*"

In JLOO, slots provide following information for concepts: 1) recording basic attributes of concepts, i.e., the metadata of learning objects. For example, the author, data, size, format, location, keywords, etc. 2) defining the relationships between concepts. A major slot in JLOO is PREREQUISITES. The PREREQUISITES indicates the concepts that must be learned to gain the prerequisite knowledge before studying a specific concept. A learning object is not to be provided unless the prerequisite condition is satisfied.

3.2 Building a Repository

All the learning objects in the repository are collected from the Internet according to some criteria. Before putting them into the repository, they are examined and formalized, and then classified according to the concept hierarchy in the JLOO. Figure 2 describes these steps and the detailed descriptions are as follows:

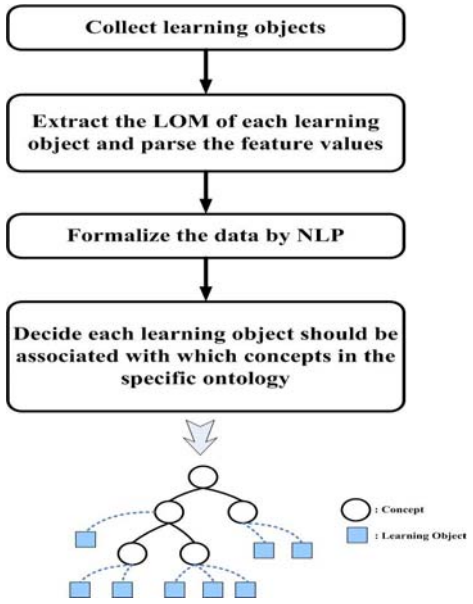


Figure 2. Building the Repository

STEP 1: Collect learning objects

To build the repository, a great number of learning objects has been collected from the Internet. All the learning objects collected are classified, fortified, and are associated to the JLOO. The collection is focused on JAVA programming related objects; most of them are web pages and others are specific learning objects.

STEP2: Extract the feature values from learning objects

The fields of the LOM constitute a standard, following which the detail information of a learning object is to be described. They are the important attributes and features of a learning object if the object is intended for sharing and reusing. As pointed above, for adaptive personalized recommendation, figuring out a learner's preferences is a necessity. But to choose proper learning objects for a learner, it's also necessary to perceive every learning object by extracting their features. If a learning object is obtained with LOM ready, its feature values are automatically extracted from the fields of the LOM (each field is considered as a feature) and are recorded in the ontology database with the learning object. Otherwise, the object is manually given a LOM with proper feature values before it is recorded into the ontology database. The ontology database is a database that uses ontology as an index for learners to look up for something. Some related definitions of the features are described as below.

Definition 1: Features of learning objects. The features are the characteristics of learning objects and the set of feature values represent a specific learning object. Nine categories (as in Table. 1) are specified in the LOM standard for describing the content of a learning object, each containing several fields. The LORM defines each field as one feature. In the features expression of learning objects, f_i represents a feature and LO^F means the set of all features.

$$LO^F = \{f_i \mid \forall f_i \in \text{the fields of LOM, and } f_i \neq f_j \text{ if } i \neq j\}$$

Definition 2: Feature values of a learning object. Each feature (or field) in the LOM may have several feature values and they may be different from the feature values of other learning objects of a similar kind. They are defined as below where $\mathbf{fv_k}$ is a feature value of feature $\mathbf{f_i}$ of a learning object **lo**. Each learning object **lo** can be described by the set of all (f_i, fv_k) pairs.

$$lo = \left\{ \begin{array}{l} (f_i, fv_k) \mid \forall f_i \in \text{the fields of LOM and } f_i \neq f_j \text{ if } i \neq j \\ \qquad\qquad , fv_k \text{ is the feature value of } f_i \end{array} \right\}$$

STEP3: Formalize the attribute values by NLP

The purpose to adopt natural language processing is two folds. One is to ensure the format of all the feature values in a same feature uniform for all the learning objects. Different learning objects may use different styles of feature values in a same feature when they are collected. Inconsistent format may result in wrong mappings to a learner's preferences when recommending learning objects. The other is to ensure precise classification of learning objects. In each learning object, there is a field that describes the content of the learning object using some statements. Generally, learners can understand what for the learning object is according to this field. For example, a JAVA learning object may have a description that reads "*This learning object introduces the concepts of variable declarations and data types such as long, int, short and byte*". A learner can then have an idea of what the learning object will teach. But such a description is difficult for a program to deal with automatically. Hence, a description is pre-processed by a NLP procedure to extract formalized keywords to form a set that are machine-readable. For example, the above description will be reduced to the keyword set as {variable, declaration, long, int, short, byte}.

STEP4: Associated with concepts in the specific ontology

All the learning objects collected are classified, and are associated to one or several concepts in the specific ontology - the JLOO. A special ontological approach (Lee, M. C. et al., 2006(LNCS); Lee, M. C. et al., 2006) is used to classify the learning objects, which makes the JLOO equivalent to an index of all the learning objects. For the time being the classification is partly manual because a lot of learning objects collected are without or with very simple LOM. If the result that comes from step 3 is able to decide automatically which concept(s) a learning object should be associated to, the object is put into the repository with the keyword set. When there is no LOM or the LOM is too simple, manual task is required to create a LOM or enhance the LOM for a learning object before it is sent into step 3 and 4. From the JLOO perspective, each of its concepts owns numerous learning objects. When a learner issues a query, the LORM will infer its semantics by matching the original keywords with the concepts in the ontology and expand the keywords with concepts from a specific part of the ontology hierarchy. The learning objects in the concepts, matched and expanded, are regarded as the candidates for the recommendation to the learner. More detail on recommendation processes is to be described in the next section.

4. Recommendation Model

The Learning Object Recommendation Model is divided into several phases as in Figure 3. The first three phases try to infer the real intention of a learner and try to find for the learner some suitable learning objects. Keywords entered in a query are expanded by exploiting the specific ontology hierarchy. The query expansion algorithm described in section 4.2 will find matched concepts and relationships among them. Matched concepts are then used to extend some more concepts by using specific rules. All the concepts, matched and extended, and relationships among them constitute a new keyword set that is used to collect learning objects within a range of suitability. In the following two phases for recommendation of learning objects, two evaluation algorithms that consider a learner's personal preference and his/her neighbours' suggestions are used to calculate the recommendation scores for ranking all the learning objects collected in the first three phases. The final phase deals with the feedbacks of learners and updates their profiles.

