Content Provisioning for Ubiquitous Learning

The context-aware and QoS-enabled approach described here uses a knowledge-based semantic recommendation method, a fuzzy logic-based decision-making strategy, and an adaptive QoS mapping mechanism to support content provisioning in ubiquitous learning.

Zhiwen Yu and Yuichi Nakamura
Kyoto University

Daqing Zhang
Institut TELECOM

Shoji Kajita and Kenji Mase
Nagoya University

The emergence of e-learning lets students access electronic course content easily and conveniently via the Web. With the vision of ubiquitous computing becoming reality, people will soon live in environments surrounded by networked computers and mobile devices. Such trends have precipitated the advent of ubiquitous learning, helping students access educational content anytime, anywhere.

A crucial feature of ubiquitous learning is adaptability—students getting the right information at the right place in the right way.[1] To achieve learning adaptability, content provisioning must consider the student's context. We can classify a user's learning context into two categories: *personal context* refers to information about the user, such as prior knowledge, goals, learning style, and schedule, whereas *infrastructure context* depicts features of the physical infrastructure such as terminal capability and network condition.

In our research, we emphasize the user's quality-of-service (QoS) requirements within context-aware content provisioning. For example, if video courseware streaming over a low-bandwidth network contains important text (such as a lecturer's writing on a blackboard), the student might want a high-resolution image stream, even at the expense of longer delay.

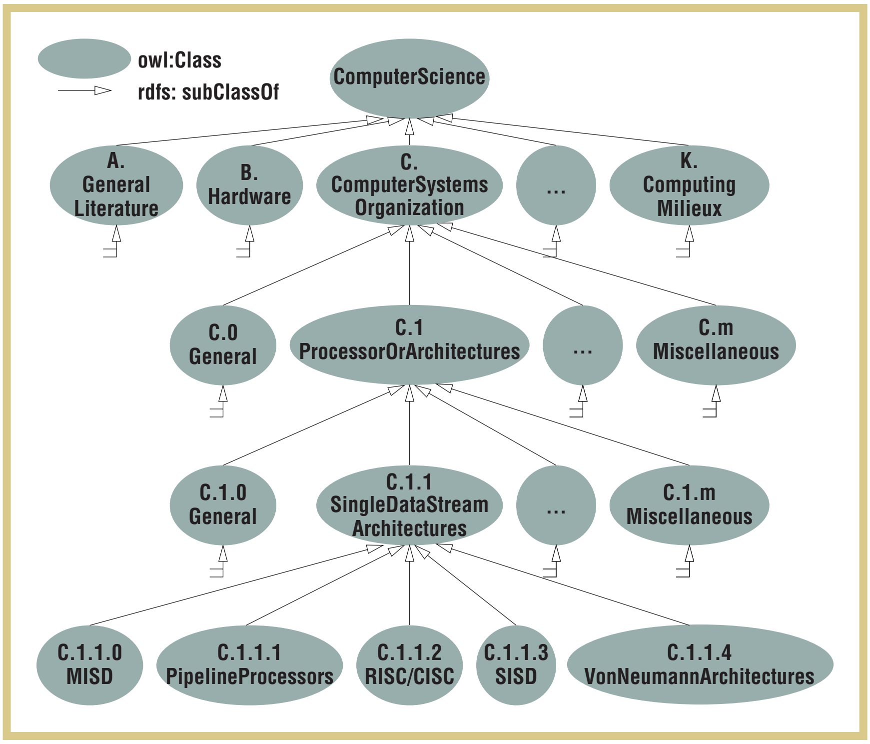
The combination of a dynamic learning context and QoS requirements poses challenges to delivering satisfactory educational content. We propose an approach that provides the right content in the right form to the right student, based on a variety of contexts and QoS requirements. We first use knowledge-based semantic recommendation to determine which content the user really wants and needs to learn. We then apply fuzzy logic theory and dynamic QoS mapping to determine the appropriate presentation according to the user's QoS requirements and device/network capability.

Representation Model

To ease knowledge interoperability and sharing, we designed three ontologies: a context ontology, a learning content ontology, and a domain ontology. The context ontology depicts the content already mastered by the student, along with his or her learning goals, available learning time, location, learning style, and interests. It also describes the hardware/software characteristics and network condition of the student's client devices. The learning content ontology defines educational content properties as well as the relationships between them. The relation *hasPrerequisite* describes content dependency informa-

Published by the IEEE CS ■ 1536-1268/08/$25.00 © 2008 IEEE

Figure 1. Computer science domain ontology. The ontology from the ACM taxonomy (www.acm.org/class/1998/) shows the conceptual proximity calculation.



tion—that is, content required for study before learning the target content.

