Estrategias de recomendación aplicadas a repositorios de recursos educativos

Proyecto Fin de Máster en Sistemas Inteligentes

Máster en Investigación en Informática
Facultad de Informática
Universidad Complutense de Madrid

Autora: Almudena Ruiz Iniesta
Directora: Mercedes Gómez Albarrán
Codirector: Guillermo Jiménez Díaz

Curso 2008/2009

El/la abajo firmante, matriculado/a en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Estrategias de recomendación aplicadas a repositorios de recursos educativos", realizado durante el curso académico 2008-2009 bajo la dirección de Mercedes Gómez Albarrán y la codirección de Guillermo Jiménez Díaz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Madrid a 14 de Septiembre de 2009
Fdo: Almudena Ruiz Iniesta

Agradecimientos

No es posible terminar este trabajo sin dedicar unas líneas a quienes lo han hecho posible, Mercedes y Guillermo. Gracias porque desde el principio habéis compartido conmigo vuestro tiempo, esfuerzo e ilusión. Gracias por vuestras correcciones y revisiones. Por leer, releer y volver a leer todo lo que iba escribiendo. Gracias por las palabras de ánimo. Podría seguir diciendo muchas cosas más, pero en ocasiones las palabras se quedan cortas.

De una manera más formal, un agradecimiento a los autores de los recursos educativos sobre Programación que hay en el Campus Virtual de la UCM que han facilitado el uso de los mismos para este trabajo.

Y por supuesto un "¡gracias!" a todos los que en estos meses de trabajo han aportado su granito de arena. Gracias a mis *criaturillas* que hacen que todo se vea de una manera distinta. Gracias a los que han dado el toque de color *naranja,* que en algún momento ha sido muy necesario. Para ti también hay uno. Gracias a mis padres y mi hermano que siempre están conmigo.

Resumen

La abundancia de recursos disponibles en repositorios educativos plantea un nuevo reto: la necesidad de proporcionar soporte a la localización de aquellos recursos que se adapten a las necesidades, objetivos, preferencias, etc. de los estudiantes, en definitiva, a la localización de los recursos que les resulten más convenientes según el momento.

Los sistemas recomendadores nacen con el propósito de facilitar la toma de decisiones en dominios y situaciones en los que las posibilidades de elección son muchas y variadas. Actúan sugiriéndonos buenos productos y/o servicios bien sea para comprar algo o para consumir. Aunque tradicionalmente los sistemas recomendadores se han aplicado al campo del comercio electrónico, recientemente su uso ha comenzado a llevarse al campo académico.

El trabajo presentado en este Proyecto Fin de Máster se engloba precisamente dentro de esta reciente línea de investigación que afronta el traslado de técnicas de recomendación al ámbito educativo. En concreto, este trabajo aborda el uso de técnicas de recomendación como soporte al acceso personalizado de recursos educativos existentes en repositorios electrónicos.

Para poder hacer una propuesta de una estrategia de recomendación, será necesario realizar un análisis del estado del arte en los sistemas recomendadores, extrayendo así las principales características de estos sistemas. Después, se estudiarán los inconvenientes de la estrategia de recomendación de recursos educativos que ha servido de punto de partida al presente trabajo. A partir de estos propondremos las nuevas estrategias de recomendación que alivian los inconvenientes detectados. Estas propuestas se ejemplificarán en el dominio concreto de la enseñanza de la Programación.

Palabras clave. Técnicas de recomendación basadas en contenido, Learning Objects, Personalización, Diversidad.

Abstract

The abundance of educational resources available in on-line repositories inevitably poses a new challenge: providing support for locating those adapted to the individual knowledge, goals and/or preferences of the students.

Research work on recommendation technologies helps to alleviate the process of selecting information, items or resources by supporting users in pre-selecting information they may be interested in. Recommender systems are designed to address many of these problems by offering users a more intelligent approach for navigating and searching in complex information spaces. They have been traditionally applied in e-commerce however, their use has been recently transferred to the academic field.

The present work describes the use of recommendation techniques providing support for locating educational resources adapted to the student knowledge. Specifically, the use of recommendation techniques that provide personalized access to the educational resources that exist in repositories.

It will be necessary to analyze the state of the art in recommender systems, in order to draw the main features of these systems. Then we examine the drawbacks of the recommendation strategy of learning objects that served as a starting point to this work. Then we will propose new strategies to alleviate the disadvantages showed. These proposals are exemplified in the specific domain of teaching Programming

Keywords. Content-based Recommendation Techniques, Learning Objects, Personalization, Diversity.

Índice

Capítulo 1

INTRODUCCIÓN

En los últimos años, y en la mayoría de las disciplinas educativas, existe una tendencia por parte de los docentes a desarrollar contenidos educativos que son puestos a disposición de los alumnos. Los repositorios de recursos electrónicos resultantes, a los que los estudiantes tienen acceso, pretenden así facilitar y fomentar el proceso de (auto)aprendizaje.

Sin embargo, la abundancia de recursos educativos en estos repositorios es un arma de doble filo y plantea un reto adicional: es necesario proporcionar soporte a la localización de aquellos recursos que se adapten a las necesidades, objetivos, preferencias, etc. de los estudiantes, en definitiva, a la localización de los recursos que les resulten más convenientes según el momento. Pasar por alto esta necesidad puede desembocar en la infrautilización de los citados repositorios y, en consecuencia, en no conseguir el efecto positivo deseado de cara al proceso de aprendizaje, a pesar del gran esfuerzo que supone desarrollar todos los contenidos educativos allí recogidos.

En cierta medida, este efecto negativo está empezando a aparecer en nuestro entorno. Un equipo docente formado por un grupo de profesores del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid (UCM), con los que la autora del presente trabajo ha colaborado en los últimos tiempos, ha trabajado en los tres últimos años en sendos Proyectos de Innovación y Mejora de la Calidad Docente financiados por la propia universidad con el objetivo de desarrollar un repositorio de recursos didácticos para asignaturas de Programación. Este repositorio, que ha ido poblándose a lo largo de este

tiempo hasta alcanzar un volumen de cerca de 400 ejemplos resueltos y ejercicios, está accesible a través del Campus Virtual de la UCM [33]. Aunque los estudiantes valoran la ayuda proporcionada por estos recursos prácticos, lo cierto es que una gran parte de los estudiantes ha señalado que echa en falta facilidades sofisticadas para el acceso a los mismos y coincidía en la conveniencia de que fuese algún tipo de acceso guiado por el conocimiento del estudiante [34].

Desde mediados de los años 90, el trabajo de investigación en técnicas de recomendación ha tenido como objetivo precisamente aliviar la carga que supone la preselección de información, productos, recursos, etc. en los que los usuarios de los mismos pudiesen estar interesados. Así pues, los sistemas recomendadores nacen con el propósito de facilitar la toma de decisiones en dominios y situaciones en los que las posibilidades de elección son muchas y variadas. Actúan sugiriéndonos buenos productos y/o servicios bien sea para comprar algo o para consumir. Aunque tradicionalmente los sistemas recomendadores se han aplicado al campo del comercio electrónico, recientemente su uso ha comenzado a llevarse al campo académico. En este sentido, las investigaciones se han llevado a cabo en el desarrollo de herramientas de recomendación para la creación de cursos y actividades de aprendizaje [26, 41, 58] y recomendadores que sugieren a los instructores las modificaciones más apropiadas que mejorarán la eficiencia de los sistemas educativos web [27].

El trabajo presentado en este Proyecto Fin de Máster se engloba precisamente dentro de esta reciente línea de investigación que afronta el traslado de técnicas de recomendación al ámbito del e-learning. En concreto, este trabajo aborda el uso de técnicas de recomendación como soporte al acceso personalizado a recursos educativos existentes en repositorios electrónicos. Las propuestas que se realizan en él suponen una continuación natural de los trabajos iniciados recientemente dentro del Grupo de Aplicaciones de Inteligencia Artificial (GAIA) de la Universidad Complutense de Madrid en el campo de la recomendación de Objetos de Aprendizaje [31, 32]. Los resultados extraídos de este trabajo está previsto que, en un futuro próximo, redunden en un mayor y más exitoso uso del repositorio de recursos didácticos de Programación mencionado anteriormente.

De una forma más precisa, los objetivos abordados en este Proyecto Fin de Máster son los siguientes:

- Realizar una revisión del estado actual de la investigación en sistemas de recomendación. A partir de esta revisión podremos establecer un marco teórico con el que caracterizar los sistemas de recomendación.
- Analizar el trabajo previo propuesto en [31, 32], con el fin de detectar sus debilidades y explorar posibles líneas de mejora y extensión. Este análisis, junto con la toma de conciencia del estado de la investigación en recomendación

conseguida al realizar la revisión indicada en el objetivo anterior, nos colocará en disposición de abordar el siguiente objetivo.

- Diseñar uno o más enfoques genéricos de recomendación de Objetos de Aprendizaje que permitan superar las debilidades detectadas en [31, 32]. Ligado a este objetivo se encuentra el subobjetivo de reconocer y abordar las necesidades de conocimiento específicas que los nuevos enfoques requieren.
- Iniciar la aplicación de las nuevas estrategias diseñadas en el dominio de la enseñanza de la Programación, usando como Objetos de Aprendizaje a recomendar los obtenidos a partir de los recursos disponibles actualmente en el Campus Virtual de la UCM que han sido mencionados con anterioridad. Esta aplicación supondrá adaptar dichos recursos educativos al estándar Learning Object Metadata (LOM) y la definición del restante conocimiento concreto para el dominio elegido.

1.1. Estructura de la memoria

El resto del contenido de esta memoria está organizado de la siguiente manera:

- Comenzaremos el trabajo mostrando en el Capítulo 2 el estado del arte en los sistemas recomendadores. Identificaremos cuáles son las principales características de estos sistemas. Junto con la revisión del estado actual de la investigación se mostrarán algunos ejemplos de sistemas que, a nuestro juicio, reúnen las principales características. Algunos de los problemas o limitaciones de los sistemas recomendadores serán también expuestos.
- Después, en el Capítulo 3, comenzaremos describiendo de forma resumida la estrategia de recomendación de Objetos de Aprendizaje que ha servido de punto de partida al presente trabajo y expondremos los inconvenientes achacables a la misma. Atendiendo a las líneas de mejora detectadas, se presentarán las dos nuevas estrategias propuestas que ayudan a solventar los inconvenientes de la inicial. La presentación de las nuevas estrategias se hará de manera genérica, incluyendo tanto el conocimiento necesario en ambas como su funcionamiento.
- En el Capítulo 4 se explicará la aplicación de las estrategias genéricas al dominio concreto de la Programación. Explicaremos cuál ha sido el proceso de aplicación seguido, y se ilustrará el funcionamiento de las estrategias resultantes.
- En el Capítulo 5 se mostrarán las conclusiones del trabajo y las líneas de trabajo futuro.
- Finalmente, se incluye un anexo que recoge algunos ejemplos representativos de Objetos de Aprendizaje construidos de cara a la aplicación de las estrategias en el dominio de la Programación.

Capítulo 2

Estado del Arte en los Sistemas Recomendadores

Diariamente estamos expuestos a una cantidad de información que aumenta mucho más rápido que nuestra capacidad para procesarla. Es, por tanto, el momento de desarrollar tecnologías que alivien esta sobrecarga de información. Los sistemas recomendadores nacen así con el propósito de facilitar la toma de decisiones en temas/dominios en los que las posibilidades de elección son muchas y variadas. Actúan sugiriéndonos buenos productos y/o servicios bien sea para comprar algo o para consumir.

Los sistemas recomendadores emergieron como un área de investigación independiente a mediados de los 90 y tradicionalmente han sido aplicados en el campo del comercio electrónico [88] (por ejemplo, para recomendar libros, CDs y otros productos en Amazon.com [50] o películas en MovieLens [55]). Recientemente su uso ha sido llevado al campo académico. En este sentido, las investigaciones se han llevado a cabo en el desarrollo de herramientas de recomendación para la creación de cursos y actividades de aprendizaje [26, 41, 58] y recomendadores que sugieren a los instructores las modificaciones más apropiadas que mejorarán la eficiencia de los sistemas educativos web [27].

En este capítulo presentaremos una revisión del estado actual en la investigación sobre sistemas de recomendación, junto con ejemplos de algunos de los sistemas que, a nuestro juicio, reúnen las principales características de los recomendadores. Comenzaremos proporcionando un marco teórico sobre los recomendadores, distinguiendo los tipos que existen y presentando algunos sistemas recomendadores de referencia (Sección 2.1), para después, en la Sección 2.2, identificar y analizar en detalle

un conjunto de características con las que describir los sistemas de recomendación. Las principales limitaciones y problemas ligados a los sistemas recomendadores se recogen en la Sección 2.3. Terminaremos el capítulo en la Sección 2.4 con unas pequeñas conclusiones después de nuestro análisis.

2.1. Tipos de recomendadores

Existen dos grandes familias de recomendadores en base a la fuente de conocimiento usada para realizar la recomendación: *basados en contenido* [61], que son aquellos que realizan la recomendación al usuario en base a la descripción de los productos, y los *colaborativos* [69], que son aquellos que utilizan valoraciones asociadas a los productos dadas por el propio usuario y/o por otros usuarios.

Otras técnicas han sido propuestas para desarrollar recomendadores, por ejemplo, *basadas en conocimiento* [19] y recomendadores basados en perfiles *demográficos* [60] si bien han sido menos usadas.

De cara a mejorar el rendimiento de las distintas técnicas surgieron los recomendadores *híbridos* [17]. Estos recomendadores surgen como combinación de otros enfoques (por ejemplo, *basados en contenido* y *colaborativos*).

A continuación, analizaremos con más detalle los tres tipos principales de sistemas de recomendación (recomendadores basados en contenido, recomendadores colaborativos y recomendadores híbridos) e introduciremos algunos ejemplos de sistemas concretos de referencia de cada uno de ellos.