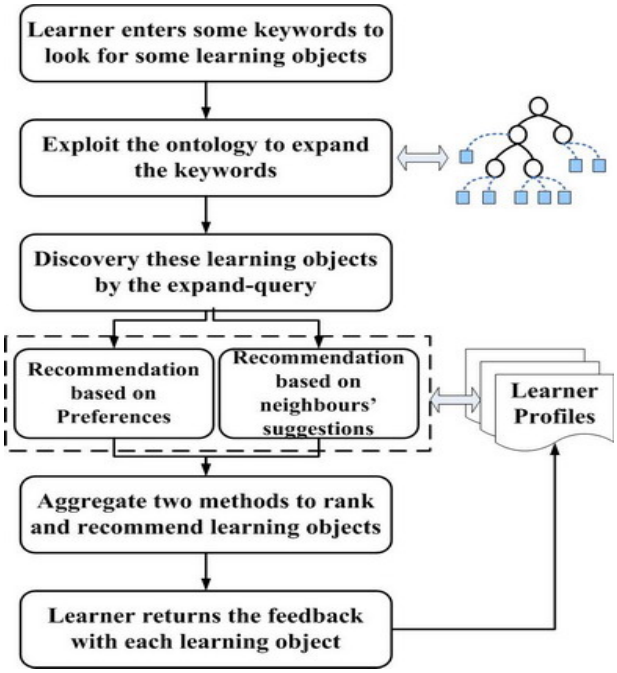


Figure 3. Phases for recommendation of learning objects

4.1 Query Expansion

Traditional search technologies, such as vector space model in informational retrieval system usually are keyword-based. The documents retrieved are those containing one or more user-specified keywords. However, documents may convey heavily desired semantic information implicitly without containing any explicit keywords, and thus may not be retrieved. This problem is usually solved by adopting query expansion algorithms, in which additional search terms are added to the original query based on statistical co-occurrence of terms (Smeaton, A. F. et al., 1993). Consequently, recall rate is increased, while precision is decreased (Peat, H. J. et al., 1991). One way to ease the precision problem is to index documents on both their context and meaning instead of keywords only. This requires a method of mapping the context and their possible meanings and the creation of a meanings- indexed database. Domain ontology, the semantic schema used to build metadata layer upon semantic web, contains the concept hierarchy and the detail features of all concept types of a domain, will provides such a conceptual and semantic indexing for documents. The LORM uses a semantic inference engine to find out the relationships between a user's query and the learning object metadata in the repository in order to semantically expand the learner's query for retrieving more candidate learning objects. Two phases are used, which are 1) *Construct the User Intention Tree (UIT) from ontology,* and *2) Expand query terms from UIT.*

STEP1: Construct the User Intention Tree (UIT)

A UIT is a sub-tree of an ontology hierarchy. It represents a user's intention by aggregating the nodes that are a minimum conjunction of matched concepts. In other words, an UIT is a semantic tree of a user's query constructed from the specific ontology.

Definition 3: Query Keywords and Concept Keywords. The set of keywords in a query ***i*** is called the query keywords set QK^i. In the specific ontology of the repository, each concept ***j*** is described by a set of CK_j.

$$QK^i = \{QK_1^i, QK_2^i, ..., QK_m^i\}, \text{ where } m \text{ represents the number of terms in a query } i$$

$$CK^j = \{CK_1^j, CK_2^j, ..., CK_n^j\}, \text{ where } n \text{ represents the number of terms in a concept } j$$

Definition 4: Basic Concepts and Basic Concept Scores. For a concept ***j*** and a query ***i***, ***j*** is called a Basic Concept (BC) for the ***i***, if $CK^j \cap QK^i \neq \varnothing$. In other words, a BC is a concept in the ontology which has at least one concept keyword that matches one of the query keywords. For each query, the query expansion algorithm first evaluates its Basic Concept Score (BCS) for each BC. These BCSs represents the similarity degree between a portion of concept hierarchy in the ontology and query keywords.

$$W_{QK_k^i, CK^j} = TF_{QK_k^i, CK^j} \times IDF_{QK_k^i, CK^j} . \quad (1)$$

$$TF_{QK_k^i, CK^j} = \begin{cases} \dfrac{1}{\left|CK^j\right|} & , QK_k^i \in CK^j \\ 0 & , \text{otherwise} \end{cases} . \quad (2)$$

$$IDF_{QK_k^i, CK^j} = \log \frac{\#\text{ of concepts in ontology}}{\#\text{ of concepts that match } QK_k^i} . \quad (3)$$

$$NW_{QK_k^i, CK^j} = \frac{TF_{QK_k^i, CK^j} \times IDF_{QK_k^i, CK^j}}{\sqrt{\sum_{x=1}^{\left|CK^j\right|} (TF_{CK_x^j, CK^j} \times IDF_{CK_x^j, CK^j})^2}} . \quad (4)$$

A modified TF-IDF approach is used to evaluate the BCS. The keyword-weighting is given by formula (1), where TF and IDF are evaluated by the formula (2) and (3), respectively. A normalized keyword-weighting NW is then calculated by formula (4).

The BCS of a basic concept ***j*** of a query ***i***, the BCS_{ji}, is the summation of all the normalized keyword-weighting of all the keywords in the query, which is also normalized to have a value between 0 and 1. Formula (5) calculates the BCS_{ji}.

$$BCS_{ji} = \frac{\sum_{k=1}^{|QK^i|} NW_{QK_k^i, CK^j}}{|QK^i|} . \quad (5)$$

Definition 5: Degree Function, Impact Score and Total Impact Score. The degree Function (Deg) defines the number of sub-concepts a concept ***i*** has, and is expressed by formula (6).

$$Deg_i = \begin{cases} \#\text{ of all descendants (includes non - direct link node)}, & i \notin \text{leaf node} \\ 1, & i \in \text{leaf node} \end{cases} \quad (6)$$

The impact score, IS_{ip} denotes the degree of the semantic impact a basic concept ***i*** may have on its ancestor concept ***p***. On the other hand, for each ancestor, the total semantic impact on it is defined as the Total Impact Score (TIS) and calculated by formula (8).

$$IS_{ip} = \left\{ BCS \times \frac{Deg_i}{Deg_p} \;\middle|\; \text{concept } p \notin \text{basic concept} \right\} \quad (7)$$

$$TIS_p = \left\{ \sum IS_{ip} \;\middle|\; \begin{array}{l} \text{concept } i \in \text{BC and is a descendant of concept } p, \\ \text{concept } p \notin \text{basic concept} \end{array} \right\} \quad (8)$$

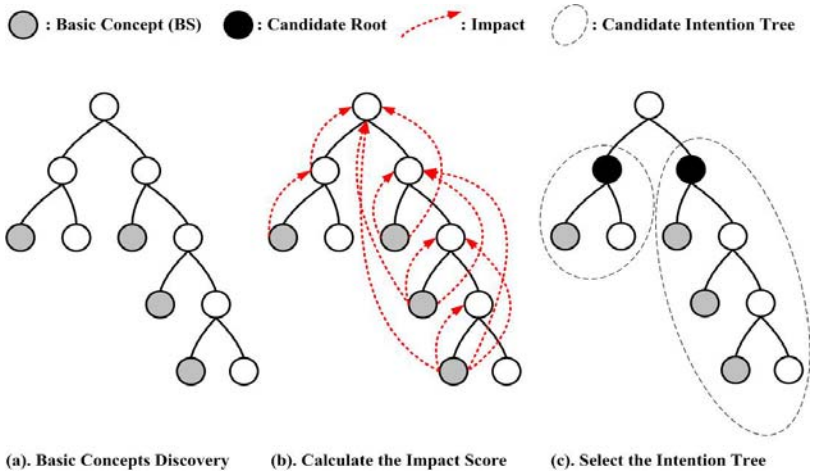


Figure 4. The process of constructing an UIT

Figure 4 shows the process of constructing an UIT. The main goal is to find an upper concept that is the root of a final intention tree associated with a user's query. First, in Fig. 4-(a), several basic concepts are discovered by

matching the QK and CK sets; their BCSs are calculated by formula (5). Each BC in the ontology hierarchy is also semantically related to its parent concept to a certain degree. In fact the kindred will trace back all the way to the root. Fig. 4-(b) shows how a BC impacts on its ancestors. A parameter λ has to be set, which is a threshold to limit the upper bound of an UIT. As shown in Fig. 4-(c), an algorithm is used to calculate the Total Impact Score (TIS) of all BCs' ancestors. Propagating up along the hierarchy, if there is an ancestor concept whose TIS is the smallest larger than λ, then it is recognized as a root and all its descendants are constructed as a candidate intention tree. Several candidate intention trees may be constructed but the one with the highest TIS will be selected as the final UIT in this step.