Today, most university departments provide a course dependency chart. We propose the domain ontology to integrate existing consensus domain ontologies such as computer science, mathematics, and chemistry. The domain ontologies are organized as a hierarchy to reflect the topic classification.

Semantic Content Recommendation

The content recommendation procedure consists of four steps:[2] semantic relevance calculation, recommendation refinement, learning path generation, and recommendation augmentation. This procedure determines which content the user really wants and needs to learn.

Calculating Semantic Relevance

For content recommendation, we first rank learning content according to how much it satisfies the student's context—in this case, we mainly consider the student's learning goal. (In this article, "learning content" denotes any content used for learning, including formal course content and informal materials.) Our system introduces the semantic relevance between the goal and learning content as the ranking criteria.

This semantic relevance concept is inspired by category theory and conceptual graphs;[3] it's based on the intuition that objects in the same or related domain have certain similarities. We calculate semantic relevance via the following steps:

1. Map the user's learning goal to the domain ontology.
2. Locate the learning content's subject in the domain ontology.
3. Estimate the conceptual proximity between the mapped element and the learning content's subject node.

The conceptual proximity ($S(e1, e2)$) is formally defined according to the following rules ($e1$ and $e2$ are two elements in the hierarchical domain ontology):

Rule (1): The conceptual proximity is always a positive number—that is, $S(e1, e2) > 0$.

Rule (2): The conceptual proximity has the property of symmetry—that is, $S(e1, e2) = S(e2, e1)$.

Rule (3): If $e1$ is the same as $e2$, then $S(e1, e2) = Dep(e1)/M$.

Rule (4): If $e1$ is the ancestor or descendant node of $e2$, then

$$S(e1, e2) = Dep(e)/M$$

$$e = \begin{cases} e1 & e1 \text{ is the ancestor node of } e2 \\ e2 & e1 \text{ is the descedant node of } e2 \end{cases}$$

Rule (5): If $e1$ is different from $e2$ and there is no ancestor–descendant relationship between them, then $S(e1, e2) = Dep(LCA(e1, e2))/M$.

In Rules 3, 4, and 5, M denotes the total depth of the domain ontology hierarchy; $Dep(e)$ is the depth of node e in the hierarchy (the root node always has the least depth, say, 1); and $LCA(x, y)$ means the least common ancestor node for nodes x and y.

Figure 1 shows the computer science domain ontology, which comes from the ACM taxonomy (www.acm.org/class/1998/). We use it as an example here to show the conceptual proximity calculation. With the Rules 3, 4, and 5, we can see that $M = 5$; $LCA(MISD, SISD) = SingleDataStreamArchitecture$; $Dep(LCA(MISD, SISD)) = 4$; hence, $S(MISD, SISD) =$

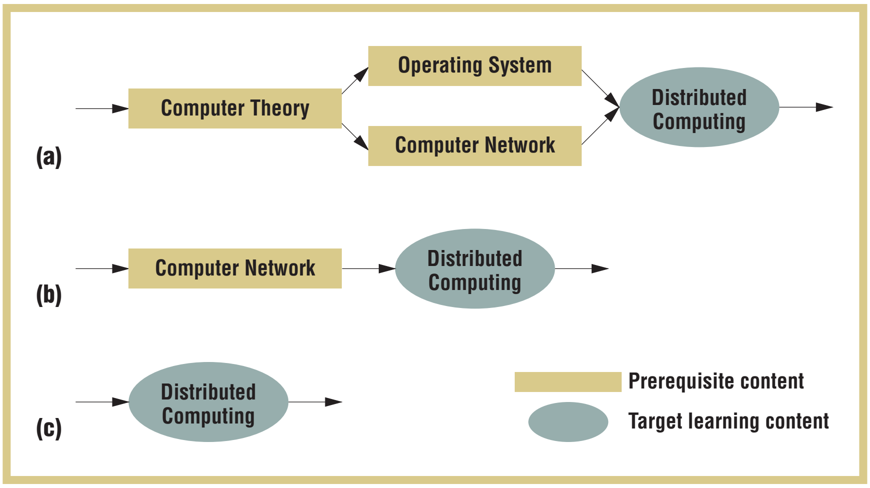


Figure 2. Learning path (example). (a) A full learning path for the user who has no knowledge about computer theory; (b) a revised learning path for the user who has already taken the "Operating System" course; and (c) a revised learning path for the user who has mastered "Operating System" and "Computer Network."

Dep(LCA(MISD, SISD)) = *4/5* = *0.8*. The semantic relevance is defined based on the intuition that two subjects with detailed contents and closer ancestors are more relevant to each other—for example, two subjects under *"SingleDataStreamArchitecture"* are known to be more relevant than two subjects under *"ProcessorOrArchitecture"*.