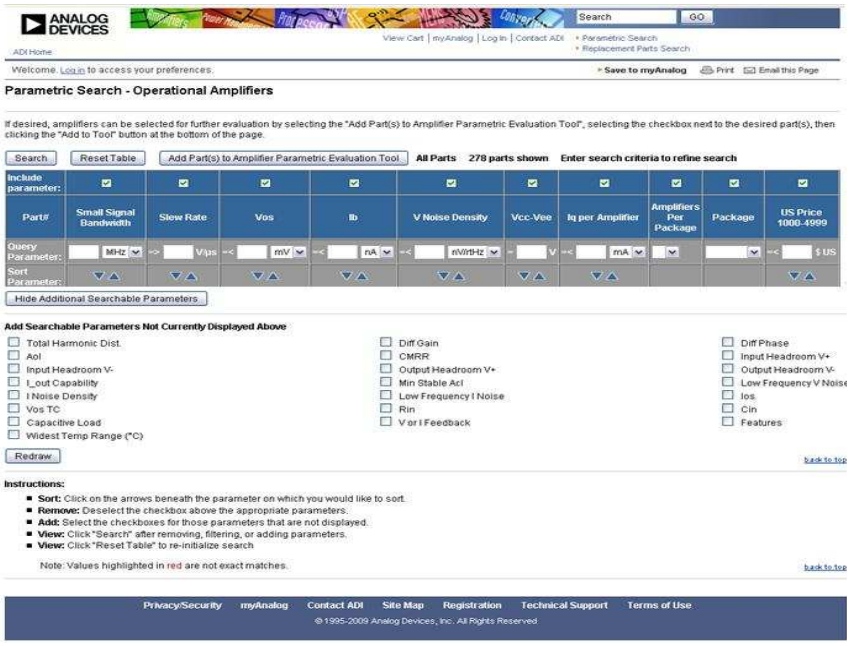
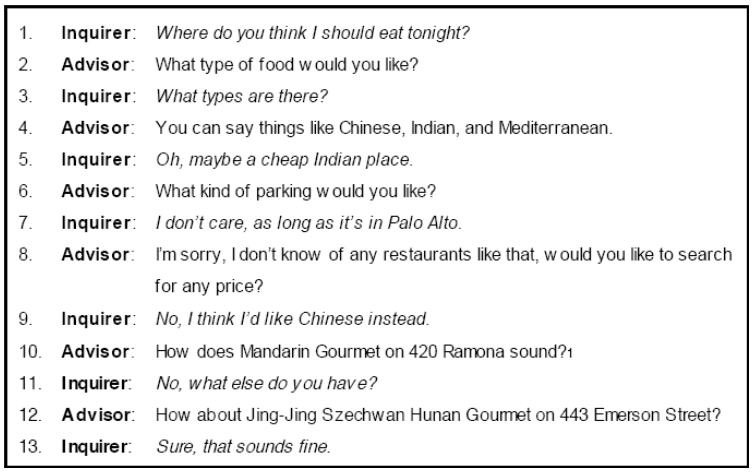


Figura 1. Vista parcial de la interfaz del recomendador de amplificadores operacionales de *Analog Devices*. (Extraída de http://www.analog.com/en/pSearch.html (último acceso: 30-8-2009)).

2.1.1. Recomendadores basados en contenido

Como ya hemos señalado, los recomendadores *basados en contenido* son aquellos que realizan la recomendación en base a la descripción de los productos a recomendar. Algunos autores sitúan dentro de los recomendadores *basados en contenido* a los *basados en casos* [76]. Según estos autores, la diferencia entre un sistema recomendador basado en contenidos y uno basado en casos radica básicamente en que en los segundos se dispone de una representación más estructurada de la descripción de los productos. Para otros autores [14], recomendadores basados en contenidos y recomendadores basados en casos son simplemente dos denominaciones diferentes de una misma familia de sistemas de recomendación, que están inspirados o tienen su origen en las técnicas de Razonamiento Basado en Casos (CBR, del inglés Case-Based Reasoning) [1].

Uno de los primeros ejemplos de sistema recomendador *basado en casos* fue el empleado por *Analog Devices* [87], uno de los principales fabricantes y distribuidores de componentes electrónicos en EEUU. Este sistema está pensado para recomendar dispositivos electrónicos (amplificadores operacionales principalmente). La Figura 1 muestra una vista parcial de la interfaz del sistema.



1.	**Inquirer**:	*Where do you think I should eat tonight?*
2.	**Advisor**:	What type of food would you like?
3.	**Inquirer**:	*What types are there?*
4.	**Advisor**:	You can say things like Chinese, Indian, and Mediterranean.
5.	**Inquirer**:	*Oh, maybe a cheap Indian place.*
6.	**Advisor**:	What kind of parking would you like?
7.	**Inquirer**:	*I don't care, as long as it's in Palo Alto.*
8.	**Advisor**:	I'm sorry, I don't know of any restaurants like that, would you like to search for any price?
9.	**Inquirer**:	*No, I think I'd like Chinese instead.*
10.	**Advisor**:	How does Mandarin Gourmet on 420 Ramona sound?1
11.	**Inquirer**:	*No, what else do you have?*
12.	**Advisor**:	How about Jing-Jing Szechwan Hunan Gourmet on 443 Emerson Street?
13.	**Inquirer**:	*Sure, that sounds fine.*

Figura 2. Un ejemplo de diálogo en *Adaptive Place Advisor*, entre un usuario (*inquirer*) y el sistema (*advisor*) en el que el usuario está intentando decidir dónde cenar. (Extraída de [84]).

Otro ejemplo de recomendador *basado en casos* es *Adaptive Place Advisor* [84], un recomendador de restaurantes en la zona de la bahía de San Francisco, EEUU. En este caso, los productos a recomendar son los diferentes restaurantes descritos por atributos tales como el tipo de cocina, la valoración, el precio, la situación, si admite reservas, si tiene parking y las opciones de pago. La Figura 2 muestra un ejemplo de diálogo mantenido en este sistema con el usuario a fin de localizar los restaurantes apropiados.

Por supuesto no podíamos dejar de hablar de *Entree*, un recomendador que sugiere restaurantes en la ciudad de Chicago [18] (véase su interfaz en la Figura 3). Entree pertenece a la familia de los sistemas *FindMe* [20] que fueron de los primeros recomendadores que consiguieron incluir retroalimentación por parte del usuario en forma de críticas a las recomendaciones propuestas. Más adelante trataremos más en profundidad estos conceptos.

Los recomendadores mencionados hasta ahora y la mayoría de los desarrollados dentro de esta familia de sistemas de recomendación siguen un enfoque muy "tradicional" a la hora de localizar los productos a recomendar y no se preocupan mucho por incluir diversidad entre los productos recomendados. Habitualmente, los recomendadores *basados en casos* recuperan aquellos productos que son más similares a las necesidades manifestadas por el usuario (ya sea a través de una consulta directa, ya sea a través de una conversación mantenida con el sistema, etc.). Si a la hora de recuperar la prioridad se pone exclusivamente sobre la similitud, un recomendador *basado en casos* "tradicional" ignorará la diversidad del conjunto recuperado

Figura 3. Interfaz de *Entree*. (Extraída de [18]).

proporcionando productos muy similares entre sí. Por ejemplo, imaginemos que un usuario desea encontrar un destino vacacional de playa para el mes de mayo en España, que tenga buen ambiente por la noche y lugares de recreo. Con bastante probabilidad las primeras recomendaciones podrían estar situadas en la Costa del Sol. Si el usuario descarta este destino, es probable que dentro de las opciones mostradas no se encuentren alternativas: todos los productos recomendados son muy similares a la consulta, y también lo serán entre ellos.

Esta aproximación "tradicional" funciona bien en muchos dominios, pero se han planteado otras iniciativas, como la de [80] en la que proponen incluir diversidad en los productos sugeridos por los sistemas recomendadores sin perder el compromiso con la similitud. Los resultados de la alternativa planteada en [80] han sido probados en *FlickFinder*, un recomendador de películas mediante el uso de dispositivos móviles (WAP-based). Detalles sobre la forma de interactuar con el sistema aparecen en la Figura 4.

Otro sistema preocupado por la inclusión de diversidad es *ExpertClerk*, un recomendador diseñado para imitar las interacciones típicas observadas en la vida real entre un vendedor de camisas y un cliente [72] (Figura 5). Más adelante en la Sección 3 analizaremos con detalle algunas de las propuestas sobre diversidad que existen, pero todas están relacionadas con la inclusión de una nueva medida de calidad en los productos que de alguna forma refine la similitud.

Figura 4. Interfaz de *FlickFinder*. (Extraída de [80]).

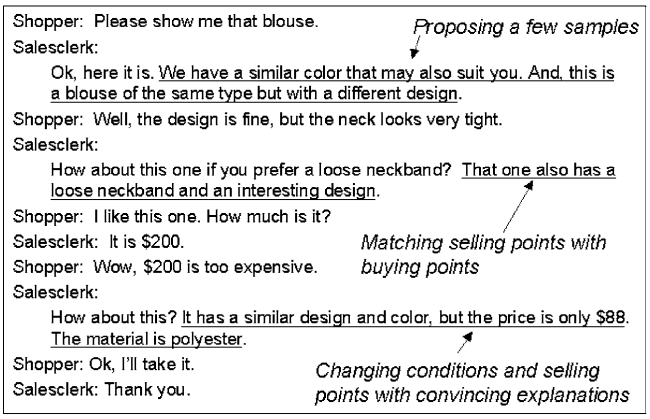


Figura 5. Un ejemplo de conversación en *ExpertClerk* entre un comprador y un vendedor de camisas. (Extraída de [73]).

Para terminar esta sección citaremos algunos otros sistemas *basados en contenido* o *basados en casos* estudiados durante la revisión realizada. *Sermo* [13], *Personal Travel Assistant* [21] y *CASPER* [11] siguen una aproximación *basada en casos*. El primero ayuda al usuario a elegir una casa de alquiler vacacional, el segundo recomienda viajes, y, por último, *CASPER* es un sistema de contratación en línea que sugiere posibles trabajos a un usuario. *Gixo* [9] es un recomendador de noticias basado en contenido. Como base de sus recomendaciones utiliza similitud textual. Los siguientes sistemas

pertenecen todos ellos a la familia de los sistemas *FindMe* [20] aunque cada uno de ellos tiene un dominio de aplicación distinto: *Car Navigator* ayuda al usuario a elegir un coche nuevo; *Video Navigator* y *PickAFlick* recomiendan películas para alquilar; *RentMe* ayuda a encontrar apartamentos; y por último *Kenwood* recomienda configuraciones para un sistema de sonido en casa.

2.1.2. Recomendadores colaborativos

Los recomendadores *colaborativos* utilizan la técnica de filtrado colaborativo (CF, del inglés *Collaborative Filtering*). Es, probablemente, la técnica más empleada en los sistemas recomendadores propuestos en la literatura. El filtrado colaborativo es el proceso por el que se evalúan y se seleccionan los productos a partir de las opiniones de otros usuarios [69]. El término filtrado colaborativo se ha comenzado a utilizar en la última década, pero el concepto que subyace es algo que las personas han hecho desde siempre: compartir opiniones entre sí (el tradicional *boca a boca*). Cuando vamos a ver una película o decidimos comprar un libro, siempre preguntamos a amigos sus opiniones, o incluso les pedimos que nos recomienden lo último que han adquirido ellos, y así filtramos la información en base a las opiniones de nuestros conocidos. A diferencia de los métodos basados en contenido, el filtrado colaborativo a la hora de ofrecer una recomendación a un usuario no considera únicamente sus preferencias personales, sino también las de otros usuarios con intereses similares a los suyos. Tal similitud se estima a partir de las valoraciones definidas en sus perfiles personales. De ahí que los enfoques colaborativos no requieran la descripción de los productos.

Una de las clasificaciones para las estrategias de CF es la propuesta por [69], que distingue dos tipos:

- CF basado en usuario (*user-based*). Sugiere a cada usuario aquellos productos que han interesado a sus vecinos. Para formar este grupo de usuarios vecinos, la estrategia considera que dos usuarios tienen preferencias similares si han clasificado los mismos productos en sus perfiles y les han asignado índices de interés parecidos. Entonces, dos usuarios se consideran vecinos si tienen preferencias similares.
- CF basado en producto (*item-based*). Un producto es recomendado a un usuario si dicho producto es similar a los incluidos en su perfil de usuario. En este caso, se considera que dos productos son similares (o vecinos) si los usuarios que han valorado uno de ellos tienden a valorar el otro asignándole índices de interés parecidos.

En ambas técnicas, el objetivo es predecir el nivel de interés del usuario en relación a un producto dado y, en función de del citado interés, sugerirlo (o no) a dicho usuario. En este proceso, se identifican dos fases:

1. En los enfoques colaborativos basados en usuario, la primera fase selecciona los usuarios cuyas preferencias son similares a las del usuario activo. Por el contrario, los enfoques basados en producto extraen las preferencias del usuario que son más similares al producto objetivo. A continuación, se forma el vecindario, ya sea del usuario o de los productos definidos en su perfil, que estará formado por los usuarios o productos más relevantes.
2. En la segunda fase, el sistema debe predecir el nivel de interés del usuario en relación al producto objetivo. Para ello, los enfoques basados en usuario consideran el nivel de interés de los vecinos del usuario activo en relación al producto objetivo. Por el contrario, la versión colaborativa basada en producto considera el nivel de interés del usuario en relación a aquellos productos de su perfil que son más similares al objetivo.

El primer recomendador colaborativo que usaremos como referencia es *MovieLens*[1], un recomendador de películas [55]. Un usuario de MovieLens valora películas en una escala de 1 a 5 estrellas, siendo 1 "Malísima" y 5 "Debe verse". Entonces el sistema utiliza las valoraciones de la comunidad para recomendar otras películas en las que el usuario podría estar interesado. Podemos ver la interfaz de MovieLens en la Figura 6.

La librería Amazon.com dispone también de uno de los más conocidos y utilizados recomendadores colaborativos en el ámbito del comercio electrónico. Utiliza una aproximación de filtrado colaborativo basada en productos [50]. Cada producto adquirido y valorado por un usuario lo empareja con productos similares, y después combina aquellos productos similares en una lista de recomendación. Para determinar la pareja más similar para un producto dado, el algoritmo construye una tabla de similitud de productos para encontrar productos que los usuarios suelen adquirir juntos. Para calcular la similitud entre productos el algoritmo utiliza la fórmula del ajuste del coseno. En la Figura 7 podemos ver la interfaz de este sistema.

[1] http://www.movielens.org (último acceso: 30-8-2009).