STEP2: Expanding query keywords from UIT

This step expands the query keywords according to the UIT constructed in the previous step. Since the proposed algorithm is aimed to be domain or ontology independent and a UIT may be one flourishing tree in the case of extracting from a huge or heterogeneously integrated ontology, there are possibly many candidate query expansion keywords which describe the concepts in the explicit UIT. However, not all of them should be added to the set of final expanded keywords. The system should only select some important keywords, each of which has closer correlations with query keywords. To determine the degrees of correlations between the candidate query expansion keywords and query keywords is equal to calculating the conditional probabilities between them (Cui, H. et al., 2003). In other words, that is the conditional probability of a candidate expansion keyword's appearance on the condition that the query keyword also appears in the LOM, and is denoted by $P(CK_x^j \mid QK_y^i)$ in which the concept keyword $CK_x^j \in UIT$. If a concept keyword CK_x^j is the ***x-th*** element of the concept keywords of concept ***j***, the query keyword QK_y^i is the ***y-th*** element of the query keywords of query ***i***, and let ***S*** be the set of LOMs in the learning objects repository, then, according to the Bayes' theorem, the conditional probability $P(CK_x^j \mid QK_y^i)$ can be defined as follows:

$$
\begin{aligned}
& P(CK_x^j \mid QK_y^i) \\
&= \frac{P(CK_x^j, QK_y^i)}{P(QK_y^i)} = \frac{\sum_{\forall LOM_k \in S} P(CK_x^j, QK_y^i \mid LOM_k) \times P(LOM_k)}{P(QK_y^i)} \\
&= \frac{\sum_{\forall LOM_k \in S} P(CK_x^j \mid LOM_k) \times P(LOM_k \mid QK_y^i) \times P(QK_y^i)}{P(QK_y^i)} \\
&= \sum_{\forall LOM_k \in S} P(CK_x^j \mid LOM_k) \times P(LOM_k \mid QK_y^i)
\end{aligned}
\qquad (9)
$$

In formula (9), it considers only these correlations which appear in the LOMs belonging to the concepts in a UIT. Each LOM_k in the set S represents the metadata of learning object k. Since direct calculation of the $P(CK_x^j \mid QK_y^i)$ is difficult, it is transferred into two parts, the $P(LOM_k \mid QK_y^i)$ and the $P(CK_x^j \mid LOM_k)$, where the $P(LOM_k \mid QK_y^i)$ is the conditional probability of the learning object k being selected in case that the QK_y^i appears among the keywords of query ***i*** and the $P(CK_x^j \mid LOM_k)$ is the conditional probability of occurrence of the CK_x^j if the learning object k is selected. The evaluation of the conditional probability $P(LOM_k \mid QK_y^i)$ and $P(CK_x^j \mid LOM_k)$ are as follows in formula (10) and (11):

$$
P(LOM_k \mid QK_y^i) = \frac{Fre(LOM_k, QK_y^i)}{Fre(QK_y^i)} \qquad (10)
$$

$$P(CK_x^j \mid LOM_k) = \frac{NW_{CK_x^j, LOM_k}}{\max_{\forall T \in LOM_k} (NW_{T, LOM_k})} \quad (11)$$

In formula (10) and (11), the $Fre(LOM_k, QK_y^i)$ is the frequency of the QK_y^i and the LOM_k (in selected learning object k) appearing together. The $Fre(QK_y^i)$ is the query frequency that contain the query keyword QK_y^i. The two factors were statistically obtained from the system logs. The $NW_{CK_x^j, LOM_k}$ is the normalized TF-IDF weight of the term CK_x^j in the LOM_k. The denominator of formula (11) is the maximum value of term weights in the LOM_k. By combining the probabilities of all query keywords, we can get the joint weight for each concept keyword in the following formula (12). Thus we can get a list of candidate expansion terms as well as the conditional probabilities between each terms and user query. The top-ranked terms are selected as the expansion terms.

$$W(CK_x^j, QK^i) = \log\left\{\prod_{y=1}^{|QK^i|} (P(CK_x^j \mid QK_y^i) + 1)\right\}, \text{where } CK^j \in UIT \quad (12)$$

4.2 Recommendation Processes

After the query expansion process, a user intention tree can be found and an original query is turned into a new expanded query by adding more semantically related keywords. According to the expanded query, several concepts in the specific ontology are discovered. As mentioned, each learning object may belong to one or several corresponding concepts in the ontology hierarchy. Thus, each concept discovered may contribute many similar learning objects. All the learning objects from all the discovered concepts constitute the recommendation result that may possibly be a huge volume. Therefore how to recommend precisely according to a learner's profile becomes an important issue. In the proposed model, the query expansion process finds a semantic-related result containing many learning objects that, for the purpose of personalized recommendation, will be precisely ranked by two algorithms, the preference-based and the correlation-based algorithms. The ranking algorithms will adjust the order the learning objects being presented to a learner.

4.2.1 Preference-based algorithm

Generally speaking, people always have their own preference on things like cars, books, and among others. Things, which are accorded with their preferences, can always interest them most; there is no exception in E-learning. A system that can recommend learning objects according to a learner's preferences will attract the learner to come back for more. The values of a feature of a learning object can help determine if a learner may prefer the object. For example, under the feature name *language*, a value *Chinese* may be preferred by some learners while a value *English* by others. Another example is the presentation style of a learning object. Learning object may be composed of various media, such as text document, audio/video clips, picture or flash, and etc. Different learners may prefer different presentations and strategies of learning objects for a same kind of topic. The preference-based algorithm is to bias the learning objects with a learner's preferences. Learning objects tending to suit a learner's preference more will get higher priorities when recommended to the learner. Before introducing the preference-based algorithm, the learner preference pattern is defined first.

Definition 6: Learner Preference Pattern. The Learner Preference Pattern (LPP^{user}) of a learner is the preference history of the features for the learner, which contains a list of Feature Scores, the (FS_1, FS_2,..., FS_n), of 2-tuples of (*fv*, *fs*). The *fs* is a feature preference score which represents how important the feature value *fv* is. After a learner studied a learning object, the LPP^{user} is updated according to the learner's preference feedback. How a personal LPP is updated will be discussed in the next section. The LPP^{user} thus represents the preference history of a learner. Each learner can explicitly choose what features he/she is interested initially in the system registering step, and can

implicitly be added a new feature when using a system recommended learning object with a feature not found in the LPP^{user} list. Formal definition is as follows.

$$LPP^{user} = \{ FS_i \}^*$$
$$FS_i = \{ (fv_k, fs_k)_{fi} \mid \forall fv_k \subseteq \text{the set of feature values for the feature } f_i \}$$

ALGORITHM: Preference-based algorithm
INPUT: Learner's ID, a learning object
OUTPUT: Preference score of a learning object

```
01 float CalculatePreferenceScore(int uid, int lo) {
02   for (each feature fi in LPP^uid) {
03     float pscore=0;
04     for (each feature value fvk in fi ) {
05       if ( lofi equal to fvk) {
06         pscore fsk of fvk += BPW(fvk);
07         break;
08       }
09     }
10   }
11   return 5*(pscore/|lo^F|);
12 }
```

Figure 5. Preference-based algorithm

Figure 5 shows the preference-based algorithm that calculates the preference score of one specific learning object *lo* when it has being used by a learner *uid*. The preference-based algorithm only calculates the preference score according to the features the learner *uid* has. If a feature value lo_{fi} of feature f_i in the learning object *lo* matches a record in the learner's LPP, the corresponding feature's preference score fs_k of feature value fv_k will be increased by a weight, the Basic Preference Weight that is defined, among others, as follows.