After we calculate semantic relevance, we can rank the contents and recommend those whose semantic relevance is larger than a preset threshold.

Refined Recommendations

The student can get a recommendation list with respect to semantic relevance, but it could include overwhelming amounts of information or contents that don't match the student's preferences, such as difficulty level. Our system offers interactive recommendation refinement, through which the student can interact with the system, critique its recommendation, and interactively refine the results until achieving acceptable options. Specifically, users can refine recommendation results according to the following features: specialty, difficulty, and interactivity.

Specialty. If the recommendation contains few items and the student wants more generalized content, the system can provide all contents whose subject is one level higher than the LCA in the hierarchy. Similarly, if the recommendation includes many items and the student wants more specialized ones, the system can return those contents whose subject is one level lower than the LCA in the hierarchy. When the user triggers the "more specialized" refining action, a dialog will pop up, asking the user to choose an LCA subclass.

Difficulty. The student can refine the recommendation by choosing easier or more difficult contents through the *hasDifficulty* property. Each content segment is assigned a difficulty level when authored, such as "very easy," "easy," "medium," "difficult," and "very difficult." This criteria applies to each item in the recommendation list, so if the student wants to obtain easier contents with item X as reference, the system will generate contents whose difficulty level is lower than that of X, while the other features remain the same.

Interactivity. Similar to difficulty, the student can get content at different levels of system participation by increasing or decreasing a particular item's interactivity level via the *hasInteractivity* property. When created, the author gives the content an interactivity level ranging from "very low," "low," "medium," "high," to "very high," according to its presentation method and layout.

Generating Learning Paths

Recommending a single learning content isn't usually sufficient for the student to meet an educational goal because learning contents themselves might have prerequisites the student hasn't mastered yet. Therefore, we must provide the student with a path to guide the learning process and suggest prerequisites that he or she must complete before tackling the target content.

When the student selects an item from the recommendation list, the system generates a learning path that connects prerequisite contents with the target content. It does this by recursively adding prerequisite content until the path reaches the content that has no prerequisites, and then it prunes the path based on the student's prior knowledge. The *hasPrerequisite* relation of a particular content provides the prerequisite course information. The learning path should be a directed acyclic graph (DAG); we just detect and eliminate cyclic graph in building the path.

Let's assume, for example, the student selects "Distributed Computing," which has two prerequisites— "Operating System" and "Computer Network" —each of which has the same prerequisite, "Computer Theory." If the user has no knowledge about computer theory, our system suggests the learning path shown in Figure 2a; if the user has already taken the "Operating System" course, it recommends the learning path in Figure 2b. It offers the learning path in Figure 2c after the user has mastered "Operating System" and "Computer Network."

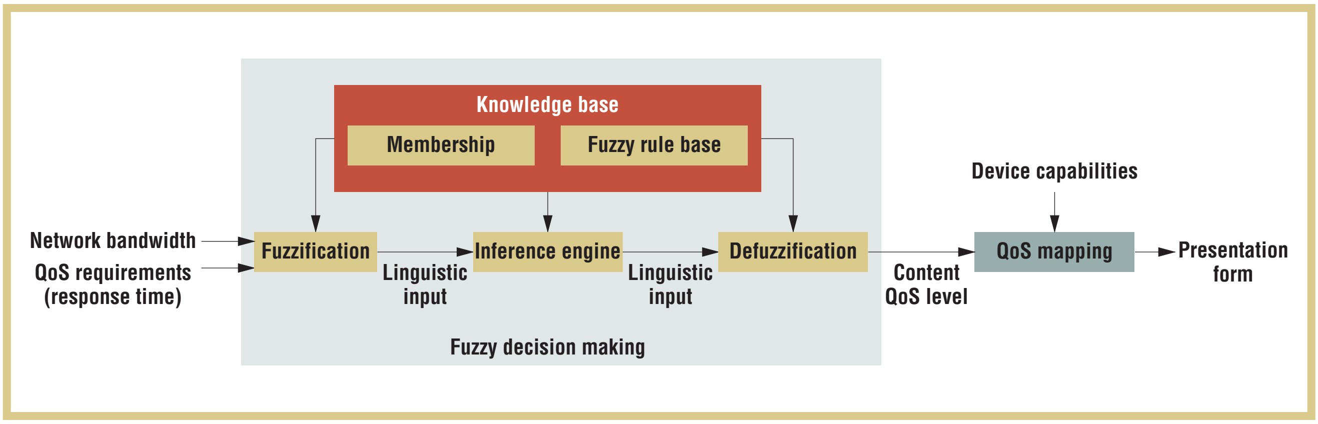


Figure 3. Presentation-form-determining procedure. The fuzzy decision making takes network bandwidth and the user's quality-of-service (QoS) requirements as inputs to generate QoS levels through a fuzzy reasoning process. The QoS mapping maps the QoS level to machine-understandable parameters according to client device capabilities.