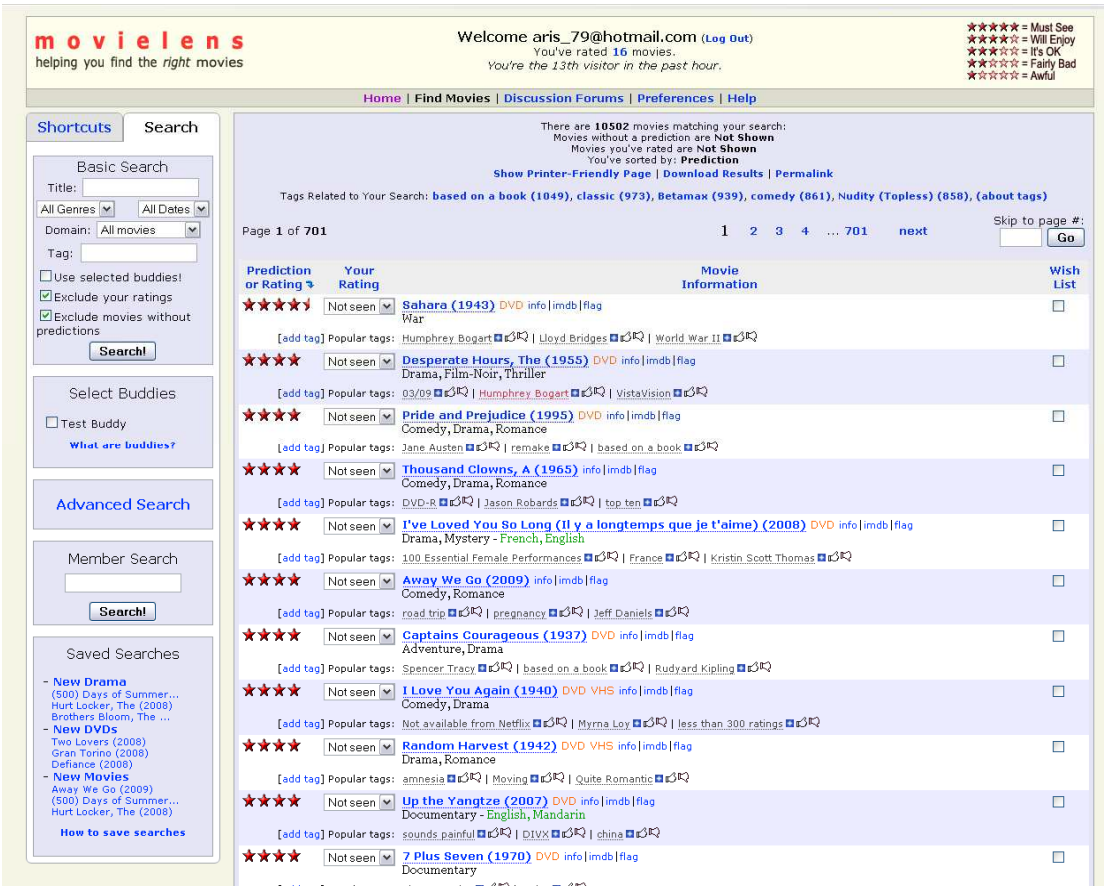


Figura 6. *MovieLens* utiliza el filtrado colaborativo para predecir que la película "Pride and Prejudice (1995)" tiene una puntuación de 4 estrellas para el usuario. (Extraída de http://www.movielens.org).

Tapestry fue el primer sistema que incorporó las opiniones de los usuarios a una base de datos de mensajes y a un buscador [29]. Este sistema recomienda listas de noticias, boletines electrónicos de artículos a usuarios de correo electrónico. *Tapestry* almacenaba el contenido de los mensajes junto con metadatos sobre los lectores, autores y los usuarios que respondían a los mensajes. También permitía que cualquier usuario almacenara anotaciones sobre los mensajes, del estilo "artículo interesante" o "Marta debería ver esto". Los usuarios del sistema podían hacer consultas que combinaban información textual (por ejemplo, que contenga la frase "sistema recomendador") con consultas semánticas de metadatos (por ejemplo, "escrito por Alfredo" o "en respuesta a Jesús") y consultas anotadas (por ejemplo, marcado como "excelente" por Rosa).

En la familia de sistemas recomendadores colaborativos los métodos más utilizados a la hora de medir la similitud entre los perfiles de varios usuarios son las técnicas de selección de los vecinos más cercanos (es decir, usuarios con un historial de

Figura 7. Ejemplo de recomendación de Amazon.com. (http://www.amazon.com (último acceso: 30-8-2009))

valoraciones sobre los productos similar al historial del usuario actual) y clustering. Además, la primera es también la estrategia más extendida entre los enfoques colaborativos basados en producto.

El método de los *vecinos más cercanos* se utiliza para detectar similitud entre las preferencias de los usuarios en los recomendadores colaborativos (tanto en los basados en usuario como en los basados en producto). En los enfoques basados en usuario, cada perfil se representa mediante un vector, cuyas componentes son las clasificaciones que el usuario ha asignado a cada uno de los productos incluidos en el mismo. Una vez modelados los usuarios, se aplican sobre sus respectivos vectores métricas como la similitud basada en coseno (1) o la correlación Pearson (2), detectando así parecidos entre sus preferencias.

$$similitud_{\cos}(A, B) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| \cdot |\vec{B}|} \qquad (1)$$

$$corr_{Pea}(\vec{A}, \vec{B}) = \frac{\sum_r (\vec{A}[r] - \overline{A}) \cdot (\vec{B}[r] - \overline{B})}{\sqrt{\sum_r (\vec{A}[r] - \overline{A})^2 \cdot \sum_r (\vec{B}[r] - \overline{B})^2}} \qquad (2)$$

En las ecuaciones (1) y (2) A y B son los valores medios de las clasificaciones definidas por los dos usuarios cuyas preferencias son comparadas (y cuyos niveles de

interés se registran en los vectores A y B, respectivamente). De acuerdo a (2), es evidente que cuantos más productos hayan clasificado a la vez los usuarios comparados, y cuanto más parecidas sean las clasificaciones asignadas a los mismos, mayor será la correlación detectada entre sus preferencias.

Por su parte, en lugar de computar la similitud entre dos usuarios concretos, los enfoques colaborativos basados en producto calculan la similitud entre dos productos. Para ello, este tipo de propuestas seleccionan los usuarios que han clasificado a la vez los productos comparados, y construyen sendos vectores a partir de los niveles de interés que éstos han asignado a ambos productos. Finalmente, basta aplicar sobre estos vectores las métricas antes mencionadas, para así obtener los valores de similitud concretos entre los productos considerados [68]. Razonando de forma análoga a la descrita en los sistemas colaborativos basados en usuario, es evidente que cuantos más usuarios hayan clasificado a la vez los dos productos comparados y cuanto más parecidos sean los niveles de interés asignados a los mismos, más significativo será el valor de similitud medido entre ambos.

Otro de los métodos más utilizados para valorar la similitud entre usuarios es el conocido como *clustering*. Orwant propuso en 1995 un método de clustering basado en definir grupos de usuarios de forma dinámica a partir de los perfiles individuales disponibles en el sistema [59]. Las características que comparten varios usuarios son utilizadas precisamente como estereotipos, de forma que una vez identificados los grupos de usuarios comunes, el sistema predice el interés de cada uno de ellos promediando las clasificaciones del resto de usuarios pertenecientes a su mismo grupo. De acuerdo a los resultados obtenidos por Breese [12], las recomendaciones elaboradas mediante los métodos basados en *vecinos más cercanos* son más precisas que las obtenidas mediante el *clustering*, hecho que limita la aplicación de esta última técnica en los enfoques colaborativos actuales.

En la literatura existen otras clasificaciones para los algoritmos empleados en los sistemas basados en CF. Podemos citar, por ejemplo la propuesta por [12], según la cual los algoritmos de filtrado colaborativo pueden agruparse en dos grandes clases: los algoritmos basados en memoria (*memory-based*) y los algoritmos basados en modelo (*model-based*). Los algoritmos basados en memoria utilizan toda la base de datos de productos y usuarios para generar predicciones. Primeramente emplean técnicas estadísticas para encontrar a los *vecinos*. Una vez que se ha construido una lista de vecinos se combinan sus preferencias para generar una lista con los N productos más recomendables para el usuario actual. Los algoritmos basados en modelo desarrollan primero un modelo de las valoraciones del usuario. Tratan el problema como un problema de predicción estadística y calculan el valor esperado para cada elemento en función de las valoraciones anteriores. Para ello se utilizan distintos algoritmos de aprendizaje máquina (por ejemplo, algoritmos de clustering o redes Bayesianas). En

general, ante las consultas responden más rápido que los basados en memoria, pero por contra necesitan de un proceso de aprendizaje intensivo

Para terminar esta sección citaremos algunos otros sistemas colaborativos estudiados durante la revisión realizada. Comenzaremos citando *TiVo*, -un recomendador colaborativo de programas de televisión- [44]. Este recomendador utiliza la técnica de filtrado colaborativo basada en producto. *GroupLens* [64] -un recomendador de noticias en Usenet-, *Ringo* -un recomendador de música- [86] y *Bellcore's Video Recommender* [39] en el dominio de la música, son sistemas recomendadores que utilizan una aproximación basada en usuario. Todos ellos emplean la técnica de vecinos más cercanos para valorar la similitud entre perfiles de usuario, al igual que lo hace *Movielens* [55].

2.1.3. Recomendadores híbridos

Un recomendador híbrido es aquel que combina múltiples técnicas en un único sistema consiguiendo una participación activa de todas ellas. Burke [17] define hasta siete métodos diferentes para combinar estrategias de recomendación:

- **Ponderado.** El resultado de las distintas técnicas de recomendación que componen el sistema se combinan y se obtiene una puntuación para cada producto en base a la puntuación asignada a cada uno de ellos por las distintas estrategias. El producto (o subconjunto de productos) con mayor puntuación será el que se ofrezca al usuario. Cada una de las estrategias pueden tener asociados distintos pesos en la combinación final. Un ejemplo de recomendador híbrido ponderado es *Movie Recommender*, descrito en [56]. Se trata de un recomendador de películas con dos componentes: uno que utiliza técnicas de filtrado colaborativo para así comparar entre perfiles de usuario la similitud entre sus preferencias; el otro usa la información sobre las características de las películas y recomienda aquellas cuyas características coinciden con los gustos del usuario. La salida de estos dos componentes se combina usando una función ponderada lineal.
- **Conmutación**. En este caso, en lugar de ejecutar todas las estrategias simultáneamente, el sistema emplea algún criterio para conmutar entre ellas: cuando se cumplen ciertas condiciones el sistema emplea una estrategia y, en caso contrario, recurre a la(s) restante(s). Un ejemplo de sistema híbrido que utiliza conmutación es *NewsDude*, un recomendador que ofrece artículos de noticias a los usuarios [8]. Este sistema se compone de tres recomendadores: dos basados en contenido y uno colaborativo. Los tres componentes están ordenados, si el primero no produce una recomendación fiable, se pasa al segundo, y así hasta el tercero. En la Figura 8 podemos ver la interfaz de este sistema.

Figura 8. Interfaz de *NewsDude*. (Extraída de [8]).

- **Mixto**. Este esquema reúne en una misma recomendación productos que han sido sugeridos mediante las diferentes estrategias implementadas en el sistema híbrido. Un ejemplo de sistema híbrido mixto es *PTVPlus*, un recomendador de programas en televisión digital [77]. *PTVPlus* se compone de dos recomendadores, uno basado en contenido y otro colaborativo, en el que el resultado final será una combinación de los resultados producidos por ambos.
- **Combinación de características.** En este modelo se funden en un único conjunto los datos que utilizan las diferentes estrategias, y con este nuevo conjunto se ejecuta un solo algoritmo de recomendación. Por ejemplo, Basu, Hirsh y Cohen [4] proponen un recomendador de películas basado en contenido que construye los modelos de usuario, y a continuación las valoraciones de los usuarios son combinadas con las características de los productos.
- **Cascada**. Un sistema híbrido en cascada funciona en dos etapas. Primero se ejecuta una de las estrategias de recomendación y obtiene un primer conjunto de productos candidatos a ser incluidos en la recomendación final. A continuación, una segunda estrategia refina la recomendación y selecciona sólo algunas de las sugerencias obtenidas en la primera etapa. Como ejemplo de recomendador en cascada tenemos *EntreeC* [17], creado añadiendo a *Entree* [18] un módulo colaborativo de marcaje-ordenación para aquellos productos con iguales valores. Este recomendador soluciona un problema encontrado en su predecesor. La devolución de muchos productos con iguales valoraciones, que no podían ser valorados y ordenados en relación con el resto de productos. La estrategia

diseñada en [32] también se corresponde con una estrategia híbrida, en este caso aplicada al dominio de la recomendación de recursos educativos. La primera etapa corresponde a un recomendador basado en casos, que es refinado en una segunda etapa por un recomendador colaborativo.

- **Incorporación de características**. En este esquema, una de las estrategias genera una recomendación junto con una característica nueva para el producto, por ejemplo, una característica del tipo: "productos relacionados". A continuación, esa información se incorpora como una característica más de ese producto para las siguientes técnicas de recomendación; en otras palabras, la salida de una de las estrategias de recomendación se incorpora como una característica más que es utilizada por la siguiente estrategia de recomendación. Melville, Mooney y Nagarajan [53] describen un recomendador híbrido que primero a través de un recomendador basado en contenido y un conjunto de datos de entrenamiento genera valoraciones para usuarios que todavía no disponen de estas. Tanto, estas valoraciones generadas como las asignadas directamente por los usuarios, son utilizadas por un recomendador colaborativo. En [57] se describe un sistema de recomendación de libros basado en contenido. Este sistema de recomendación extrae información sobre libros a partir de la información que aparece en *Amazon*. Esta información contiene las recomendaciones que *Amazon* hace al usuario, del tipo "títulos relacionados" o "Autores que también pueden ser de interés". Así pues, el sistema de recomendación con incorporación de características, además de obtener como elementos de entrada para el aprendizaje del sistema las propiedades del producto recomendado (como puede ser autor de la obra, tipo de obra o número de páginas), también incorpora las recomendaciones, en este caso del tipo "títulos relacionados" o "Autores que también pueden ser de interés", a su motor de aprendizaje.
- **Metanivel**. En este caso, el modelo completo generado por una de las estrategias se utiliza como entrada en las otras existentes en el recomendador híbrido. La diferencia entre este método de recomendación híbrida y el basado en incorporación de características es que, en este último, el modelo aprendido sólo se utiliza para generar valores de características que se usan como entrada en las siguientes estrategias, mientras que en el metanivel se utiliza todo el modelo como dato de entrada. *Fab* [3] es un recomendador de documentos que usa la estructura *"collaboration through content"* propuesta por Pazzani [60], que utiliza un recomendador basado en contenido para construir los modelos de usuario y después un módulo colaborativo que filtra los usuarios.

2.2. Características de los recomendadores

En esta sección presentaremos algunas de las características encontradas en los sistemas recomendadores. Hemos hecho una clasificación de las mismas ya que cada una tiene que ver con distintas funcionalidades o aspectos de diseño del recomendador. Más adelante analizaremos en detalle cada una de esas características con un ejemplo concreto de sistema recomendador.

Las características han sido obtenidas después de la revisión de un gran número de artículos sobre sistemas recomendadores de los últimos años. Estas características las hemos asociado con los siguientes aspectos de diseño: (a) las decisiones acerca de quién lleva la iniciativa en la recomendación, (b) la estrategia de interacción usuario-recomendador empleada, (c) la estrategia de selección seguida, (d) el nivel de personalización incluido, y (e) la inclusión de facilidades que permitan aumentar la confianza en la recomendación.

En las siguientes subsecciones se analizan cada uno de estos aspectos y las distintas formas en las que se han plasmado en los trabajos científicos.

2.2.1. Iniciativa en la recomendación

Podemos distinguir dos tipos de recomendadores en base a quién lleva la iniciativa en la recomendación. Así, podemos tener un recomendador *reactivo*, en el que es el usuario quien lleva la iniciativa realizando una consulta al sistema. En el otro lado están los recomendadores *proactivos*. En este caso el que lleva la iniciativa es el recomendador, que realizará una propuesta inicial al usuario basándose en el historial pasado del usuario, en valoraciones asociadas a los productos, o en cualquier otro dato relevante. Habitualmente, en este segundo caso entre usuario y sistema se mantiene una conversación en la que el usuario puede ir indicando al sistema hacia dónde se dirigen sus intereses.