Definition 7: Basic Preference Weight. The Basic Preference Weight (BPW) is a weight to represent the degree of a learner's preference for a feature value in a feature. It is calculated by formula (13). The $\mathbf{fv}_{\mathbf{k}}^{\mathbf{fi}}$ represents the *k*-th feature value in the feature f_i and the $\mathbf{fs}_{\mathbf{k}}^{\mathbf{fi}}$ represents the feature preference score of the feature value $\mathbf{fv_k}$.

Definition 8: Preference Score. The preference score (PS) of a learning object is a score to represent the degree of a learner's preference for a learning object. The PS of each learning object is calculated according to the BPWs form a learner's LPP. After accumulating BPWs of feature values of all the features of a learning object, the preference score is the average of the sum divided by the number of features a learner currently has, the $|\mathbf{lo}^{\mathbf{F}}|$. Formula (14) calculates a PS.

$$BPW(fv_k^{fi}) = \frac{fs_k^{fi}}{\max\{fs_j^{fi} \mid \forall fs_j^{fi} \in \text{the score of feature } f_i\}} \quad (13)$$

$$PS(lo) = 5 \times \frac{\sum BPW(fv_k^{fi})}{\#\text{ of the features } LO^F \text{ a learner currently has}} \quad (14)$$

4.2.2 Correlation-based algorithm

It is not comprehensive by considering only a learner's preference for a learning object. Helpfulness of learning objects to a learner is not taken into account by the preference-based algorithm. But to pre-analyze a learning object's helpfulness for a learner does not seem easy. It can only be judged by the learner after being used. However, suggestions from other learners who have a similar profile as the learner are helpful and, for this reason, a correlation-based algorithm is proposed to strengthen the precise of recommendations. This algorithm predicts how helpful a specific learning object will be for a learner by analyzing other similar learners' feedbacks. A similar learner group is defined as the group of learners who used the same learning objects in the past and returned similar feedbacks. In general, in a similar learner group each learner may have several characteristics in common with other learners. So the idea of calculating helpfulness score of a learning object for a learner by consulting other similar learners' suggestion can be justified if a similar learner group is formed properly. Two main steps are carried out in the correlation-based algorithm.

STEP1: Look for similar learners

This process is to find out a group of learners who have similar learning experiences as a learner is summarized in Figure 6. The **s_threshold** is set as a similarity threshold. Those learners whose learning experiences are similar to a learner to a certain degree that is great or equal to the **s_threshold** are added to the set of similar learners, $\mathbf{Sim_{uid,lo}}$. In $Sim_{uid,lo}$, each element has two information, including a similar learner's ID, the **sl**, and his/her degree of similarity, the **sim(uid,sl)**, to the learner respectively. The value of **sim(uid,sl)** is calculated by formula (14) (Konstan, J. A. et al., 1997):

$$\mathrm{sim}(uid,sl) = \frac{\sum_{s \in S_{uid,sl}} (r_{uid,s} - \overline{r_{uid}})(r_{sl,s} - \overline{r_{sl}})}{\sqrt{\sum_{s \in S_{uid,sl}} (r_{uid,s} - \overline{r_{uid}})^2 \sum_{s \in S_{uid,sl}} (r_{sl,s} - \overline{r_{sl}})^2}} \qquad (14)$$

Where $\mathbf{S_{uid,sl}}$, is an intersection set that contains learning objects that both learners **uid** and **sls** have studied and returned their feedbacks. The $\mathbf{r_{uid,s}}$ and $\mathbf{r_{sl,s}}$ represent the returned scores for an identical learning object **s** from learners **uid** and **sl** respectively. The $\overline{\mathbf{r_{uid}}}$ and $\overline{\mathbf{r_{sl}}}$ are the averages of feedback scores of these learning objects that learners **uid** and **sl** have studied so far respectively.

ALGORITHM: algorithm of looking for similar-learners
INPUT: A learner's ID, a specific learning object, the similarity-threshold
OUTPUT: a group of similar learners

```
LookforSimilarLearners(int uid, int lo, float s_threshold) {
  Collect learners sls who have assigned feedbacks to learning object, lo;
  assign these learners sls to the set Sim_uid,lo;
  for (each learner sl in the Sim_uid,lo) {
    calculate the value of sim(uid, sl);
    if (sim(uid, sl) is equal to / greater than s_threshold)
      assign (sl, sim(uid, sl)) to Sim_uid,lo;
  }
  return the set Sim_uid,lo;
}
```

Figure 6. Look for similar-learners algorithm

STEP2: Calculate the helpfulness score

This second process is to calculate for a learner the helpfulness score for each learning object in the $\mathbf{LO_{qe}}$, which is the set of learning objects discovered by the query expansion algorithm. The detailed steps are shown in Fig. 7.

Different learners may evaluate a same learning object on different basis due to inherent characteristic differences. This relative error should be corrected to counterbalance the different learner attitudes toward a same learning object. So, a relative error score **rErrorScore** is calculated first by formula (15) for this purpose. In other words, formula (15) uses the learning experiences of other similar learners to decide how interested a target learner is possibly in a learning object.

$$\text{rErrorScore}(\text{uid}, \text{Sim}_{\text{uid,lo}}) = \left(\sum_{\text{sl} \in \text{Sim}_{\text{uid,lo}}}^{|\text{Sim}_{\text{uid,lo}}|} (r_{\text{sl,lo}} - \bar{r}_{\text{sl}}) \text{sim}(\text{uid}, \text{sl}) \right) \Big/ \left(\sum_{\text{sl} \in \text{Sim}_{\text{uid,lo}}}^{|\text{Sim}_{\text{uid,lo}}|} | \text{sim}(\text{uid}, \text{sl}) \right) \quad (15)$$

Definition 9: Helpfulness Score. The helpfulness score (HS) is the score which is calculated by considering other similar learners' learning experiences. In other words, the helpfulness score of a learning object **HS(lo)** is predicted by aggregating and revising other similar learners' feedbacks. Formula (16) shows how to calculate the HS. Among, the rErrorScore is a relative value, so the average of past feedbacks of a learner (ex. the value $\bar{r}_{\text{uid}}$ at line 2 in Figure 7) must be added to it and an absolute helpfulness score is produced.

$$\text{HS(lo)} = \text{avg}(\text{FB}_{\text{uid}}^{\text{all}}) + \text{rErrorScore}(\text{uid}, \text{Sim}_{\text{uid,lo}}) \quad (16)$$

```
ALGORITHM: algorithm for calculating helpfulness score
INPUT: A learner's ID, the set of similar learners for ID
OUTPUT: Helpfulness score of a lo
01  CalculateHelpfulnessScore (int uid, Set Sim_uid,lo) {
02    float hScore=0;
03    hScore = r_uid + rErrorScore(uid, Sim_uid,lo);
04    return hScore;
05  }
```

Figure 7. The helpfulness score calculation algorithm

4.2.3 Learning objects rating

For each learning object in the $\mathbf{LO_{qe}}$, the recommendation score (**RS**) is the aggregation of both the preference score and the helpfulness score, which are calculated according to a learner's preference and other similar learners' feedbacks. The two scores are aggregated by formula (17). All candidate learning objects in a $\mathbf{LO_{qe}}$ will be ranked by their **RSs**. The larger the RS a learning object has, the higher the priority the learning object will get.

$$\text{RS}_{\text{lo}} = \text{wPS} + (1 - \text{w})\text{HS} \quad (17)$$

The parameters, the **w** and the **(1-w)**, are used to weight complementarily the preference score and the helpfulness score. Although static values can be assigned to them (e.g. w=0.5), it is better to tune the optimal ratios for the two weights dynamically. To achieve true adaptive personalized recommendation, the proposed model adjusts values of w dynamically and periodically. Experiments will show some of the result of tuning.