Augmenting Recommendations

While studying the main course content, the student usually has to refer to some appendant content within it—for instance, when given a concept, the student hopes to see some examples to strengthen his or her understanding of it, maybe by taking a quiz (here, "main" distinguishes course content with other materials such as quizzes and exercises). In our system, we provide recommendation augmentation with references to examples, exercises, quizzes, and examinations related to the main course the user is studying. It does this by aggregating the course contents through "*hasExample*," "*hasExercise*," "*hasQuiz*," and "*hasExamination*." Then, the system provides links for appendant contents along with the main course content. With the recommendation augmented, the student merely has to click on a button rather than look up extra material.

QoS-Enabled Presentation

After selecting the learning content, our system determines the presentation form using the procedure shown in Figure 3. This process has two steps:[4] fuzzy decision making and QoS mapping. The fuzzy decision making takes network bandwidth and the user's QoS requirements (response time) as inputs. As a result, it generates appropriate QoS levels through a fuzzy reasoning process. The QoS mapping dynamically maps the QoS level to machine-understandable parameters (such as frame size, frame rate, and so on) according to client device capabilities. Finally, it decides how to present the learning content.

Fuzzy Decision Making

The fuzzy decision making is based on fuzzy theory, and it consists of four steps.

Define membership functions for I/O. In the decision-making process, we set network bandwidth and desired response time as input, with content QoS level as output. Figure 4a and 4b show fuzzy membership functions of the network bandwidth and response time. To present the bandwidth and response time universally, we normalize them in the range of [0, 1], according to the following equations:

$$E_1(network\_bandwidth) = \begin{cases} 1 & if\ network\_bandwidth \geq 512\text{ kbps} \\ \dfrac{network\_bandwidth}{512\text{ kbps}} & if\ network\_bandwidth < 512\text{ kbps} \end{cases}$$

$$E_2(response\_time) = \begin{cases} 1 & if\ response\_time \geq 60\text{ s} \\ \dfrac{response\_time}{60\text{ s}} & if\ response\_time < 60\text{ s} \end{cases}$$

We classify the bandwidth and response time into three fuzzy sets, respectively. Each particular input can belong to one or two fuzzy sets with a corresponding degree of membership. Figure 4c shows the output's fuzzy membership function (that is, QoS level). It's represented with five fuzzy sets: "very low," "low," "medium," "high," and "very high."

Map to fuzzy membership. By using the membership functions defined earlier, we translate the input values of network bandwidth and response time into a set of linguistic values and assign a membership degree for each linguistic value.

Get the linguistic values of QoS level. The inference engine makes decisions based on fuzzy logic inference rules. Each rule is an IF-THEN clause in nature, which determines the

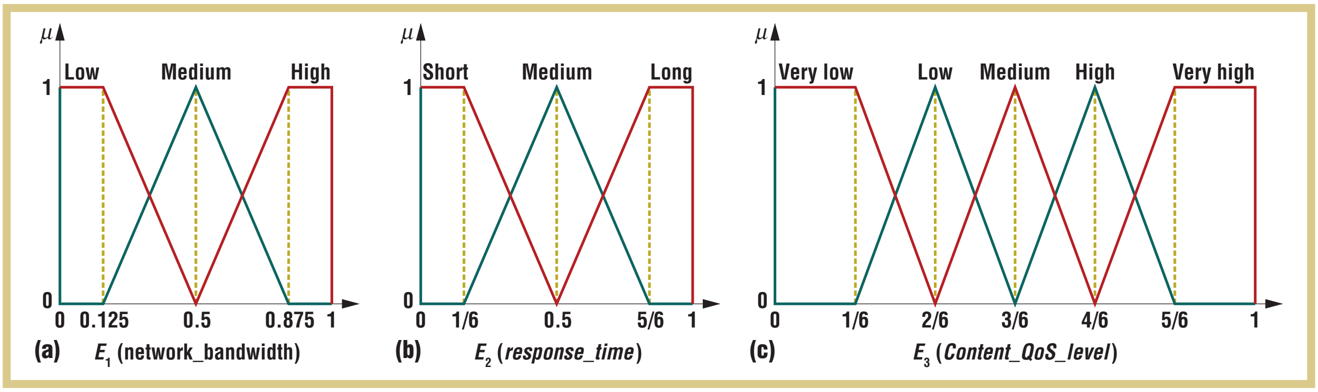


Figure 4. Fuzzy membership functions. (a) Network bandwidth, (b) response time, and (c) output variable (content quality-of-service [QoS] level).