Un ejemplo de recomendador reactivo es *Analog Devices* [87], donde el usuario debe rellenar un extenso formulario indicando los parámetros del dispositivo electrónico que está buscando. *Entree* [18] también opera de manera reactiva, haciendo recomendaciones basadas en una consulta en la que el usuario ha especificado características tales como el tipo de cocina, el precio, el lugar, etc. relativos al restaurante al que le gustaría acudir.

En los recomendadores reactivos la consulta puede realizarse de múltiples maneras. Hay sistemas que presentan un formulario con múltiples características de los productos. El usuario rellena dicho formulario con sus preferencias usando un vocabulario controlado de posibles valores para cada característica, como ocurre en [20,

87]. Por otro lado existen otros recomendadores reactivos que muestran recomendaciones sobre una característica; por ejemplo, "recomiéndame películas por género". Tal es el caso de *Entree* [18] que permite buscar el restaurante que más nos guste en función del tipo de cocina, o el precio. En estos sistemas el usuario sólo debe indicar una característica del producto para la recomendación. Otros sistemas cuentan con sofisticados módulos de procesamiento de lenguaje natural que hacen posible que el usuario escriba su consulta en texto libre usando lenguaje natural [71, 84].

En el otro lado tenemos *PTVPlus* [77] que hace recomendaciones de programas de televisión en base a las preferencias que el sistema ha aprendido del usuario, sin necesidad de que éste formule ninguna consulta directa. El recomendador de Amazon.com [50] también funciona de una manera proactiva, el usuario tiene la opción de seleccionar la opción "mis recomendaciones" y en base al perfil aprendido por el sistema se le mostrará un conjunto de productos.

Tanto Amazon.com [50] como *PTVPlus* [77] contemplan la posibilidad de funcionar de manera reactiva, bien sea solicitando recomendaciones de una categoría de productos o rellenando una consulta. Si el usuario selecciona, por ejemplo, "recomiéndame productos de bebés", el sistema le mostrará aquellos productos en base a su perfil que se adaptan a sus características.

2.2.2. Estrategia de interacción usuario-recomendador

En cuanto a la interacción que se produce entre el usuario y el recomendador distinguimos dos estrategias: *single-shot* y *conversacional* [76].

Los recomendadores *single-shot* son aquellos en los que se muestra un conjunto de productos recomendados al usuario y éste tiene la oportunidad de elegir uno o descartarlos. Si la recomendación no agradara al usuario, éste debería empezar de nuevo para obtener nuevos productos. Es decir, en una solicitud de recomendación el usuario no puede refinar sus requisitos.

Analog Devices [87] es un ejemplo de recomendador *single-shot*. En la recuperación el sistema muestra los 10 mejores resultados que cumplen los contenidos de la consulta. Si ninguno de estos resultados satisface al usuario, éste deberá rellenar el formulario de nuevo, pudiendo incluso asignar prioridad a los atributos que considere más importantes. Finalmente, cuando el usuario consigue el dispositivo que busca puede acceder a las especificaciones detalladas del mismo mediante un enlace.

La interfaz reactiva de *PTVPlus* [77], al igual que *Analog Devices*, funciona con una estrategia *single-shot*. En un principio el usuario selecciona un interés, por ejemplo "comedia americana", y obtendrá como resultado series de televisión que se encuentran en la programación que pudieran interesarle. Además en esta muestra de programas que

se le ofrecen tiene la posibilidad de valorarlos proporcionando así al sistema información sobre sus gustos.

Los recomendadores *conversacionales* son aquellos en los que la recomendación se entiende como un proceso iterativo en el que el usuario puede ir refinando sus requisitos hasta obtener productos adecuados para él. Básicamente, existen dos estrategias de conversación: *navegación por propuesta* o *navegación por pregunta*. En la primera, un conjunto de productos son propuestos al usuario y el sistema obtiene retroalimentación del usuario con la que refinar sus requisitos. En la *navegación por preguntas* el sistema recoge los requisitos del usuario a partir de la formulación de un conjunto de preguntas cuidadosamente seleccionadas en forma y en tiempo.

En los recomendadores de *navegación por propuesta* la retroalimentación por parte del usuario es un punto clave. De manera general, existen tres formas de proporcionar esta retroalimentación:

- *En forma de valoraciones*, como hacen en [17, 78, 79]. Se trata de involucrar al usuario solicitando valoraciones para cada uno de los productos recuperados. Esta técnica es especialmente utilizada en los recomendadores colaborativos.Para más información puede verse [69].
- *Basado en críticas*, como se hace en *Entree* [18]. En este método el usuario expresa sus restricciones en alguna de las características.
- *Basado en preferencias*, como hacen en [76, 91]. En este simplemente un usuario expresa su preferencia por un producto frente a otros.

El método de *navegación por pregunta* es, sin duda, la forma más directa de elaboración de los requisitos de un usuario y puede conducir a diálogos muy eficaces. *Adaptive Place Advisor* [84] emplea este tipo de interacción. Este sistema sigue una aproximación que denominan *frame-based* [10]: las preguntas pueden provenir del usuario o del sistema, lo cual añade complejidad al manejo de la entrada del usuario que puede realizar las preguntas que desee mediante la introducción de texto libre. El diálogo entre el recomendador y el usuario está guiado por una máquina de estados. En cada paso de diálogo el sistema actualizará las variables de estado, haciendo que pase de un estado a otro.

Una de las cuestiones clave que debe abordarse en la *navegación por pregunta* se refiere a la cuestión de qué tipos de preguntas se debe hacer en cada paso de la conversación. El orden y número de preguntas pueden tener un impacto significativo en la facilidad de uso y esfuerzo por parte del usuario.

Doyle & Cunningham fueron de los primeros que obtuvieron resultados experimentales en este tema dentro del área de CBR [22], mediante la evaluación de diferentes criterios de selección. Entre ellos se incluía un método basado en entropía, el

cual evaluaba la ganancia de información de unas características dadas y se inspiraba el método de construcción de árboles de decisión fruto del trabajo de [62, 63].

Schmitt [70] propuso una aproximación alternativa llamada *simVar*, que fue especialmente hecha a medida para recomendadores en el ámbito del comercio electrónico. En lugar de utilizar la ganancia de información basada en la entropía, este enfoque considera la influencia que una determinada característica puede tener sobre la similitud de productos.

Una nueva línea de trabajo se ha abierto en el área de la conversación, produciendo innovaciones, como la conocida iniciativa mixta, que propone modelos de interacción entre el usuario y el recomendador de una manera más flexible [13]. Un ejemplo de sistema que utiliza esta iniciativa mixta es *ExpertClerk* [71]. Dicho sistema implementa una combinación de *navegación por pregunta* y *navegación por propuesta*.

Los sistemas que emplean iniciativa mixta necesitan mantener un modelo mucho más sofisticado que tenga información sobre el estado en el que se encuentra la conversación y cada uno de sus integrantes, y además necesita tener definido un buen protocolo de comunicación que facilite el intercambio de información entre usuario y sistema. Por ejemplo, *Sermo* [13] utiliza también conversación con iniciativa mixta.

2.2.3. Estrategia de selección

A la hora de seleccionar los productos a recomendar hay sistemas que siguen una aproximación "tradicional" basada en la similitud con o entre productos, o con o entre usuarios, y recomendadores que apuestan por innovar en la estrategia de selección introduciendo una medida de calidad que tenga en cuenta otros aspectos aparte de la similitud.

Analog Devices [87] es un claro representante del enfoque "tradicional". Para evaluar la similitud entre la consulta y las características de los dispositivos electrónicos primero aplica una función de similitud local entre cada par de parámetros. A continuación estas similitudes locales se usarán para calcular la similitud global. La similitud global será la suma ponderada de las similitudes locales. La influencia de cada parámetro en la suma es asignada por un experto en dispositivos, pero siempre puede ser modificada por el usuario.

La mayoría de los trabajos que introducen estrategias de selección más refinadas combinan la similitud con medidas que valoran la diversidad entre los productos seleccionados. De esta forma se prima no sólo a los productos que por similitud permiten satisfacer mejor los intereses o preferencias del usuario sino aquellos que, al mismo tiempo, permiten que el conjunto de productos sugeridos finalmente sea lo más diverso posible. En [80] se proponen alternativas para mantener el compromiso con la

similitud, pero a la vez introduciendo diversidad. Más adelante volveremos sobre este punto.

PTVPlus [77] introduce de una manera sencilla la diversidad a la hora de hacer recomendaciones. Para ello añade un módulo de filtrado colaborativo que filtrará las recomendaciones teniendo en cuenta la opinión de otros usuarios. Esto se hace comparando la similitud entre usuarios. De este modo, las recomendaciones se basaran en productos que le gustaron en un pasado al usuario, y programas (productos) que otros usuarios similares a él vieron. Para determinar si dos usuarios son similares se computa la similitud entre perfiles de usuario mediante la siguiente ecuación.

$$PRFSIM(u,u') = \frac{\sum_{p(u)\cup p(u')} \left| r(p_i^u) - r(p_i^{u'}) \right|}{4 \cdot \left| p(u) \cup p(u') \right|}$$

donde, $p(u)$ y $p(u')$ son los programas marcados en cada perfil de usuario, y $r(p_i^u)$ es la valoración asociada al programa p_i por el usuario u. En el perfil de usuario se almacena información sobre la hora preferida de ver la televisión, programas y géneros favoritos, guía favorita y algún dato más.

Una vez que se han obtenido los k perfiles más similares, se elabora una lista de recomendación a partir de los programas que se encuentran en los perfiles similares que no están en el perfil actual. Esta lista es ordenada con las valoraciones, y los r primeros programas son seleccionados para la recomendación.

En [16] se define la diversidad del conjunto de productos a recomendar $c_1,...c_n$ como se indica a continuación:

$$Diversidad(c_1 \ldots c_n) = \frac{\sum_{i=1..n} \sum_{j=1..n} (1 - Similitud(c_i, c_j))}{\frac{n}{2} * (n-1)}$$

y se exponen tres alternativas para incluir esta forma de diversidad en recomendadores reactivos.

La estrategia más simple que proporciona diversidad al conjunto recuperado es la *selección aleatoria* (*Bounded Random Selection*), que consiste en seleccionar k productos aleatoriamente a partir de un conjunto de $b \cdot k$ productos similares a la consulta, con $b > 1$.

Una mejor opción para incluir diversidad y mantener el compromiso con la similitud es el método de *selección voraz* (*Greedy Selection*). Este método construye en cada paso

un conjunto *R* de productos recomendables. Durante cada paso, los restantes productos candidatos a ser recomendados son ordenados en función de su *calidad*, y aquel con mayor calidad se añadirá a *R*. La clave de este método es incluir una métrica de calidad que combina similitud y diversidad. Así, la calidad de un producto *c* es proporcional a la similitud entre *c* y la consulta *t*, y la diversidad de *c* relativa a los productos que ya se encuentran en *R*:

$$Calidad(t, c, R) = Similitud(t, c) * DiversidadRel(c, R)$$

$$DiversidadRel(c, R) = 0 \qquad si\ R = \{\};$$

$$DiversidadRel(c, R) = \frac{\sum_{i=1..m} (1 - Similitud(c, r_i))}{m} \qquad en\ otro\ caso$$

En el algoritmo descrito, se elige como primer producto a incluir en *R* al más similar a la consulta. En cada ciclo del proceso iterativo el siguiente producto que se incluye en *R* será aquel que sea más similar a la consulta y a la vez guarde mayor diversidad con el conjunto de productos ya seleccionados. El problema de este algoritmo es su coste, que es demasiado elevado.

Para reducir la complejidad de cálculos impuesta por la anterior alternativa está la *Selección voraz acotada* (*Bounded Greedy Selection*). En este caso el algoritmo primero selecciona los *bk* productos más similares a la consulta y luego aplica sobre estos la selección voraz antes explicada. Por supuesto esta ganancia de eficiencia tiene un coste: podemos perder algún producto que aún teniendo una menor similitud con la consulta tiene una gran diversidad con el resto de productos, lo cual compensaría una baja similitud y haría que pudiese entrar en el conjunto de productos elegidos para ser mostrados al usuario.

ExpertClerk también apuesta por incluir diversidad en sus recomendaciones, pero lo hace de una manera distinta a la propuesta por [80]. *ExpertClerk* propone tres alternativas al usuario. Estos tres productos se seleccionan a partir del conjunto *R* que contiene a los *k* productos recuperados. El primero de ellos será el producto que más se aproxime a la mediana del conjunto recuperado. La mediana se calcula con la siguiente fórmula:

$$r_{med} = \left(\frac{1}{k}\sum_{j=1}^{k} f_{j1}, \frac{1}{k}\sum_{j=1}^{k} f_{j2}, \ldots, \frac{1}{k}\sum_{j=1}^{k} f_{jk}\right)$$

donde f_{ji} es el valor del atributo *j* de un determinado producto r_i. A continuación se seleccionan el segundo y tercer producto a partir del cálculo de las características

positivas y negativas. Para cada característica de cada producto recuperado se calcula la distancia entre dicho producto y la mediana: si el valor obtenido supera un cierto umbral esta característica se marca como positiva, en caso contrario como negativa. El número total de características será la suma de todas las características, positivas y negativas. El producto que tiene el mayor número de características será seleccionado como segundo producto, y el segundo producto con mayor número de características será presentado como tercera opción.

Otro enfoque interesante es el algoritmo DCR-1 propuesto en [52]. En este trabajo, ante una consulta *Q* formulada por el usuario la colección de productos se divide en capas de similitud. En cada capa se agrupan aquellos productos que tienen igual similitud con respecto a *Q*. La primera capa es la de mayor similitud, esto es, aquella cuyos productos son los que mejor satisfacen *Q*. Cada capa contendrá los productos que satisfacen *Q* mejor que lo que lo hacen los productos de la capa siguiente.