4.3 Learner feedback

Feedbacks are the important information for this model to recommend learning objects. For the time being, after learning a learning object, a learner may return two kinds of feedbacks, the **content feedback** and the **preference feedback**. The former is to report whether the content of a learning object is helpful for a learner and is accorded with the learner's ability while the latter is to say whether a learner prefers the style of a learning object. For the

content feedback, it can be intuitive, but for preference feedback, the evaluation is more difficult. Since the LOM of a learning object contains so many features, it's difficult to figure out exactly which features a learner prefers. Hence, two rules and a formula are designed to solve this problem.

Rule1. If a learner chooses one learning object to study, but later returns no feedback or the preference feedback is less than 3, then no increase is made to each feature value of the learning object in the learner's LPP.

Rule2. If a learner chooses one learning object to study and later returns a preference feedback greater than or equal to 3, then increase is made to each feature value of the learning object in the learner's LPP. The magnitude of the increase Δs is calculated using formula (18).

$$\Delta s = pf \times FVS \text{, where}$$

$$FVS = \log \frac{|LO_{qe}|}{\#\text{ of the LOs which own the same feature value in } LO_{qe}} \quad (18)$$

It is assumed that the distribution of feature values in all LOs is a uniform distribution. Thus, formula (18) assumes that the distribution of the feature values in LO_{qe} is also a uniform distribution. The value **pf** represents the preference feedback and the value of **FVS** represents the significance of the feature value in a LO. The value of **FVS** is higher if the feature value appears in fewer learning objects and vice versa. In other words, the feature value is seen as an important preference for the learner or not. The **FVS** might need to be revised if the distributed of the feature values is not a uniform distribution. The revision is done by formula (19). It is clear that the distribution of feature values in all LOs are considered directly and the cost in calculation is huge if there are many LOs in the LO repository.

$$FVS = \log \frac{|LO|}{\#\text{ of the LOs which own the same feature value in LO}} \quad (19)$$

The purpose of the rules is that the system must understand whether a learner really prefers the style of a learning object or not. With them, the system can give and increase each feature value of a learning object a suitable score when a learner returns feedbacks for these learning objects the learner has used.

5. Experimental Evaluation

Several experiments are conducted to evaluate our approach to show its feasibility, effectiveness, and benefits. Through them the results will be analyzed in order to observe whether the learning objects recommendation model can offer adaptive and proper learning objects to learners. As mentioned before, for the experimental domain ontology in the proposed framework, an introductory java programming ontology has been built and learning objects are collected and classified to associate them to the domain ontology. It currently has totally 158 concepts and 94 relations that cover the concepts of [Data Models], [Control Structures], [Order of Execution], [Encapsulation], [Relationships among Encapsulated Components], and [Testing and Debugging], which were defined in the field of introductory curriculum in CC2001. The experiments will also introduce the evaluative measures and dynamically adjust several parameters in the recommendation algorithms to tune the variations of the model to an optimal status. To see how the experiments are evaluated, the recommendation procedures are summarized again as follows.

1. A learner enters the some query keywords (are also called original query keywords) from the user interface.
2. These query keywords are inferred and expanded by using the JAVA ontology, the JLOO, to form a new expanded query. By executing the semantic-aware concept discovery algorithm, a set of learning objects LO_{qe} will be returned.
3. For each learning object in the LO_{qe}, two kinds of score the preference score (PS) and the helpfulness score (HS), are calculated. For the preference score, the preference-based algorithm will compare the feature values of each learning object in the LO_{qe} with the preference in a learner's LPP and each learning object in the LO_{qe} will be assigned a preference score, which indicates the degree of the preference a learner is for a learning object. For the helpfulness score, the correlation-based algorithm will look for these users who are similar to a learner and calculate the helpfulness score for each learning object according to the similar users' feedbacks.

4. Lastly, the two scores, the PS and the HS, are aggregated into a recommendation score (RS) for each learning object in the LO_{qe}. Then the recommendation module uses the RS score to rank the learning objects and present the top 10 ones to the learner.

5.1 Evaluation Measures

The effectiveness of an information retrieval system is usually measured by two criteria, the "**Recall**" and the "**Precision**" rate. The definition of "**Recall**" rate is the number of retrieved relevant documents divided by the number of all the relevant documents, which are previously identified by domain experts. The definition of "**Precision**" rate is the number of retrieved relevant documents divided by the number of all retrieved documents. For the proposed model, if *Relevant* is the set of relevant learning objects for a specific query and *Retrieved* is the learning objects retrieved by the system for the specific query, then the formal definitions of **Recall** and **Precision** can be represented as the following forms:

$$\text{Recall} = \frac{\text{number of retrieved relevant LOs}}{\text{number of all relevant LOs}} = \frac{|\text{ Relevant LOs }| \bigcap |\text{ Retrived LOs }|}{|\text{ Relevant LOs }|}$$

$$\text{Precision} = \frac{\text{number of retrieved relevant LOs}}{\text{number of all retrieved LOs}} = \frac{|\text{ Relevant LOs }| \bigcap |\text{ Retrived LOs }|}{|\text{ Retrieved LOs }|}$$

In addition to the comparisons with other approach, experiments are also conducted to measure the precision of the model, in which the original learner's feedbacks and the **Mean Absolute Error (MAE)** are used to evaluate the recommendation results. Two different learner's feedbacks, the preference feedbacks and the content feedbacks, which are returned from similar learners, are employed. The preference feedback is defined as the degree of preference indicating how much a learner prefers a specific learning object, while the content feedback is to represent the degree of the preference and helpfulness other similar learners feel on a specific learning object. The MAE is the error between a learner's feedback and the system's prediction, shown as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N},$$

where, N represents the number of comparisons. In each comparison, two kinds of MAEs are compared, the preference MAE and the helpfulness MAE. The prediction accuracy is better when the MAE value is lower.

5.2 Experiment Results and Discussion

Two series of experiments are conducted. The first one is focused on how to discover suitable learning objects for a learner. In this series, comparisons are made on the Keyword Based Discovery (KBD) and the proposed Semantic-Aware Discovery (SAD). The KBD uses a simple discovery mechanism that only calculates the degree of syntactic similarity between keywords in a learner's query and those in the descriptions of learning objects. The SAD, however, considers semantically the relations between keywords in a learner's query and those in the descriptions of learning objects by consulting the domain ontology, the JLOO. The second series use the preference-based, and the correlation-based algorithms to recommend learning objects for a learner. The experiments will demonstrates the benefits of these recommendation algorithms and checks whether the features and contents of recommended learning objects are in accord with learners' preferences and are helpful for them.