linguistic value of the QoS level (E_3) according to the linguistic values of network bandwidth and response time (E_1 and E_2). Based on the experiences and analysis, we set several sample rules as follows:

1. If E_1 is "low" and E_2 is "short," then E_3 is "very low."
2. If E_1 is "low" and E_2 is "medium," then E_3 is "low."
3. If E_1 is "low" and E_2 is "long," then E_3 is "medium."
4. If E_1 is "medium" and E_2 is "short," then E_3 is "low."
5. If E_1 is "medium" and E_2 is "medium," then E_3 is "medium."
6. If E_1 is "medium" and E_2 is "long," then E_3 is "high."
7. If E_1 is "high" and E_2 is "short," then E_3 is "medium."
8. If E_1 is "high" and E_2 is "medium," then E_3 is "high."
9. If E_1 is "high" and E_2 is "long," then E_3 is "very high."

The first rule infers the content QoS level as "very low" if the available bandwidth is "low" and the user's desired response time is "short." But the QoS level will rise if the user is willing to wait for a longer time, as indicated by Rules 2 and 3.

Transform the QoS level's linguistic value into a crisp value and generate the final QoS level. We adopted the most common defuzzification method, called *center of gravity*, to get the QoS level's crisp value (that is, the real number). The center of gravity method is as follows:

$$Content\_QoS\_level = \frac{\sum_{i=1}^{n}\left(\mu[i] \times y_i\right)}{\sum_{i=1}^{n}\mu[i]},$$

where

- $\mu[i]$ is the height of output area from the ith rule,
- y_i is the gravity's horizontal coordinate of output area from the ith rule, and
- n is the total number of matching rules for given values of E_1 and E_2.

With the crisp value of content QoS level, we map it into its fuzzy membership and choose the linguistic value whose membership degree is the largest as the final QoS level. For instance, if the crisp value of content QoS level is 0.7, according to Figure 4c, we get

μ ContentQoSlevel = "high"(0.7) = 0.8,
μ ContentQoSlevel = "*very* high"(0.7) = 0.2.

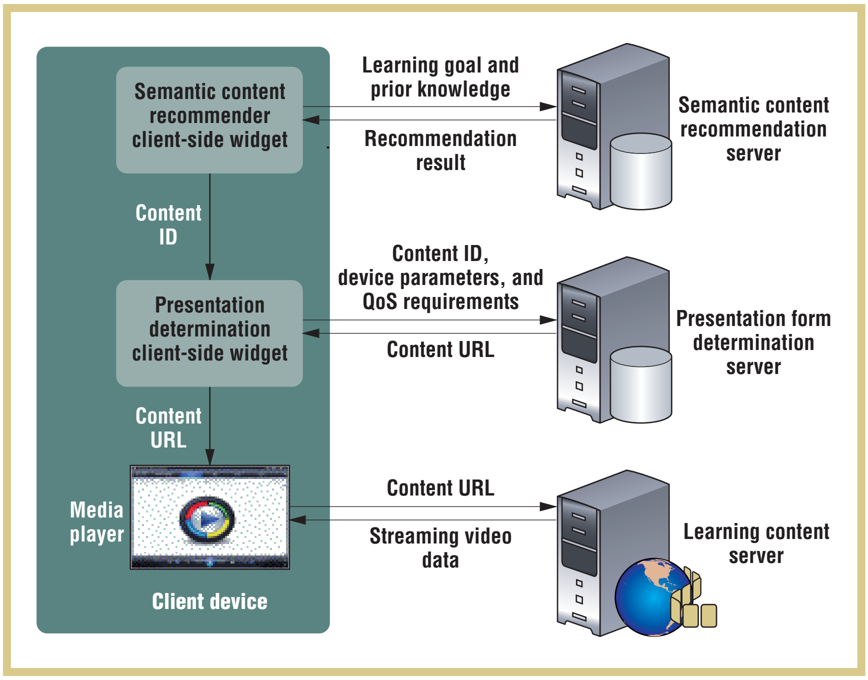
Hence, the set with the largest membership degree is "high"—that is, the final QoS level is "high."

QoS Mapping

The system itself can't understand QoS values suggested through fuzzy decision making, so we should map the QoS level to machine-understandable parameters. Existing systems usually conduct QoS mapping statically before the application starts, but they don't take into account changing device features. Sometimes, existing systems can't guarantee the QoS—for instance, the frame size largely relies on the device's resolution size. To address this, we propose an adaptive QoS mapping strategy that dynamically sets quality parameters at runtime according to the client device's capabilities.