Una vez que han sido reorganizados todos los productos en capas, el algoritmo DCR-1 se encarga de seleccionar los *k* productos que serán mostrados al usuario fomentando la existencia de diversidad. Este algoritmo actúa en tres pasos:

- Paso 1: determina el conjunto *CSP* de *k* productos que serían recomendados utilizando una métrica de similitud pura.
- Paso 2: identifica la capa de menor similitud que contribuye a la formación de *CSP* (es decir, la capa en la que se encuentra el producto menos similar de los que constituyen *CSP*). Usando un ejemplo tremendamente simplificado, si hubiésemos obtenido tres capas de similitud L_1, L_2 y L_3 con dos productos en L_1, tres productos en L_2 y uno en L_3, y asumiendo que $k = 4$, la capa de menor similitud que contribuiría a la formación de *CSP* sería L_2.
- Paso 3: ejecuta un algoritmo auxiliar que, dado un conjunto inicial de productos, un conjunto de candidatos a añadir a ese conjunto inicial y un valor de *k*, consigue un conjunto *conj_recomendados* de productos que será el que definitivamente se presente al usuario. *conj_recomendados* contendrá todos los elementos del conjunto inicial más algunos del conjunto de candidatos a ser recomendados. La decisión de qué producto del conjunto de candidatos se añade en cada paso del algoritmo se basa en localizar el producto del conjunto de candidatos que tiene mayor diversidad relativa con los que ya se encuentran en *conj_recomendados* en ese momento. El conjunto inicial estará formado por todos los productos que se encuentran en capas de similitud mayor que la capa identificada como la de menor similitud que contribuye a la formación de *CSP*, siempre que el producto de mayor similitud con la consulta, P_{max}, no se encuentre en la capa identificada como de menor similitud que contribuye a la formación de *CSP*. Si, por el contrario, P_{max} se encuentra en la capa identificada como la de menor similitud que contribuye a la formación de *CSP*, entonces el conjunto

inicial es $\{P_{max}\}$. El conjunto de candidatos a añadir al conjunto inicial estará formado por todos los productos de la capa identificada como la de menor similitud que contribuye a la formación de *CSP* si P_{max} no se encuentra en la capa de menor similitud. Si por el contrario P_{max} sí se encuentra en la capa de menor similitud que contribuye a la formación de *CSP*, el conjunto de candidatos serán todos los demás productos de dicha capa. En el ejemplo simplificado usado en el paso anterior, donde $k = 4$, el conjunto inicial estaría formado por los dos elementos de la capa L_1. Por su parte, el conjunto de candidatos a añadir estará formado por los tres productos en L_2, de los cuales se elegirían dos de forma que sean los más diversos en relación con los de la capa L_1.

Esta aproximación también permite mejorar la diversidad preservando la similitud con la consulta del usuario. Sin embargo, la mejora en la diversidad sigue siendo menor que con el algoritmo de selección voraz acotada explicado anteriormente, ya que todos los productos de la capa de mayor similitud son siempre incluidos en el conjunto resultado sin ninguna mejora en la diversidad. La segunda alternativa que proponen en [52] está basada en la noción de "intervalos de similitud", que abarcan cada uno más de una capa de similitud de las usadas por DCR-1. La ventaja de esta aproximación es que se puede mejorar la diversidad de los resultados relajando las restricciones de similitud con la consulta. La similitud con la consulta se reduce pero a un nivel tolerable definido por el ancho de los intervalos de similitud.

2.2.4. Personalización

La personalización de las tecnologías promete al usuario una experiencia en línea más atractiva, una experiencia que se adapte a sus preferencias a largo plazo, así como a sus necesidades a corto plazo, tanto en términos de la información que se le presenta como en la forma en la que se presenta. En particular, las preferencias del usuario deben ser consideradas como una importante fuente de información con la que guiar el proceso de recomendación.

En teoría, los sistemas de recomendación personalizados deben ser capaces de responder eficazmente a los usuarios con menos información en la consulta. Es decir, si el sistema ha aprendido preferencias del usuario, éstas deben poder completar la información sin necesidad de que el sistema tenga que volver a pedírsela, dando lugar a sesiones de recomendación más cortas, en las que el usuario alcanzará un producto deseado más fácilmente.

La clave entonces en la construcción de un recomendador personalizado se basa en la capacidad de aprender y mantener un modelo de usuario a largo plazo (perfil de usuario) que almacene las preferencias en la recomendación.

El recomendador empleado por Amazon.com [50] utiliza algoritmos de recomendación personalizada para cada cliente. La tienda cambia radicalmente si quien la visita es una madre primeriza a la que se le mostrarán productos para bebés, o un joven interesado por los videojuegos y la tecnología. Esto es posible gracias al perfil de usuario que almacena con el historial de navegación, los productos adquiridos y las preferencias marcadas en el perfil.

Adaptive Place Advisor [84] captura las preferencias que un usuario puede tener en relación a los productos, los atributos y sus valores incluyendo la importancia relativa de un determinado atributo, el valor preferido para un atributo, y productos específicamente preferidos, todo ello codificado como distribuciones de frecuencia.

Existen diferentes alternativas a la hora de representar el perfil de usuario. Según [61] se pueden distinguir dos grandes grupos en los sistemas recomendadores:

- Un modelo basado en las preferencias de usuario. En el perfil se almacena una descripción de los productos junto con las preferencias del usuario. La alternativa más común en este tipo de representación es una función que para cada producto estime el interés del usuario por el mismo en base a las características que lo describen.
- Un modelo basado en las interacciones del usuario con el sistema. Esto debería incluir una historia de los productos que el usuario ha visto junto con otra información sobre la interacción del usuario (es decir,, si el usuario ha comprado el artículo o lo ha valorado).

Existen, en general, dos formas en las que se puede obtener información sobre los usuarios en los sistemas recomendadores: de manera explícita y de manera implícita. Muchos recomendadores utilizan las evaluaciones que los usuarios hacen sobre determinados productos recogidas después de preguntarles sobre la valoración que le dan a un determinado producto a partir de una escala [90]. *MovieLens* [55], por ejemplo, solicita al usuario una valoración después de que éste ha visto la película, expresada en forma de escala de 5 estrellas. El usuario puede así introducir de manera explícita sus preferencias. Esta manera de recoger información requiere de un cierto esfuerzo por parte del usuario, necesita dedicar un tiempo a rellenar las preferencias, y se ha comprobado que de esta manera los usuarios no siempre introducen sus preferencias, o no proporcionan información real [26].

Otro tipo de información son las preferencias que el usuario ha llevado al sistema de una manera implícita. Los usuarios no son conscientes de cómo el recomendador opera. El sistema monitoriza su comportamiento y automáticamente infiere las preferencias [26]. Esto puede hacerse analizando su historial de visita, o a partir de la información asociada a los productos que ha seleccionado [50].

2.2.5. Confianza en la recomendación

Proporcionar buenas explicaciones inspira confianza en el usuario en cuanto al sistema que utiliza, aumenta la satisfacción por los resultados obtenidos, hace rápido y sencillo a los usuarios encontrar aquello que quieren, y los persuaden para aceptar aquel producto que le fue recomendado [38, 85]. Los sistemas recomendadores también pueden proporcionar explicaciones al usuario, justificando así por qué los productos mostrados como resultado de la recomendación fueron recuperados.

Se definen siete posibles objetivos de explicación en un recomendador [85]:

- Transparencia. Una recomendación acompañada de una explicación puede clarificar al usuario cómo funciona el sistema.
- Análisis. Las explicaciones deben ser parte de un ciclo donde el usuario entiende lo que está pasando en el sistema y ejerce el control sobre el tipo de recomendaciones realizadas, mediante la corrección de supuestos erróneos.
- Confianza. Un usuario puede ser más indulgente, y tener más confianza en las recomendaciones, si entiende por qué le realizó una mala recomendación y así podrá evitar que ocurra de nuevo. Los usuarios también aprecian que un sistema sea "franco" y admita que no ha realizado una buena recomendación.
- Efectividad. Ayuda al usuario a tomar buenas decisiones. Una explicación eficaz ayuda al usuario a evaluar la calidad de los productos sugeridos de acuerdo a sus propias preferencias.
- Capacidad de persuasión. Ayuda a convencer a los usuarios para que elijan entre alguna de las propuestas.
- Eficiencia. Ayuda al usuario a tomar decisiones más rápidamente. Las explicaciones deben hacer más rápido para los usuarios el proceso de decidir cuál de los productos recomendados es mejor para él.
- Satisfacción. Incrementa la facilidad de uso o disfrute. Las explicaciones pueden aumentar la satisfacción del usuario con el sistema [83], aunque si las explicaciones son pobres probablemente disminuya el interés del usuario [75], o la aceptación del sistema [38].

Un ejemplo de recomendador que incorpora explicaciones es *LIBRA* (Learning Intelligent Book Recommending Agent) [6], un recomendador híbrido de libros. Este recomendador proporciona tres tipos de explicaciones: basada en palabras clave, basada en los vecinos y por último basada en la influencia de cada producto en la recomendación. Otro ejemplo que utiliza explicaciones por palabras clave es el recomendador de noticias creado por Billsus y Pazzani [7], donde las explicaciones muestran al usuario sus preferencias con respecto a noticias similares en interacciones pasadas.

2.3. Problemas asociados a los recomendadores

Durante las secciones anteriores hemos podido intuir algunos de los problemas o limitaciones que surgen en los sistemas recomendadores. Resumiremos a continuación cuáles son.

En cuanto a los recomendadores basados en contenido se nos presenta el problema de la sobreespecialización, es decir, obtener en una sesión de recomendación productos muy similares entre sí. A este problema pretenden hacer frente aquellas iniciativas en las que se refina la estrategia de selección (por ejemplo, combinando similitud y diversidad, como hemos señalado en la sección previa). Otra limitación de los recomendadores basados en contenido tiene su origen en la especificación de los atributos de los productos considerados, una tarea costosa que, en algunos casos, incluso requiere la participación de un experto en el dominio de aplicación, capaz de describir el dominio del sistema de forma precisa.

En los recomendadores basados en contenido que almacenan perfiles de usuario nos encontramos con el problema del perfil incompleto, un perfil que no tiene preferencias asociadas. Un recién llegado al sistema tendrá que valorar un número considerable de productos antes de que el recomendador basado en contenido pueda realmente entender cuáles son las preferencias del usuario. Por lo tanto, un nuevo usuario que tiene pocas valoraciones, no obtendrá buenas recomendaciones hasta que no haya utilizado el sistema durante un tiempo.

En cuanto a los sistemas que utilizan filtrado colaborativo comenzaremos hablando del llamado *arranque en frío*, cuyos efectos se ponen de manifiesto durante las primeras etapas de funcionamiento de los sistemas colaborativos. Hasta que no se alcanza un número suficientemente elevado de usuarios registrados, el recomendador no dispone de información suficiente para crear de una forma precisa el vecindario (del usuario activo o del producto objetivo), minando de esta forma la calidad de las sugerencias ofrecidas.

Dos problemas en relación con este mismo son los de la incorporación de nuevos usuarios al sistema y la incorporación de nuevos productos. El primero hace que sea dificultosa la formación del vecindario asociado a un usuario que acaba de llegar al sistema, cuyo perfil registrará, típicamente, un número muy reducido de preferencias. Y el segundo problema está asociado a que los enfoques colaborativos convencionales sólo sugieren productos incluidos en los perfiles de los usuarios del sistema. Como consecuencia de esto último, es posible identificar un tiempo de latencia desde que el sistema conoce nuevos productos hasta que éstos son recomendados a los usuarios activos (dado que deben ser clasificados por un número representativo de usuarios antes de ser incluidos en alguna recomendación). Este problema se conoce en la literatura como *long-tail*. Tal como apunta el trabajo descrito en [37], esta limitación es crítica en algunos dominios de aplicación (por ejemplo, a la hora de recomendar programas de la televisión digital), en los que no sólo hay un flujo constante de productos nuevos, sino

que, en algunos casos, éstos sólo están disponibles durante un tiempo limitado. En este escenario, es especialmente importante reducir los tiempos de latencia mencionados antes, para así poder ofrecer a los usuarios los productos más novedosos del momento sin retardos innecesarios.

Otro de los problemas es el de la dispersión. Este problema está relacionado con la noción de vecindario adoptada en los enfoques colaborativos tradicionales. Sus efectos se manifiestan a medida que aumenta el número de productos disponibles en el sistema recomendador. Ante tal diversidad, es menos probable que dos perfiles contengan exactamente las mismas preferencias y, por consiguiente, es más difícil encontrar vecinos tanto para el usuario activo como para el producto objetivo, fase crucial en los enfoques colaborativos basados en usuario y en ítem, respectivamente.

Después tenemos el llamado efecto *gray sheep*, especialmente perjudicial para aquellos usuarios que, al tener preferencias muy diferentes a las del resto de usuarios, no pueden recibir recomendaciones colaborativas (debido a la imposibilidad de crear su vecindario).

Otro problema asociado a las valoraciones de los productos es el de su fiabilidad: si cualquiera puede recomendar, los propietarios de determinados productos podrían generar recomendaciones positivas de sus productos y negativas de los de otros. De ahí, que se ponga esfuerzo en crear sistemas robustos frente a estos comportamientos no deseados por parte de usuarios.

Los problemas de escalabilidad también limitan los enfoques colaborativos. Es evidente que a medida que aumenta el número de productos y usuarios en el sistema, también se incrementa el coste computacional del proceso de cálculo del vecindario (del usuario activo o de los productos definidos en su perfil), etapa crucial en este tipo de enfoques. Este problema también afecta a los enfoques basados en contenido, siendo necesario reducir el coste de comparar la consulta con las descripciones de los productos.

Por último comentar el problema de la privacidad, es decir, usuarios que no quieren que se conozcan sus hábitos o preferencias. Una posible solución a este problema es permitir la participación anónima en el sistema o bajo un pseudónimo. Algunos sistemas ofrecen las valoraciones de los usuarios de manera anónima, es decir, por ejemplo a la hora de generar explicaciones que incrementen la confianza del usuario sólo se le muestra un identificador de usuario junto con valoraciones, pero ningún tipo de dato personal.

2.4. Conclusiones

Después de este repaso al mundo de los recomendadores podemos destacar algunas ideas. Como primera, que es un campo de investigación muy amplio y con muchas

posibilidades, no en vano los sistemas aquí presentados corresponden mayoritariamente a la última década.

También hemos visto que existen muchas y variadas alternativas de diseño:

- *Basado en contenido (o en casos)* ¿frente a? *Colaborativo,* ¿están enfrentados estos dos tipos de recomendadores? Ambas alternativas están ofreciendo buenos resultados en trabajos de investigación. Elegir una u otra puede venir condicionado por la información inicial de que se disponga y sobre la que se pueda trabajar. Como hemos visto están surgiendo muchas iniciativas de recomendadores híbridos, que suelen combinar las dos grandes familias de recomendadores obteniendo así mejores resultados. Sin duda éste es un campo novedoso y con muchas posibilidades.
- *Reactivo* frente a *Proactivo*, ¿quién es mejor que lleve la iniciativa en la recomendación, el sistema o el usuario? Con un recomendador proactivo liberamos al usuario de la tarea de formular una consulta, pero también nos exponemos a errar en la recomendación al no conocer exactamente sus intereses a corto plazo, cosa que sí ocurre en un recomendador reactivo en el que es el usuario quien guía la recomendación.
- La siguiente alternativa: *Single-shot* frente a *Conversacional.* Los recomendadores *single-shot* requieren que el usuario tenga suficiente conocimiento del dominio como para proporcionar una especificación clara, y así obtener una buena recomendación a la primera. Por el contrario los recomendadores conversacionales pueden convertirse en una alternativa mejor cuando el campo de exploración es muy amplio o cuando el usuario no tiene suficiente conocimiento del dominio como para saber exponer sus necesidades o tener claras sus preferencias sobre un determinado producto/servicio desde el comienzo de la recomendación. Hay trabajos que muestran que se obtienen buenos resultados realizando una buena combinación de los dos tipos de conversación existentes: *Navegación por pregunta* y *Navegación por propuesta*. Las propuestas son una manera de atraer al usuario hacia un determinado producto de una manera más visual, por así decirlo, pero si previo a esa muestra de productos hemos realizado una serie de preguntas clave que nos han permitido hacernos una idea de lo que el usuario quiere, la recomendación podría ser mucho más efectiva.
- *Uso de similitud pura frente a medidas de calidad más elaboradas,* ¿es conveniente tener en cuenta otros aspectos distintos aparte de la similitud con la consulta a la hora de elaborar las recomendaciones? Como hemos visto, existe un buen número de trabajos que han afrontado el problema de la sobreespecialización utilizando medidas de calidad que tengan en cuenta la diversidad. Pero, ¿es la única forma en la que la medida de calidad se puede

refinar? Conviene analizar el dominio en cuestión y ver qué otras medidas de valoración se pueden tener en cuenta para refinar la medida de calidad, sin perder de vista la similitud con los intereses mostrados por el usuario.