Keyword Based Discovery & Semantic-aware Discovery

In these experiments, retrieval effectiveness of different discovery approaches is quantified by standard measures of average recall and precision rates. Two aggregate metrics are used: (1) the interpolated 11-point average precision and (2) the mean average precision over the cutoff value of 2, 4, 6, 8, and 10 expanded keywords. Figure 8 shows the interpolated 11-point recall-precision curves for the SAD approach using different number of expanded keywords and the KBD approach. Obviously the SAD approach performs better than the KBD approach in precision by at least

66% improvement in average, and, when limiting the number of expanded keywords to between 2 and 6, it yields better precision than the KBD approach regardless of the recall rates. On the other hand, the result shows that the algorithm performs well while the number of expanded keyword is limited to 2, 4 and 6. When the recall is less than 10%, expanding 6 keywords yields the best performance. However in the query expansion phase, using an average precision is more adequate than using the top-ranked precision. Thus, in the second series of the experiments, the number of expansion terms was set as 4, as can be seen in the figure that SAD-4 shows the better performance than others in average.

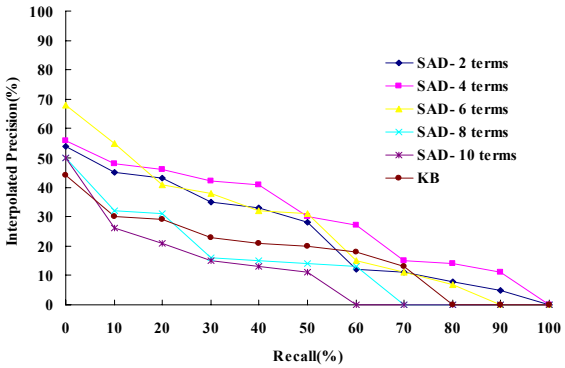


Figure 8. Interpolated Recall-Precision curve for different expanded terms (2~10) and KB

Recommendation Based on Preference and Helpfulness with Different Weights

In this series of experiments, learning objects are recommended for learners. Candidate learning objects are selected by the semantic-aware discovery algorithm (SAD) and placed to into LO_{qe} and feedbacks from learners are evaluated according to the preferences of and helpfulness for the learners. The default number of learning objects for recommendation is set to 10 and each learner has to return two kinds of feedbacks, the preference feedback and the helpfulness feedback after he/she has studied every learning object received. The system will calculate the preference and helpfulness of each learner afresh. For learners to feedback easily, the range of feedback values is divided into five levels representing *very bad, bad, common, good* and *very good* respectively. To understand how the recommendation results affect learners, not only the learners' feedbacks are observed but also the MAE values are measured between the learners' feedbacks and system predictions. Observing the learner's feedbacks directly is to understand whether the proposed model recommends learning objects in accord with learners' desire, while calculating MAEs shows whether it can infer learner's preference and interest accurately or not. In the experiment, different weights are used between the preference and the helpfulness; they are set to a proportion of *w and 1-w* where *w* is the weight of the preference and *1-w* represents the weight of the helpfulness.

Experiment Result 1 –Analysis variation of preference feedbacks with different weights

This experiment observes the variation of preference feedbacks for the learners with different weights, i.e w= 0.3 and 1-w=0.7, w= 0.5 and 1-w=0.5, and w= 0.7 and 1-w=0.3. The results are shown in Figure 9.

Figure 9(a) displays how soon the learners' preference patterns are formed. The x axis represents the number of learning objects studied. For example, (Y=2, X=5) means that the average score of preference feedbacks for the first 5 uses of learning objects is 2. Obviously, the (0.7/0.3) proportion performs slightly better than the others. The best average preference feedback score is about 2.3, but it does not seem to be a satisfying result. The preference MAE has to be observed to understand the reasons that result in poor results of low preference feedback score. Figure 9(b) shows how the preference MAE varies against the number of learning objects studied. All three weight proportions have their average MAEs lower than 1. This means in fact the system can infer learners' preferences accurately to recommend suitable learning objects. The reason behind the poor performance is probably that the feature values of candidate learning objects are not completely in accord with learners' preferences. Figure 9(c) demonstrates how the

preference MAE varies against the number of experiments. The values oscillate between 0.5 and 1 and this proves again that the algorithms can infer learners' preferences accurately.

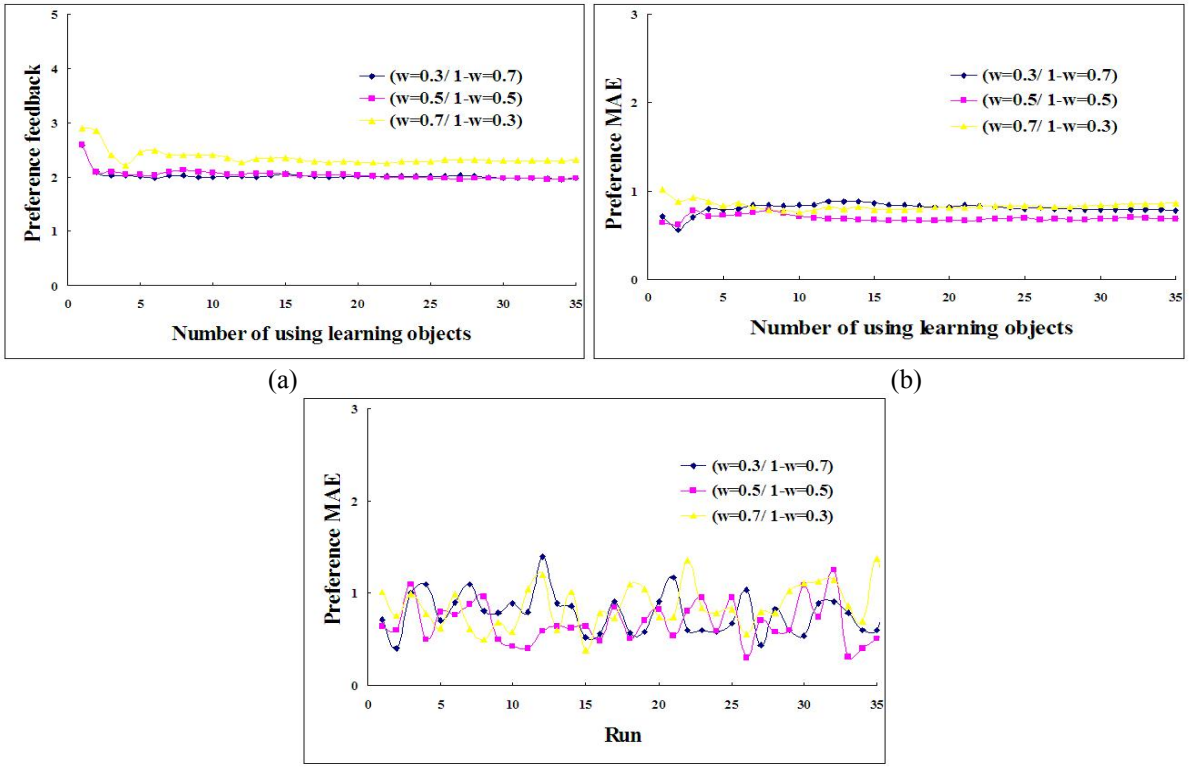
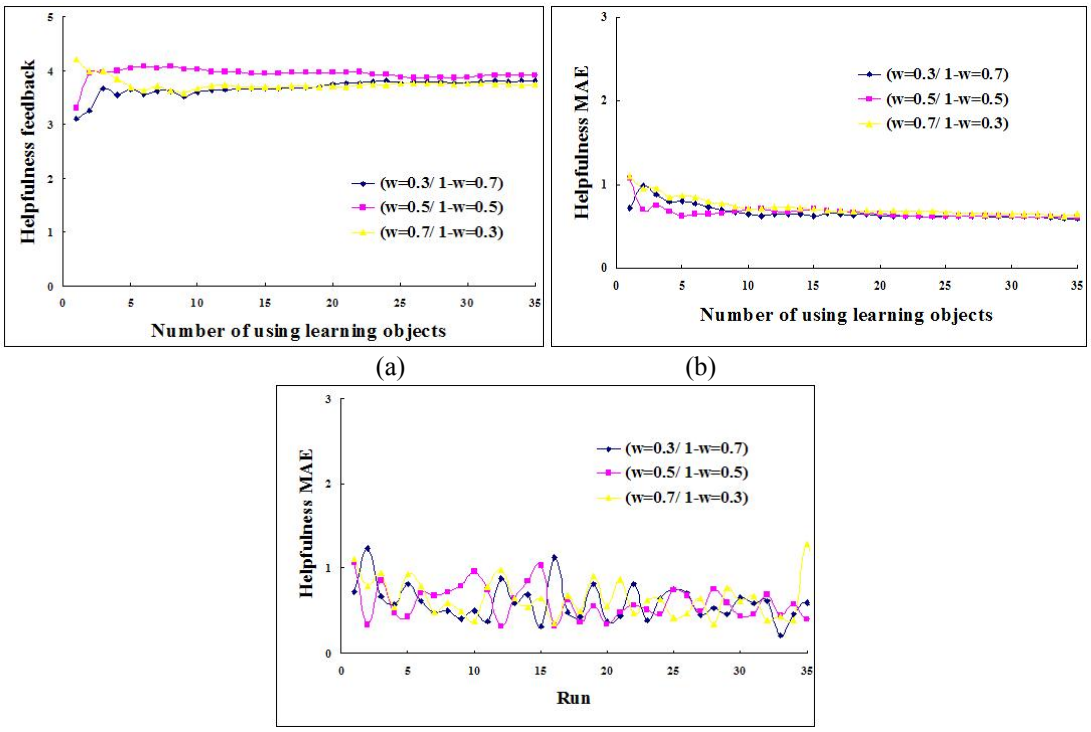