We could map different QoS parameters for different media modalities, such as video, audio, or image. Let's take video streaming as an example and assume the QoS dimensions include frame size, format, frame rate, and quantization scale. Our system divides frame size into eight levels: 740×480, 640×480, 480×360, 360×240, 240×176, 176×144, 160×120, and 128×96. Maximum frame size relies on the device's display resolution; format depends on the operating system and software installed. Usually the maximum frame rate for video streaming is 30 frames per

Figure 5. Prototype architecture. It consists of a client device, a semantic content recommendation server, a presentation determination server, and a learning content server. Three components in the client device interact with the three servers, respectively.



second (fps). The quantization scale is related to the image quality and takes integer values ranging from 1 to 31, with a lower value means better quality but larger files. Using a value of 1 theoretically leads to the highest image quality but, again, generates very large files, so in practice we use 2 as the maximum quality quantization value. The five-level QoS mapping is as follows:

- Q5 ("very high"). To present content of the highest quality, our system sets the quantization scale to 2 and the other three categories as the maximum value that the device supports.
- Q4 ("high"). On the basis of Q5, our system decreases the frame rate to 20 fps and sets the quantization scale to 10.
- Q3 ("medium"). On the basis of Q4, our system decreases the frame rate to 15 fps, sets the quantization scale to 17, and decreases the frame size one level if possible.
- Q2 ("low"). On the basis of Q3, our system decreases the frame rate to 10 fps, sets the quantization scale to 24, and decreases the frame size one level if possible.
- Q1 ("very low"). On the basis of Q2, our system decreases the frame rate to 5 fps, sets the quantization scale to 31, and decreases the frame size one level if possible.

So, given the device capabilities and the suggested QoS level, the QoS mapping finally generates the machine-understandable presentation form for the learning contents.

Implementation and Evaluation Results

We developed a prototype of a context-aware and QoS-enabled learning content provisioning system. We then conducted experiments from both the system perspective and user viewpoint to evaluate it.

Prototype Implementation

Figure 5 illustrates our prototype architecture, which mainly consists of a client device, a semantic content recommendation server, a presentation determination server, and a learning content server. The system stores content metadata in the semantic recommendation server and various kinds of file formats, frame rates, and resolutions in the learning content server.

The client device contains three collaborating components: a semantic content recommender widget, a presentation determination widget, and a media player. The semantic content recommender widget provides interaction between the student and the recommendation server. It lets the student indicate learning goals and prior knowledge and displays recommendation results from the server. When the user decides to learn a particular topic, the presentation determination widget launches itself and asks the user to input device capabilities and QoS requirements in terms of response time. Then the presentation determination server decides what QoS parameters should accompany the content and returns the specific content variation's URL. Finally, the media player uses the content URL to retrieve material from the learning content server.

Evaluation Results

To evaluate our system's performance, we measured the overhead of the semantic content recommendation and presentation determination algorithms. We deployed the recommendation server on a PC with a 1.60-GHz Pentium 4 CPU and 1 Gbyte memory running Windows XP and the presentation determination server on an Apple MacBook, 2.0-GHz Pentium 4 CPU with 512 Mbytes RAM running the Mac operating system. The total ontology contains roughly 4,000 Resource Description Framework (RDF) triples. The content server holds 250 learning contents. The time for each experiment is an average value of 10 runs. We observed that

Related Work in Ubiquitous Learning

Researchers have proposed numerous ubiquitous learning systems in the past few years. The University of Tokyo[1] built a system that enabled people to learn anytime, anywhere by deploying RFIDs onto a variety of objects, such as food, medicine, and resorts. The European Learning in Process project[2] provides immediate learning on demand for knowledge-intensive organizations by incorporating context into the design of e-learning systems. Hiroaki Ogata and Yoneo Yano[3] at the Tokushima University built a ubiquitous learning environment that supported learning in polite expressions Japanese by using the student's situational and personal information. Iraklis Paraskakis[4] at the University of Sheffield proposed a paradigm of ambient learning that provided access to material at the time, place, and pace that best suits the individual student. Stephen J.H. Yang[5] at the National Central University built a ubiquitous learning system that allowed peer-to-peer content access and real-time group discussion. Qun Jin[6] at the Waseda University developed collaborative services to facilitate social intercommunion in ubiquitous learning.

A few projects address the content adaptability of ubiquitous learning. The European Elena project[7] provides resource filtering according to text and category. iWeaver[8] offers students different media experiences based on their learning styles. Coldex[9] considers the student's preferences and hardware/software characteristics in serving educational materials. Birgit Bomsdorf[10] at the University of Hagen used a rule-based ascertainment engine to identify educational resources according to the student's situation.

Our research differs from previous work in several aspects. First, for adaptive provisioning, we consider not only the user's learning context (both personal and infrastructure) but also his or her quality-of-service (QoS) requirements. Second, we provide content recommendation based on ranking, recommendation refinement, learning path generation, and recommendation augmentation in a knowledge-based semantic approach. Third, we determine presentation by utilizing fuzzy logic theory and dynamic QoS mapping.