- La personalización de los recomendadores, ¿es decisiva a la hora de aceptar los productos recomendados? ¿Hasta qué punto se pueden obtener personalizaciones adecuadas mediante la captura implícita de intereses? ¿Qué porcentaje de los usuarios está decidido a dar valoraciones de manera explícita de lo que se les recomienda? El papel que juegan los modelos de usuario en cuestiones de personalización es decisivo. La personalización también puede verse como una forma de obtener medidas de calidad más elaboradas. Por ejemplo, en un sistema reactivo la consulta puede utilizarse para medir la validez de los productos a recomendar en base a los intereses del usuario a corto plazo, mientras que tener en cuenta el perfil del usuario puede ser empleado para refinar la medida de calidad teniendo en cuenta el conocimiento a medio-largo plazo que se tiene del mismo.
- Por último, la inclusión de explicaciones, ¿es decisiva a la hora de aceptar las recomendaciones? ¿Qué coste se está dispuesto a pagar a cambio? Generar explicaciones puede conllevar un alto coste, según lo detalladas y personalizadas que se deseen. Las posibilidades también dependen del conocimiento de que se disponga, y disponer de ello puede incrementar el proceso de autoría.

Por último, y no menos importante, hay que tener en cuenta que los sistemas de recomendación se han aplicado en la mayoría de las ocasiones al dominio del comercio electrónico. Pero sería interesante explorar la aplicación de estos conocimientos a otros dominios tanto o más interesantes, con las implicaciones que ello conlleve.

Capítulo 3

RECOMENDACIÓN DE OBJETOS DE APRENDIZAJE

Recientemente el uso de sistemas recomendadores se ha transferido del campo del comercio electrónico al ámbito académico. En este sentido, hay trabajos que plantean herramientas de recomendación de cursos y actividades de aprendizaje [26, 41, 58] y recomendadores que sugieren a los profesores modificaciones para mejorar la efectividad de los sistemas educativos web [27].

El uso de recomendadores en el ámbito académico tiene otra aplicación clara: proporcionar acceso personalizado a Objetos de Aprendizaje (LOs, del inglés *Learning Objects*) que se hallan en repositorios educativos. Dado el gran número de LOs que suelen contener estos repositorios se hace necesario dotar a los mismos de mecanismos de acceso que faciliten la localización de aquellos recursos que sean de interés para el estudiante y que se adapten al conocimiento individual, a los objetivos y/o a las preferencias del mismo.

La necesidad de contar con mecanismos sofisticados de acceso la hemos podido constatar en la tarea docente diaria. En los últimos tres años, y a través del Campus Virtual de la Universidad Complutense de Madrid, se ha puesto a disposición de los estudiantes de asignaturas de la materia de "Introducción a la programación" de diversas titulaciones un conjunto de recursos educativos. En concreto se trata de más de 400 ejemplos resueltos y ejercicios para resolver organizados por paquetes temáticos y niveles de dificultad [33]. Aunque los estudiantes valoran muy bien la ayuda que estos recursos les proporcionan, el 68% de los estudiantes echaba en falta facilidades más sofisticadas para el acceso a los mismos y el 41% coincidía en la conveniencia de que

fuese algún tipo de acceso guiado por el conocimiento del estudiante [34]. Por este motivo, dentro del Grupo de Aplicaciones de Inteligencia Artificial (GAIA) de la UCM se iniciaron trabajos en el campo de la recomendación de LOs que ayudaran a los estudiantes a seleccionar aquellos contenidos mejor adaptados a su estado de conocimiento. Fruto de estos esfuerzos se definió una aproximación híbrida de recomendación de LOs descrita en [31, 32]. En este trabajo proponemos cómo mejorar esa primera aproximación desarrollada dentro del grupo de investigación.

En la literatura se pueden encontrar otros esfuerzos realizados en la línea de desarrollar mecanismos de recomendación para repositorios de recursos educativos. Por ejemplo, hay bibliotecas digitales que aplican métodos de recuperación basados en contenido. Consideran los recursos educativos como documentos y los mecanismos de búsqueda aplican métodos de Recuperación de Información, como el modelo del espacio vectorial, para localizar los recursos que satisfacen la consulta del estudiante [48]. Otros repositorios de contenidos educativos incorporan filtrado colaborativo. Estos tienen en cuenta las valoraciones asociadas a los recursos asignadas por los estudiantes, independientemente del perfil de los mismos, como en el caso de Knowledge Sea [16], una plataforma de acceso a documentos electrónicos sobre el lenguaje de programación C.

Comenzaremos el capítulo describiendo la estrategia de recomendación de LOs que ha servido de punto de partida al presente trabajo y analizando los inconvenientes achacables a la misma (Sección 3.1). En la Sección 3.2 introduciremos las dos estrategias propuestas que ayudan a solventar dichos inconvenientes. El conocimiento necesario en ambas estrategias se describe en la Sección 3.3. El funcionamiento de las mismas se presenta en las Secciones 3.4 y 1.1. Terminaremos el capítulo con las conclusiones obtenidas (Sección 0).

3.1. Recomendación en repositorios de Objetos de Aprendizaje: una primera aproximación

El trabajo desarrollado en este Proyecto de Fin de Máster parte de una primera aproximación presentada en [31, 32] en la que se describía un enfoque novedoso para recomendar LOs. En dicho trabajo se planteaba una estrategia de recomendación híbrida en cascada [17]: un enfoque de recomendación reactivo, single-shot, basado en contenido, actuaba en primer lugar y sus decisiones eran refinadas a continuación por una estrategia colaborativa. La estrategia de recomendación empleada localizaba un conjunto relevante de LOs después de que el estudiante hubiera planteado una consulta al sistema. El resultado era una lista ordenada de LOs. La prioridad era para aquellos LOs más similares a la consulta del estudiante, adaptados a su nivel de conocimiento y que a la vez son los mejor valorados por otros estudiantes. Explicaremos a continuación

cómo funcionaba esta estrategia, la cual consta de dos etapas: recuperación y ordenación. En la parte izquierda de la Figura 9 podemos ver un esquema de funcionamiento de la estrategia. Como puede observarse, la estrategia estaba soportada por una indexación de LOs basada en una ontología del dominio. Cómo veremos, este esquema de indexación es crucial tanto en la etapa de recuperación como en la de ordenación.

La etapa de recuperación busca LOs que satisfagan, de una manera aproximada, los objetivos de aprendizaje del estudiante. El estudiante plantea una consulta al sistema utilizando conceptos del dominio. Esta consulta representa los objetivos de aprendizaje que quiere aprender en esa sesión de estudio. El *componente de recuperación* encuentra aquellos LOs indexados por los conceptos de la consulta. Si no hay LOs que satisfagan esta condición, la recuperación exacta anterior se sustituye por una recuperación aproximada que se realiza teniendo en cuenta un subconjunto de los mismos o similares conceptos a los planteados en la consulta del estudiante, para lo cual la taxonomía de conceptos del dominio es una fuente de conocimiento fundamental.

Una vez que los LOs han sido recuperados, el *componente de ordenación* los ordena de acuerdo a la relevancia asignada a cada LO. La etapa de ordenación combina una estrategia basada en compromiso (recuperar aquellos LOs de acuerdo a la consulta realizada) con una estrategia de filtrado colaborativo basada en usuario con el fin de generar una puntuación para cada LO. La puntuación de un LO *L* para un estudiante *S* se calcula como la suma de dos elementos:

(a) *Un elemento que representa el compromiso* y que combina*:*

   a. *La relevancia debida a los objetivos cumplidos por L*. Cuanto mayor es el número de conceptos de la consulta que *L* permite aprender, mayor es el valor de la relevancia. Cuanto más similares son los conceptos de la consulta y los que *L* permite aprender, mayor es el valor de la relevancia.

   b. *La relevancia debida al grado de adaptación de L al conocimiento actual de S*. El conocimiento de *S* está representado en su perfil dentro del 'repositorio de perfiles de estudiantes' (véase Figura 9). El objetivo es penalizar a *L* si incluye conceptos (diferentes de aquellos que satisfacen la consulta) que no se encuentran en el perfil de *S*.

(b) *Un elemento que representa la relevancia debida a la utilidad asignada a L por otros estudiantes con objetivos y perfiles similares a S*. El 'repositorio de preferencias de estudiantes' (véase Figura 9) almacena las valoraciones asignadas a los LOs utilizados por los estudiantes junto con los objetivos y el perfil del estudiante que los valoró en el momento de utilizarlos. El peso asignado a cada valoración se asigna de acuerdo a la similitud entre el perfil del estudiante que valoró el LO y el actual perfil de *S*. Cuanto más similares son los perfiles, mayor es la relevancia asignada a la valoración.

Esta estrategia híbrida alivia algunos de los problemas asociados a los recomendadores colaborativos (explicados en el capítulo anterior). Si un LO tiene pocas valoraciones o un estudiante es nuevo en el repositorio, la estrategia devuelve un conjunto de LO relevantes de acuerdo a la consulta realizada por el estudiante. Por otra parte, el filtrado colaborativo basado en usuario que actúa en segundo lugar tras el basado en contenido no necesita explorar el repositorio completo de preferencias de usuario para buscar estudiantes similares. Los perfiles escogidos son aquellos que ya han valorado un LO candidato (aquel seleccionado en la etapa de recuperación). Esto reduce los problemas de *escalabilidad* y de *dispersión* asociados a los recomendadores colaborativos basados en usuario.

Aunque a esta estrategia de recomendación se le pueden atribuir las ventajas enunciadas, también es posible achacarle los siguientes inconvenientes:

- Primero, la débil personalización proporcionada por la recomendación basada en contenido descrita anteriormente. Puede decirse que dicha estrategia implementa un tipo de personalización que se limita, en gran medida, a las necesidades manifestadas por el estudiante en la sesión de recuperación: se toma en cuenta los objetivos a corto plazo propuestos en forma de consulta. De esta manera, para dos estudiantes que planteen la misma consulta en una sesión se obtendrán las mismas recomendaciones en el proceso de recuperación, incluso si sus objetivos de aprendizaje a largo plazo y su destreza en el dominio difieren en gran parte. El conocimiento que del dominio tiene previamente el estudiante sólo se tiene en cuenta en el proceso de ordenación, a la hora de calcular la relevancia que representa el compromiso.
- Otra de las desventajas radica en la naturaleza reactiva de la estrategia propuesta. El estudiante debe formular una consulta en términos de los conceptos en los que está interesado. El planteamiento de la consulta puede imponer una empinada curva de uso para aquellos estudiantes con escasos conocimientos. Esto además puede ser un obstáculo para estudiantes con una actitud pasiva.
- La última desventaja es el problema asociado a los recomendadores basados en contenido que siguen un enfoque de similitud pura: la sobreespecialización de la que ya se ha hablado en el capítulo anterior (Sección 2.3). Como ya se indicó, el uso de enfoques basados en una similitud pura da lugar a la sobreespecialización, o lo que es lo mismo, a recomendaciones que carecen de diversidad: sólo aquellos LOs que estén altamente relacionados con la consulta del estudiante son candidatos a ser recomendados. En este sentido, las recomendaciones mejor clasificadas apenas difieren unas de otras. Consecuentemente, cuando a un estudiante no le gusta la primera recomendación probablemente tampoco estará satisfecho con las otras. Esto supone un problema en nuestro contexto, donde es necesario restringir el número de recursos que se mostrarán al estudiante para así no sobrecargarle con una alta carga de trabajo en cada sesión de estudio. El

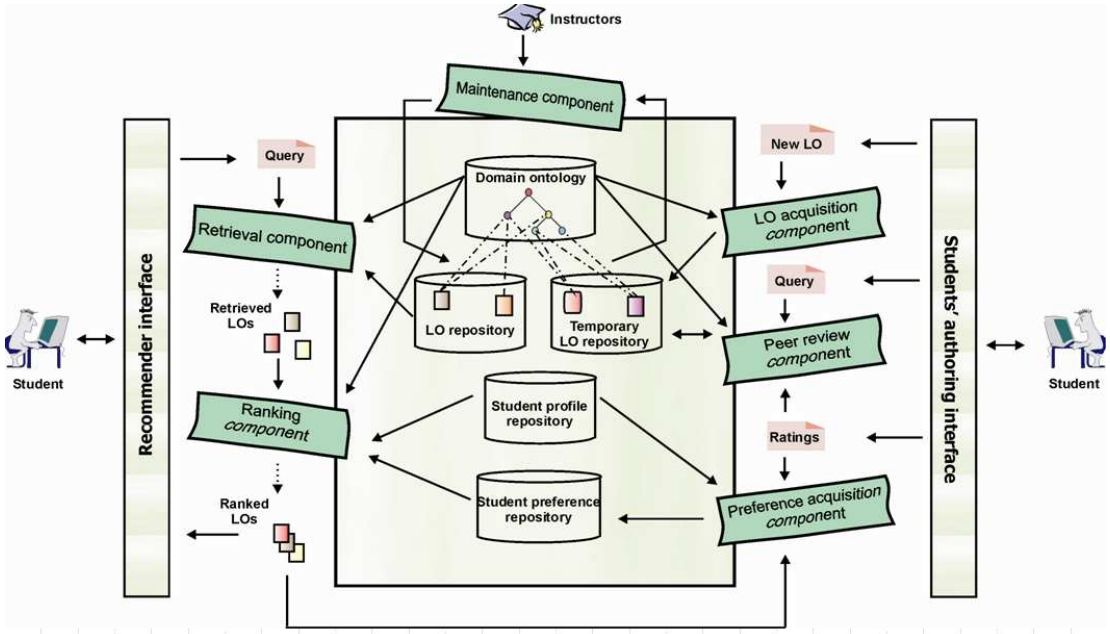


Figura 9. Un boceto de la estrategia presentada en [31, 32] [2].

indeseado efecto del problema de la sobreespecialización se hace más grave para aquellos estudiantes con poco conocimiento del dominio porque propondrán consultas menos exactas.