Figure 9. Comparison of preference feedbacks. (a) Direct preference feedbacks
(b) Preference MAE varies against the number of learning objects studied.
(c) Preference MAE varies against the number of experiments.

Experiment Result 2 –Analysis the variation of helpfulness feedbacks with different weights

This experiment observes the variation of preference feedbacks for the learners with the same set of weight proportions as the first experiment. Learners send feedbacks according to how they feel on the content of learning objects recommended for them, helpful or not. As mentioned, five levels have been defined for the values of feedback. The results are shown in Figure 10.

In Figure 10(a), it's clear that the helpfulness feedbacks of three different weight proportions are between 3.5 and 4. As shown in Figure 10(b) the helpfulness MAE is close to 0.7 with increasing number of learning objects studied. And in Figure 10(c), it can be seen that, with more learning objects studied, the variation of content MAE becomes milder. This indicates that the system can accurately predict learners' preference and learning objects' suitability when it becomes more stable. The so-call cold-start problem can be observed here that erroneous predictions reduced when there are enough feedbacks from learners in the system. The experiment also shows that this problem can be reduced or eliminated even faster in the proposed model when learning objects are also ranked according to the average of other learners' feedbacks.



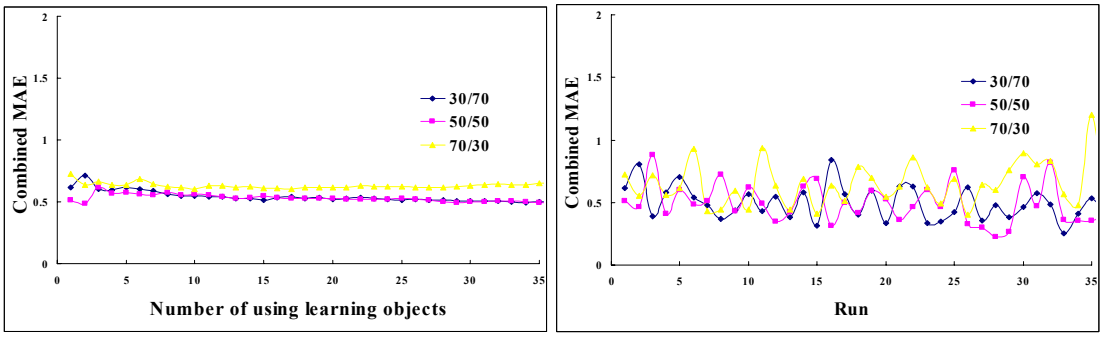
(a) (b)

(c)

Figure 10. Comparisons of helpfulness feedbacks. (a) Direct helpfulness feedbacks.
(b) Helpfulness MAE varies against the number of learning objects studied.
(c) Helpfulness MAE varies against the number of experiments.

Experimental Result 3 – Analysis variation of combined feedbacks with different weights

Figure 11(a) and (b) show how the aggregated MAE of both preference MAE and helpfulness MAE vary. Figure 11(a) reveals that when w=0.5/1-w=0.5 and w=0.3/1-w=0.7 better recommendations can be obtained. Similarly, the oscillations of MAE in both cases are also milder. This means that there is only a minor error at each prediction when using both proportions.



(a) (b)

Figure 11. Comparison of combined feedbacks.
(a) Combined MAE varies against the number of learning objects studied.
(b) Combined MAE varies against the number of experiments.