REFERENCES

1. K. Sakamura and N. Koshizuka, "Ubiquitous Computing Technologies for Ubiquitous Learning," *IEEE Int'l. Workshop Wireless and Mobile Technologies in Education* (WMTE05), IEEE CS Press, 2005, pp. 11–20.
2. A. Schmidt and C. Winterhalter, "User Context Aware Delivery of E-Learning Material: Approach and Architecture," *J. Universal Computer Science*, vol. 10, no. 1, 2004, pp. 28–36.
3. H. Ogata and Y. Yano, "Context-Aware Support for Computer-Supported Ubiquitous Learning," *IEEE Int'l. Workshop Wireless and Mobile Technologies in Education* (WMTE04), IEEE CS Press, 2004, pp. 27–34.
4. I. Paraskakis, "Ambient Learning: A New Paradigm for E-Learning," *3rd Int'l. Conf. Multimedia and Information & Communication Technologies in Education*, Formatex, 2005, pp. 26–30.
5. S.J. Yang, "Context Aware Ubiquitous Learning Environments for Peer-to-Peer Collaborative Learning," *Educational Technology and Society*, vol. 9, no. 1, 2006, pp. 188–201.
6. Q. Jin and G. Zhang, "Research on Collaborative Service Solution in Ubiquitous Learning Environment," *6th Int'l Conf. Parallel and Distributed Computing, Applications and Technologies* (PDCAT05), IEEE CS Press, 2005, pp. 804–806.
7. B. Simon et al., "Smart Space for Learning: A Mediation Infrastructure for Learning Services," *12th Int'l. World Wide Web Conf.* (WWW03), ACM Press, 2003, pp. 616–620; http://zoltan.miklos.googlepages.com/p616-simon.pdf
8. C. Wolf, "iWeaver: Towards 'Learning Style'-based e-Learning in Computer Science Education," *5th Australian Computing Education Conf.*, Australian CS Press, 2003, pp. 273–279.
9. N. Baloian et al., "A Model for a Collaborative Recommender System for Multimedia Learning Material," *10th Int'l. Workshop Groupware*, 2004, Springer-Verlag Press, LNCS 3198, pp. 281–288.
10. B. Bomsdorf, "Adaptation of Learning Spaces: Supporting Ubiquitous Learning in Higher Distance Education," *Mobile Computing and Ambient Intelligence: The Challenge of Multimedia*, Dagstuhl-Seminar 05181, Saarland, 2005; http://drops.dagstuhl.de/opus/volltexte/2005/371/pdf/05181.BomsdorfBirgit.Paper.371.pdf

the content recommendation list generation time was the largest, at 78 ms, but the total time for semantic recommendation was less than 100 ms; the presentation determination algorithm needed just 3 ms.

Next, we conducted a study to evaluate the system's usability. We primarily measured user acceptance based on the provisioned content, response time, and the interface. We invited 14 students (eight majoring in information science and six majoring in economics and arts) to use the system and complete a questionnaire; Table 1 shows the results. The testers expressed satisfaction with content provisioning, response time, refinement, and learning path, but they had mixed feelings about the interactivity during content refinement. Specifically, those who weren't familiar with IT technologies reported difficulty in understanding the interactivity levels. Furthermore, when reviewing the appendant content, one participant said he was interested in citations (such as related works) to which the current content referred. Despite these issues, all participants appreciated the learning tool's overall system features, such as its ubiquity and flexibility.

We also conducted a comparative user study in two learning contexts: we asked one group (three men and two

TABLE 1
Usability study results.*

Question	Average rating	Standard deviation
I was satisfied with the recommended content and its presentation.	3.8	0.69
I was satisfied with the response time.	4.3	0.75
It was quick to reach my target through refinement—that is, by adjusting the features of specialty, difficulty, and interactivity.	4.0	0.71
It was easy to understand the difficulty and interactivity levels in content refinement.	3.5	1.41
The learning path was useful to guide my learning.	4.5	0.71
The four kinds of appendant content offered were appropriate for my study.	4.0	1.00
I would use this ubiquitous learning tool again.	4.6	0.48

*5 = strongly agree, 4 = agree, 3 = neutral, 2 = disagree, 1 = strongly disagree

TABLE 2
Comparative study results.*

	Stationary learning without context-aware and quality-of-service enabled (CAQE)		Ubiquitous learning with CAQE	
Question	Average rating	Standard deviation	Average rating	Standard deviation
Ease of use (it was easy, convenient, and quick to find the materials that you want to learn).	1.4	0.49	4.4	0.49
Accessibility (the learning materials could be accessed anytime and anywhere).	1.2	0.40	4.2	0.75
Adaptability (the content could be adapted according to network condition or user requirements).	1.4	0.49	4.0	0.63
Information display (the content and user interface were displayed clearly for easy reading, understanding, and usage).	4.4	0.49	3.4	0.49
Time usage (I could make full use of my time for learning the materials).	2.8	0.75	4.2	0.75
Learning effect (the system was very effective in learning your new materials).	2.2	0.75	4.6	0.49