Estas desventajas nos han llevado a plantear nuevas estrategias de recomendación basadas en contenido que pudieran sustituir a la existente y que alivien estas dificultades. Las nuevas estrategias basadas en contenido, de validez por sí mismas, podrían seguir siendo encajadas en una estrategia híbrida como la descrita aquí.

3.2. Nuevas estrategias de recomendación basadas en contenido

Una vez vistos los posibles inconvenientes de la anterior propuesta de recomendación de LOs, nuestro trabajo ha ido dirigido a plantear dos aproximaciones de recomendación que aliviaran dichos inconvenientes. Estas dos aproximaciones son las siguientes:

- La primera aproximación pretende mejorar la capacidad de adaptación al usuario explorando un modelo de personalización fuerte [66, 67]. Esta aproximación se concibe como un recomendador basado en contenido que mantiene un modelo de

[2] En esta figura también podemos ver la parte de autoría incluida en la estrategia. Dicha parte no tiene relevancia para el trabajo actual pero se puede encontrar descrita en [31,32].

operación reactivo, esto es, bajo demanda del estudiante: el estudiante deberá formular una consulta indicando qué es lo que quiere aprender en su sesión de estudio. A la hora de recuperar LOs veremos que se tiene en cuenta no sólo la consulta formulada por el estudiante sino también el conocimiento que del dominio tiene el mismo, incorporando así una personalización mayor desde la primera etapa del proceso recomendador. Esta estrategia asigna la prioridad a aquellos LOs que son más similares a la consulta del estudiante (objetivos a corto plazo) y, al mismo tiempo, tienen una utilidad pedagógica mayor de acuerdo a su perfil (objetivos a largo plazo). Esta aproximación incluye la definición de una métrica que combina de manera flexible la similitud con la consulta y la utilidad pedagógica del LO. Como veremos, en nuestro caso, hemos decidido aplicar una estrategia pedagógica que asigna altas utilidades pedagógicas a aquellos LOs que permiten remediar carencias de conocimiento reflejadas en el perfil del estudiante. Obviamente se pueden seguir otras estrategias que midan de diferente forma la utilidad pedagógica (por ejemplo, un LO podría considerarse de gran utilidad pedagógica si cubre muchos conceptos ya revisados por el estudiante) y que podrían encajarse en la estrategia definida aquí sin más que cambiar la métrica que permite valorar la utilidad pedagógica a largo plazo.

- La segunda aproximación intenta aliviar las desventajas mostradas por la sobreespecialización y por la naturaleza reactiva del recomendador inicial [65]. Para aliviar la primera desventaja, esta estrategia promueve la diversidad del conjunto recuperado sin comprometer, de manera significativa, el interés que los LOs recomendados puedan tener para el estudiante. La diversidad se consigue dividiendo el repositorio de LOs en grupos *de interés* y seleccionando LOs representantes de cada uno de los grupos para formar el conjunto recomendado. Para la segunda desventaja, esta aproximación seguirá una estrategia proactiva que propondrá a los estudiantes LOs que puedan ser de su interés en una sesión de estudio sin la necesidad de que ellos hagan una consulta explícita; así, estudiantes poco motivados o que no tienen conocimiento suficiente para proponer una consulta recibirán una recomendación de LOs acorde a sus necesidades de aprendizaje. Esta aproximación se ha diseñado teniendo en mente un esquema de navegación por propuesta [76]. De esta manera, conseguimos captar al estudiante en un sencillo proceso conversacional que evita la realización de preguntas directas, pero que presenta recomendaciones alternativas y solicita al estudiante realimentación basada en preferencias: el estudiante mostrará predilección por una alternativa frente a las otras, haciéndole así partícipe del proceso de recomendación.

Estas dos aproximaciones seguirían encajando perfectamente en el recomendador híbrido propuesto en el trabajo previo. Al igual que ocurría en [31, 32], la recomendación basada en contenido dirigiría la búsqueda de los posibles LOs

interesantes y la recomendación CF ayudaría a seleccionar los LOs finalmente propuestos al estudiante.

3.3. Las fuentes de conocimiento

El uso de recomendadores en el ámbito académico impone unos requisitos específicos y es posible sacar partido de diversos tipos de conocimiento en el proceso de recomendación [23, 24]. Por ejemplo, los recomendadores pueden sacar provecho del estado cognitivo del estudiante, el cual cambia a lo largo del tiempo. Esto permitiría incrementar el nivel de personalización a largo plazo.

Por otro lado, los buenos itinerarios y estrategias de aprendizaje también puede proporcionar buenas guías para el recomendador. Por ejemplo, el recomendador puede sacar provecho de una regla pedagógica simple como 'ir de tareas fáciles a difíciles' o 'reducir de manera gradual la cantidad de orientación'. Los itinerarios de aprendizaje pueden representar rutas y secuencias diseñadas por los profesores a partir de experiencias positivas en el aula, o se pueden corresponder con el comportamiento de estudiantes avanzados.

En esta sección describimos de dónde proviene el conocimiento al que sacarán partido las aproximaciones de recomendación propuestas: la ontología del dominio (Subsección 3.3.1), los Objetos de aprendizaje y sus metadatos (Subsección 3.3.2) y por último el perfil de estudiante (Subsección 3.3.3).

3.3.1. La ontología del dominio

Ontología en Filosofía significa "*una explicación sistemática de lo que existe"*. Es una rama de la metafísica que trata de describir o proponer las categorías y relaciones básicas del ser o la existencia para definir las entidades y decir de qué tipo son**.**

En Inteligencia Artificial, y en particular en Ingeniería del Conocimiento, la palabra *ontología* se utiliza para denotar una base de conocimiento reutilizable, es decir, una representación explícita, expresada en un lenguaje formal, del conocimiento acerca de un dominio:

> *"Ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base" [82]*
>
> *"Ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base". [5]*

Al contrario de lo que ocurre en Filosofía, en los sistemas basados en conocimiento lo que existe es exactamente lo que se puede representar y formalizar. Una ontología, en este sentido, hace referencia a un intento de formular un esquema conceptual dentro de

un dominio dado, de manera que pueda ser reutilizable por cualquier otro sistema que trabaje sobre el mismo dominio. Una de las definiciones más aceptadas de lo que es una ontología, y que hace referencia al concepto de conceptualización, es la dada por Gruber en 1993:

"a formal, explicit specification of a shared conceptualization" [36]

El termino conceptualización es uno de los más usados como sinónimo de ontología y se refiere a un modelo abstracto de algún fenómeno del mundo del que se identifican los conceptos que son relevantes.

Una ontología es mucho más que una taxonomía de conceptos. Las ontologías se componen también de instancias, relaciones que representan un tipo de interacción entre conceptos del dominio, funciones que son consideradas como un tipo especial de relación, y axiomas que son proposiciones siempre verdaderas y que se expresan en un lenguaje lógico.

Debido al esquema general que proporcionan las ontologías, que nos permite incluir conocimiento sobre la similitud entre los conceptos que representan los temas del dominio, proponemos utilizar una ontología para indexar los LOs contenidos en el repositorio. El conocimiento sobre la similitud entre los conceptos y sobre las relaciones que existen entre ellos veremos que es crucial en las estrategias de recomendación. Además las ontologías proporcionan un lenguaje común a la hora de compartir información.

Existen otros trabajos que también hacen un uso satisfactorio de ontologías en el ámbito académico. Por ejemplo en [49] utilizan una ontología para relacionar LOs con los conceptos del dominio y así a través de una búsqueda semántica mejorar la navegación por el material disponible en el repositorio. En [2] los recursos del repositorio están indexados gracias a una ontología de conceptos de aprendizaje. En [30] emplean una ontología de conceptos para representar el conocimiento del dominio. Esta ontología de conceptos es utilizada por el modelo de usuario para seleccionar qué conceptos deben ser mostrados a un determinado estudiante. También encontramos en [28] una ontología para LOs de un repositorio de contenidos educativos.

Para poder utilizar las características de las ontologías es necesario identificar todos los elementos que componen el dominio junto con todo aquello que deba tenerse en cuenta en el proceso de enseñanza y aprendizaje.

En primer lugar, es necesario hacer una representación de los objetivos pedagógicos que debe conseguir el estudiante. Esta representación se lleva a cabo mediante conceptos que el estudiante debe ser capaz de dominar después del proceso de aprendizaje. Así, es necesario realizar una completa taxonomía de los conceptos del dominio, estructurando estos conceptos de manera jerárquica. Los conceptos y subconceptos se organizan en una taxonomía utilizando la relación *is_a* (véase Figura 10).

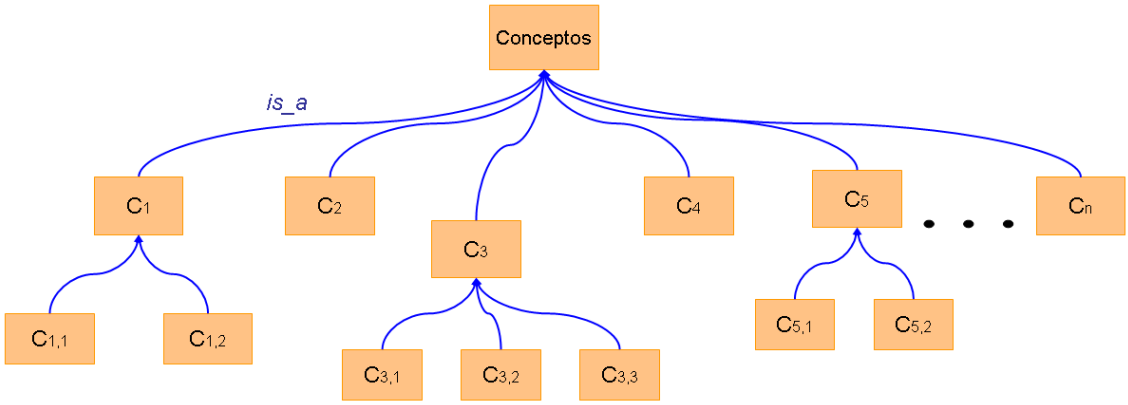


Figura 10. Ejemplo de una taxonomía de conceptos.

Un papel muy importante en cualquier proceso de entrenamiento y enseñanza es el itinerario (o itinerarios) de aprendizaje que elijamos para llevarlo a cabo. De ahí que otra buena fuente de información en un proceso de recomendación sean esos itinerarios de aprendizaje, que pueden representarse en la ontología a través de otras relaciones entre conceptos. Estos itinerarios podrán servir de guía al recomendador en el proceso de recuperación. El itinerario de aprendizaje debe ser propuesto por el responsable del aprendizaje de los estudiantes y puede incluir aquellas reglas pedagógicas que considere oportunas.

En esta aproximación la regla pedagógica está representada a través de la llamada relación de precedencia entre contenidos. Un estudiante debe cumplir un secuenciamiento en el orden de aprendizaje: antes de poder aprender contenidos nuevos debe haber alcanzado un cierto nivel de competencia en los contenidos previos. El secuenciamiento se lleva a cabo gracias al uso de diferentes *Lecciones*. Cada una de estas lecciones engloba los conceptos que deben estudiarse en un determinado momento. Esto se representa mediante la propiedad *has_contents*. El itinerario de aprendizaje definido en nuestro sistema se encuentra reflejado en la relación de precedencia entre lecciones, que se lleva a cabo mediante la relación simétrica *has_previous* y *has_next.* El criterio sobre el número de lecciones que deben definir el itinerario de aprendizaje así como los conceptos que forman parte de cada una de ellas depende de los criterios aplicados por el responsable del aprendizaje de los estudiantes en cada dominio concreto.

En la Figura 11 podemos ver un ejemplo de definición de un itinerario de aprendizaje. Observamos que se han definido 5 lecciones, junto con los conceptos que

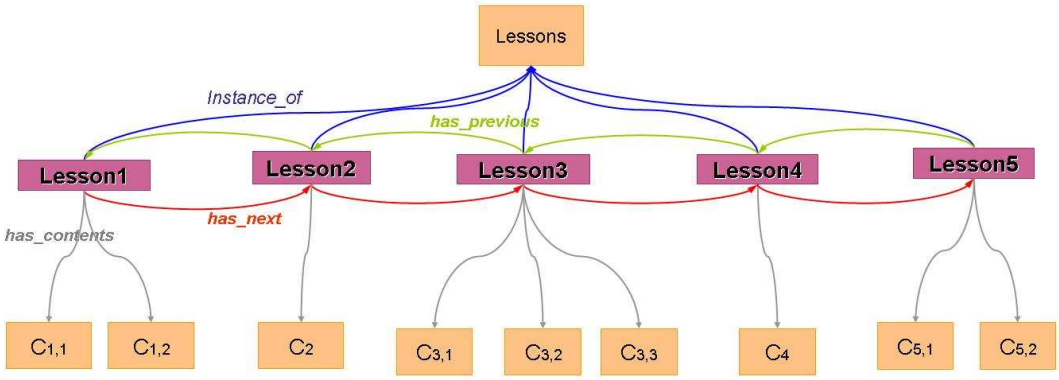


Figura 11. Vista parcial de las lecciones junto con los contenidos que tiene cada una de ellas y la relación de precedencia definida.

deben aprenderse durante su estudio (*has_contents)*. En esta figura también vemos la relación de precedencia establecida entre lecciones (relación *has_previous* y *has_next*), que define el itinerario de aprendizaje.

3.3.2. Los objetos de aprendizaje

Existen diferentes definiciones de lo que se entiende por LO. A continuación enumeramos algunas de las definiciones más populares en la literatura:

- Un LO se define como cualquier entidad, digital o no digital, que puede ser utilizada para el aprendizaje, la educación o para practicar [42].
- Un LO es cualquier recurso digital que puede ser reutilizado como soporte al aprendizaje. Esta definición incluye cualquier recurso que puede ser entregado a través de la red mediante demanda, ya sean grandes o pequeños. Ejemplos de pequeños recursos digitales reutilizables pueden ser imágenes o fotografías, canales de video, o fragmentos de audio, pequeños fragmentos de texto, animaciones, y pequeñas aplicaciones web, como una calculadora hecha en Java. Ejemplos de recursos digitales reutilizables grandes pueden ser, páginas web enteras que combinen texto, imágenes u otras aplicaciones multimedia que presenten una experiencia completa, como un caso de enseñanza completo [89].
- Un LO se define como la más pequeña experiencia estructural independiente que contiene un objetivo, una actividad de aprendizaje y una evaluación. Por objetivo se entiende un componente estructural de un LO que es un estado que describe el resultado previsto basado en ciertos criterios establecidos, para una actividad de aprendizaje. Actividad de Aprendizaje es un componente estructural de un LO

que enseña un objetivo. Evaluación es un elemento estructural de un LO que determina si un objetivo ha sido cumplido [47].