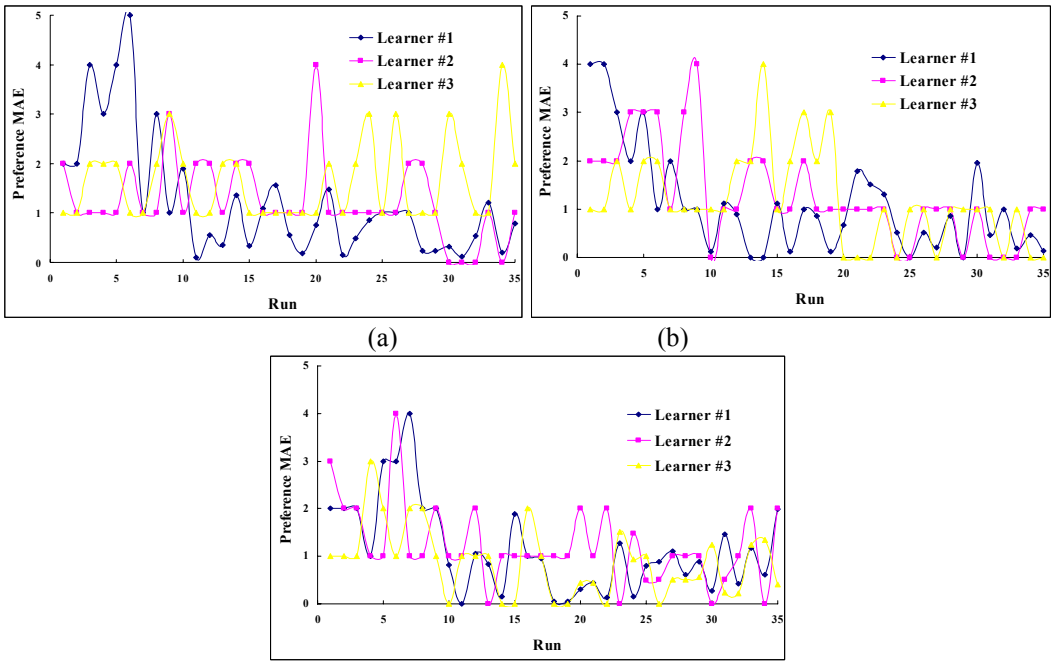


Figure 12. Variation of preference MAE in the learners' LPP (a) Preference weight = 30% (b) Preference weight = 50%. (c) Preference weight = 70%.

Experimental Result 4 – Construction of a learner's preference pattern

In the last experiments, we describe the process of forming the learner's preference patterns (LPP). The Figure 12(a), (b) and (c) describe variations of the preference MAEs of three different weights. When the preference aspect is assigned to higher weight likes 50% or 70%, the preference patterns seem to be formed more quickly. The result displays in Figure 12(b) and (c). Although, certain of recommendations still obtain the high errors, the preference MAE of majority of recommendations are close to 1 or less than 1.

6. Conclusion

In the near future, E-learning will become more and more popular. Issues have to be solved before a learner can be really benefited from the vast amount of E-learning object repositories on the Internet. This paper proposes an adaptive recommendation model for retrieving and recommending for a learner suitable learning objects. First, the system correctly classifies learning objects to the corresponding concepts within the JLOO. Next, the intention of learners is understood by semantic-aware discovery algorithm that uses knowledge in JLOO to infer what learning objects a learner should study and what candidate learning objects the system should retrieve automatically from the repository. These suitable learning objects retrieved are presented to a learner according to the result of a ranking mechanism that employs the preference-based algorithm and the correlation-based algorithm, which not only infer a learner's preference periodically but also predict the suitability of a learning object by referencing the history of usages of other similar learners. The model has been experimentally proved efficient in the adaptive personalized learning object recommendations, and can be easily embedded into an ontology-based E-learning tutoring system of repositories.

References

ADL-1 (2001). Advanced Distributed Learning (ADL) initiative. *SCORM Specifications – The SCORM Content Aggregation Model Version 1.2*, Retrieved June 7, 2007, from, http://www.adlnet.gov/scorm/history/Scorm12/

index.aspx.

ADL-2 (2001). Advanced Distributed Learning (ADL) initiative. *SCORM Specifications – The SCORM Run-Time Environment Version 1.2*, Retrieved June 7, 2007, from, http://www.adlnet.gov/downloads/downloadpage.aspx ?ID=218.
ADL (2003). Advanced Distributed Learning (ADL) initiative. *SCORM Specifications – SCORM Version 1.3 Application Profile Working Draft Version 1.0*, Retrieved June 7, 2007, from, http://www.adlnet.gov/News/articles/ index.aspx?ID=126.

ARIADNE (1998). *The Alliance of Remote Instructional Authoring and Distribution networks for Europe*, Retrieved June 7, 2007, from, http://ariadne.unil.ch/.

Belkin , N., & Croft, B. (1992). Information Filtering and Information Retrieval. *Communications of the ACM, 35*(12), 29-37.

CC2001 (2001). *Computing Curricula of The Joint IEEE Computer Society/ACM Task Force*, Retrieved June 7, 2007, from, http://www.computer.org/education/cc2001/.

Cui, H., Wen, J. R., Nie, J. Y., & Ma, W. Y. (2003). Query Expansion by Mining User Logs. *IEEE Transaction on Knowledge and Data Engineering, 15*(4), 829-839.

Hofmann, T. (2003). Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. *Paper presented at the 26th Annual International ACM SIGIR Conference,* July 28-August 1, 2003, Toronto, Canada.

Hofmann, T. (2004). Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems, 22*(1), 89-115.

Jin, R., Si, L., & Zhai, C. (2003). Preference-Based Graphic Models for Collaborative Filtering. *Paper presented at the 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003)*, August 7-10, 2003, Acapulco, Mexico.

Jin, R., Si, L., Zhai, C., & Callan, J. (2003). Collaborative Filtering with Decoupled Models for Preferences and Ratings. *Paper presented at the International Conference on Information and Knowledge Management*, November 2-8, 2003, New Orleans, Louisiana, USA.

Jung, S. Y., Hong, J. H., & Kim, T. S. (2005). A statistical model for user preference. *IEEE Transactions on Knowledge and Data Engineering, 17*(6), 834-843.

Konstan, J. A., Miller, B. N., Maltz, D. J., Herlocker, L., Gordon, L. R. & Riedl, J. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM, 40*(3), 77-87.

Lee, M. C., Ye, D. Y., & Wang, T. I. (2005). Java Learning Object Ontology. *Paper presented at the 5th IEEE International Conference on Advanced Learning Technologies*, July 5-8, 2005, Kaohsiung, Taiwan.

Lee, M. C., Tsai, K. H., Ye, D. Y., & Wang, T. I. (2006). A Service-Based Framework for Personalized Learning Objects Retrieval and Recommendation. *Lecture Notes in Computer Science, 4181*, 336-351.

Lee, M. C., Tsai, K. H., & Wang, T. I. (2006). An Ontological Approach for Semantic-Aware Learning Object Retrieval. *Paper presented at the 6th IEEE International Conference on Advanced Learning Technologies*, July 5-7, 2006, Kerkrade, The Netherlands.

LOM (2002). *Final LOM Draft Standard*, Retrieved June 7, 2007, from, http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-Boosted Collaborative Filtering for Improved Recommendations. *Paper presented at the 18th National Conference on Artificial Intelligence*, July 28–August 1, 2002, Edmonton, Canada.

Mooney, R. J., & Roy, L. (1999). Content-Based Book Recommending Using Learning for Text Categorization. *Paper presented at the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, August 19, 1999, Berkeley, USA.

Peat, H. J., & Willett, P. (1991). The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science, 42*(5), 378-383.

Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2001). Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. *Paper presented at the 7th Conference in Uncertainty in Artificial Intelligence*, August 2-5, 2001, Seattle, Washington, USA.

SCORM (2003). *Sharable Courseware Object Reference Model*, Retrieved June 7, 2007, from, http://www.adlnet.gov/downloads/downloadpage.aspx?ID=243.

Smeaton, A. F., & Rijsbergen, C. J. (1993). The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal, 26*(3), 239-246.

Soboroff, I., & Nicholas, C. (1999). Combining Content and Collaboration in Text Filtering. *Paper presented at the International Joint Conference on Artificial Intelligence – Workshop on Machine Learning for Information Filtering*, August 1, 1999, Stockholm, Sweden.

Tran, T., & Cohen, R. (2000). Hybrid Recommender Systems for Electronic Commerce. *Proceedings of the AAAI-00 Workshop on Knowledge-Based Electronic Markets,* Technical Report WS-00-04, AAAI Press.