*5 = strongly agree, 4 = agree, 3 = neutral, 2 = disagree, 1 = strongly disagree

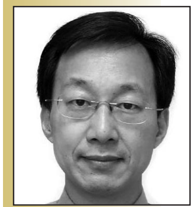
women) to use a stationary device (desktop PC) without context-aware and QoS-enabled (CAQE) function (just menu-based content selection and fixed-form presentation) and the other group (four men and one woman) to use our system (ubiquitous learning with CAQE features). The learning devices included a desktop PC, handheld PC, and PDA, and the connection included wired and wireless networks. After testing two different systems for 10 days, the subjects filled out a questionnaire based on their experience; Table 2 depicts the results. It is clear that the ubiquitous learning with CAQE is superior to the stationary learning without CAQE in terms of ease of use, accessibility, adaptability, time usage, and learning effect. Because the stationary learning always used a big screen for content and user interface display, the first group expressed more satisfaction with the information display than the ubiquitous learning group. However, the level of satisfaction with the information display in the ubiquitous learning group was still acceptable.

Taking the student's changing context and QoS requirements into consideration during learning content provisioning is crucial for intelligent ubiquitous learning. An initial user study showed that our novel and practical system offers appropriate support for content recommendation in a pervasive learning setting. We plan to address the user interface issues identified in the user study in our future work. We'll also consider shared knowledge among group members so as to recommend content to a group of students.[5]

the AUTHORS

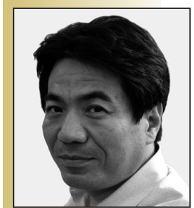
Zhiwen Yu is a research fellow at Kyoto University, Japan. His research interests include pervasive computing, context-aware systems, and personalization. Yu has a PhD in computer science from the Northwestern Polytechnical University, P.R. China. He is a member of the IEEE. Contact him at yu@ccm.media.kyoto-u.ac.jp.

Yuichi Nakamura is a professor at Kyoto University, Japan. His research interests include image understanding, video analysis, intelligent media, e-learning, and human communication. Nakamura has a PhD in electrical engineering from Kyoto University. He is a member of the IEICE, ACM, and IEEE. Contact him at yuichi@media.kyoto-u.ac.jp.



Daqing Zhang is a professor at Institut TELECOM, France. His research interests include pervasive computing, service-oriented computing, and context-aware systems. Zhang has a PhD in electrical engineering from the University of Rome "La Sapienza" and University of L'Aquila, Italy. Contact him at Daqing.Zhang@it-sudparis.eu.

Shoji Kajita is an associate professor at Nagoya University, Japan. His research interests include signal processing, speech processing, and education systems. Kajita has a PhD in information engineering from the Nagoya University. Contact him at kajita@nagoya-u.jp.



Kenji Mase is a professor at Nagoya University, Japan. His research interests include gesture recognition, computer graphics, artificial intelligence, and their applications for computer-aided communications. Mase has a PhD in information engineering from Nagoya University. He is a member of the IPSJ, IEICE, Virtual Reality Society of Japan, and ACM and a senior member of the IEEE Computer Society. Contact him at mase@nagoya-u.jp.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under the projects of "Development of Fundamental Software Technologies for Digital Archives" and "Cyber Infrastructure for the Information-Explosion Era."

REFERENCES

1. Y. Chen et al., "A Mobile Scaffolding-Aid-Based Bird-Watching Learning System," *IEEE Int'l. Workshop Wireless and Mobile Technologies in Education* (WMTE02), IEEE CS Press, 2002, pp. 15–22.
2. Z. Yu et al., "Ontology-Based Semantic Recommendation for Context-Aware E-Learning," *4th Int'l. Conf. Ubiquitous Intelligence and Computing*, LNCS 4611, Springer, 2007, pp. 898–907.
3. J.F. Sowa, *Conceptual Structures*, Addison-Wesley, 1984.
4. Z. Yu et al., "Fuzzy Recommendation towards QoS-Aware Pervasive Learning," *IEEE 21st Int'l. Conf. Advanced Information Networking and Applications*, IEEE CS Press, 2007, pp. 604–610.
5. Z. Yu et al., "TV Program Recommendation for Multiple Viewers Based on User Profile Merging," *User Modeling and User-Adapted Interaction*, vol. 16, no. 1, 2006, pp. 63–82.