Desde nuestro punto de vista entendemos un LO como un recurso (generalmente digital) que puede ser usado y reutilizado como soporte a la enseñanza y al aprendizaje y que además es una experiencia de aprendizaje que contiene un/os objetivo/s y una actividad de aprendizaje. Podíamos decir que entendemos los LOs de una manera muy cercana a como lo hacen [89] y [47].

Los recomendadores cuentan con una colección de productos que pueden ser recomendados a los usuarios del sistema. En nuestro contexto estos productos son los LOs contenidos en un repositorio educativo.

Nuestros LOs han sido desarrollados de acuerdo al estándar Learning Object Metadata (LOM) [42], un modelo de datos utilizado para describir LOs y contenidos digitales utilizados como soporte a la enseñanza y al aprendizaje. LOM permite etiquetar LOs de acuerdo a conjunto predefinido de categorías y asignar valores a cada una de ellas.

LOM está compuesto por una jerarquía de elementos. En el primer nivel existen nueve categorías, cada una de ellas contiene subelementos; estos subelementos pueden ser elementos simples, como datos fijos, o pueden ser un conjunto de elementos, que contengan más subelementos.

Hemos decidido utilizar en el más alto nivel las siguientes categorías de LOM: *General, Ciclo de vida, Meta-metadatos, Técnica, Educativa, Relación y Clasificación.* Hemos descartado otras categorías, como *Derechos* y *Comentarios.*

La categoría *General* veremos que juega un importante papel en la etapa de recuperación. Esta categoría contiene palabras clave que describen qué conceptos de la ontología están cubiertos por el LO concreto. Estas palabras clave servirán de punto de enlace entre los LOs contenidos en el repositorio y la jerarquía de conceptos representada en la ontología.

La categoría *Clasificación* la hemos usado para describir el LO en relación a un sistema de clasificación. Proponemos que cada LO sea clasificado de acuerdo a la versión revisada de la Taxonomía de Bloom [46] (véase Figura 12). La Taxonomía de Bloom puede entenderse como "los objetivos del proceso de aprendizaje". Esto quiere decir que después de realizar un proceso de aprendizaje, el estudiante debe haber adquirido nuevas habilidades y conocimientos que son los reflejados en dicha taxonomía. La taxonomía de Bloom asume que el aprendizaje a niveles superiores depende de la adquisición del conocimiento y habilidades de ciertos niveles inferiores. Los objetivos cognitivos giran en torno del conocimiento y la comprensión de cualquier tema dado. Estos objetivos (en la versión revisada de la taxonomía) son: *recordar, comprender, aplicar, analizar, evaluar, crear*. Cada LO pertenece a una de las categorías de esta taxonomía. Como veremos, esta clasificación de los LOs permite al

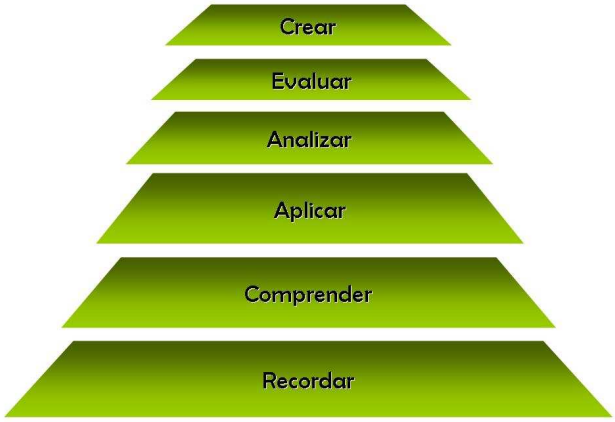


Figura 12. Versión revisada de la taxonomía de Bloom.

recomendador una mejor adaptación al estado cognitivo del estudiante, ya que habrá contenidos que sea mejor recomendarlos en las primeras etapas del aprendizaje y otros que se adecuen más a las siguientes etapas.

La categoría *Ciclo de vida* identifica al autor y el estado actual en el proceso de desarrollo del LO. La categoría *Educativa* nos permite identificar el tipo de recurso educativo al que pertenece el LO. Entre los distintos tipos de recursos educativos podemos encontrar: ejercicios, ejemplos resueltos, cuestiones de autoevaluación, presentaciones, etc. La categoría *Técnica* agrupa los requisitos técnicos y las características técnicas del LO. Por último, la categoría *Relación* nos permite identificar si un LO es una versión de otro contenido en el repositorio.

Es necesario tener en la ontología una representación de los LOs contenidos en el repositorio junto con los conceptos que cubre cada uno de ellos. Esta representación nos proporcionará conocimiento sobre cada LO que posteriormente será utilizado por las estrategias de recomendación, para decidir la idoneidad de cada LO. Este conocimiento se refleja en la ontología mediante una clase denominada *LearningObject.* Para cada LO contenido en el repositorio existirá una instancia de esta clase que lo represente. La relación *covers* representa los conceptos que el LO concreto cubre. En la Figura 13 podemos ver un ejemplo de algunos LOs que forman parte del repositorio junto con los conceptos cubiertos por cada uno de ellos.

3.3.3. El perfil de estudiante

La tarea de modelado de usuario constituye un campo de investigación muy extenso donde se trata de definir modelos de comportamiento y conocimiento de los usuarios en diferentes tipos de dominios, la mayoría fuera de la enseñanza.

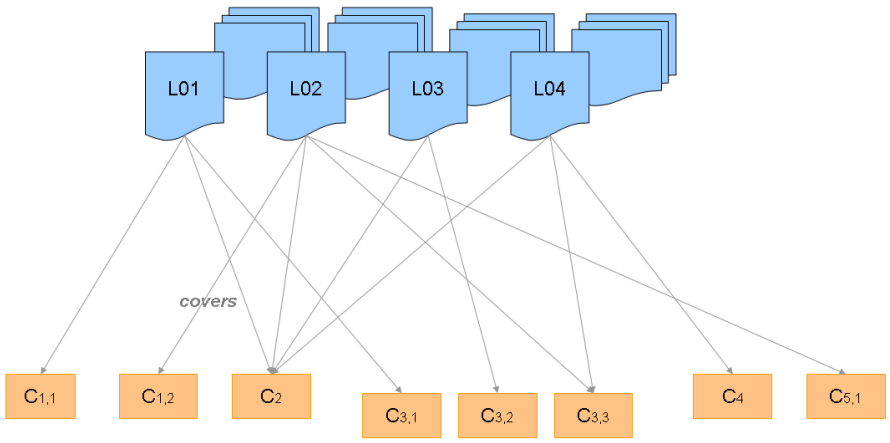


Figura 13. Algunos LOs junto con los conceptos de la ontología que cubren.

Sin embargo, el modelado de estudiantes es una técnica que se aplica dentro de sistemas de enseñanza inteligente, en este caso para representar los comportamientos y estados de los estudiantes, relacionado con los intereses pedagógicos del sistema.

La construcción de un modelo de usuario involucra definir: el "qué", o los objetivos, esquemas, actitudes, capacidades del estudiante; el "cómo" el modelo es adquirido y mantenido; y el "por qué", incluyendo si se obtiene información por parte del estudiante, si es para dar asistencia a los alumnos, o para proporcionar realimentación al estudiante, o para interpretar el comportamiento del estudiante, etc [81].

El estilo de aprendizaje usado por el estudiante, su nivel de conocimiento, sus intereses con respecto al proceso de aprendizaje, su comportamiento de navegación y uso de los recursos, etc. son diferentes elementos que se han empleado a la hora de definir modelos de estudiante en entornos pedagógicos, ya sea considerados de forma individual o combinando varios de ellos [40].

En los sistemas recomendadores, como en otros sistemas, la capacidad de personalización está relacionada con la capacidad de almacenar los perfiles de cada usuario y con la información que contengan los mismos.

En los perfiles de usuario empleados habitualmente por sistemas de recomendación se puede almacenar diferentes tipos de información, que según [61] se resumen principalmente en dos:

- Un modelo con las preferencias del usuario, es decir, la descripción de los tipos de productos que interesan al usuario.
- Un modelo que almacene la historia de las interacciones del usuario con el recomendador. Esto incluye almacenar los productos visitados por el usuario junto con otra información adicional sobre su interacción (por ejemplo, si el usuario ha valorado el producto o si lo ha comprado).

Evidentemente, lo anterior corresponde a sistemas recomendadores en el ámbito del comercio electrónico, que ha sido, cómo ya dijimos, básicamente la principal área de aplicación de este tipo de sistemas.

En nuestro caso, y de cara a conseguir recomendaciones adaptadas a nuestro tipo de usuario (los estudiantes), el perfil almacenará los objetivos involucrados en el proceso de aprendizaje junto con el nivel de competencia o maestría alcanzado en cada uno de ellos, y la información sobre la historia de navegación del estudiante –los LOs que ha visitado.

Para saber si un determinado estudiante *S* ha explorado o no un determinado concepto *C*, en el perfil estarán reflejados todos los conceptos (objetivos) junto con el nivel de competencia[3] alcanzado en el aprendizaje de los mismos (esto es, el nivel de competencia de un estudiante *S* en un concepto *C* representa el nivel de conocimiento alcanzado en dicho concepto). Los conceptos no explorados no tienen nivel de competencia asociado. En determinadas ocasiones, las estrategias de recomendación deberán conocer cuál es el nivel de competencia alcanzado en una determinada lección. El nivel de competencia de una lección se puede calcular de diversas formas a partir de los niveles de competencia de los conceptos que engloba, por ejemplo, como la media aritmética de los niveles de competencia de los conceptos que forman parte de ella. El perfil del estudiante contendrá, como se ha dicho, los conceptos y sus niveles de competencia, pero no las lecciones, cuyo nivel de competencia se calculará como se ha indicado.

El nivel de competencia asociado a cada concepto permitirá al recomendador seleccionar aquellos LOs que se adapten mejor al nivel de conocimientos del estudiante. En un determinado momento el recomendador puede considerar que el estudiante debe repasar ciertos conceptos debido a que el nivel de competencia alcanzado en los mismos es bajo, o, por el contrario, que está preparado para aprender conceptos nuevos porque el nivel de competencia alcanzado en los que lo preceden, según los itinerarios establecidos, así se lo permite. Para que el recomendador pueda llevar a cabo estas decisiones es necesario definir un umbral de progreso para el nivel de competencia. Si

[3] La valoración de los niveles de competencia se puede realizar mediante algún examen o cuestionario de forma totalmente externa al proceso de recomendación, pudiéndole asignar valores en el rango deseado.

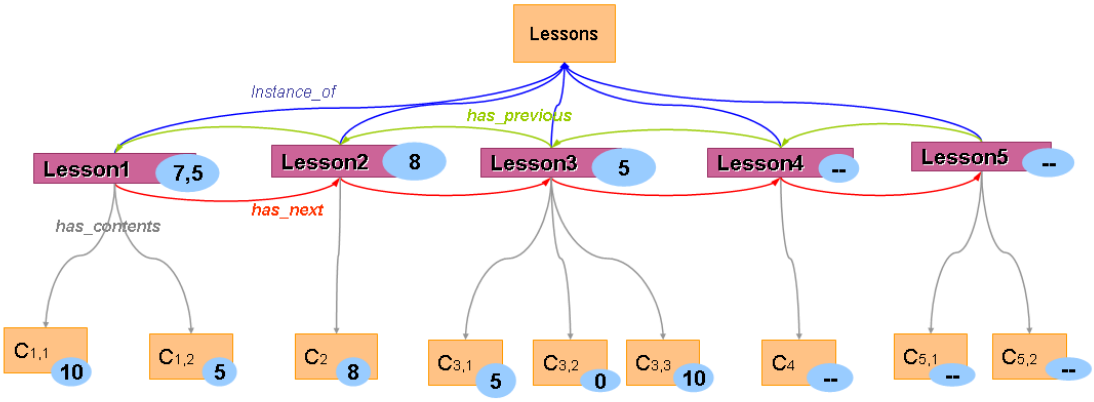


Figura 14. Representación gráfica parcial del perfil de un estudiante (los conceptos C_4, $C_{5,1}$ y $C_{5,2}$ aún no han sido explorados). Se visualizan también los valores de competencia de las lecciones que componen el itinerario de aprendizaje, calculados a partir del contenido del perfil.

el nivel de competencia supera dicho umbral indicará que los objetivos de un concepto han sido cumplidos.

En la Figura 14 tenemos representado esquemática y parcialmente el perfil del estudiante (se ha excluido la historia de navegación). Cada concepto tiene asociado un valor numérico que representa el nivel de competencia alcanzado en el mismo. Los conceptos no explorados no tienen nivel de competencia asociado. También está reflejado en la figura el nivel de competencia de las lecciones, computado a partir de los valores de competencia de los conceptos que almacena el perfil del estudiante.

3.4. Estrategia de recomendación reactiva combinando objetivos a corto y largo plazo

Esta primera estrategia de recomendación [66, 67] pretende extender y mejorar la estrategia basada en contenido que se describía en [31, 32] superando la primera debilidad achacable que se ha mencionado en la Sección 3.1: la existencia de una personalización *débil*. Para ello se introduce una personalización *fuerte*, esto es, se incorpora la consecución de objetivos a largo plazo a la consecución de objetivos a corto plazo. Como ya indicamos, la consecución de objetivos a corto plazo implica satisfacer, de forma más o menos aproximada, la consulta formulada por el estudiante.

La estrategia basada en contenido que proponemos mantiene un comportamiento reactivo: el estudiante proporciona una consulta explícita al sistema y el recomendador

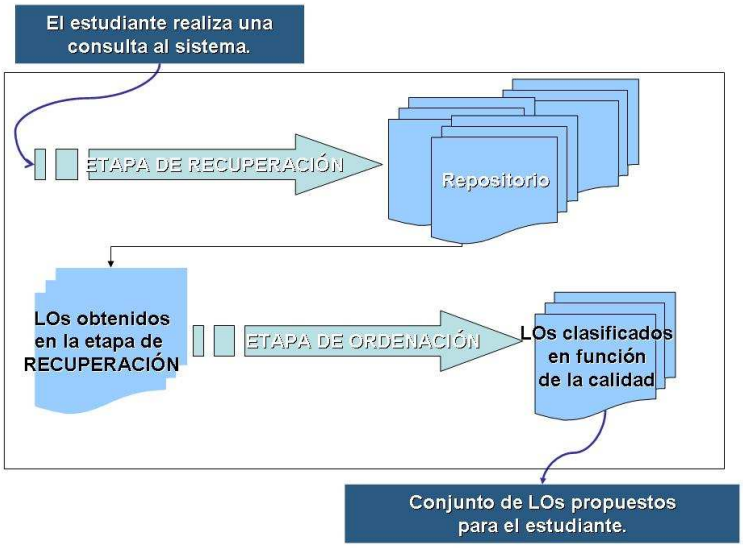


Figura 15. Diagrama de funcionamiento del recomendador reactivo.

responde con una recomendación. La consulta que realiza el estudiante está basada en los conceptos existentes en la ontología del dominio por lo que es importante que el estudiante conozca cuáles son estos conceptos para poder expresar sus necesidades en forma de consulta. Esta consulta representa los objetivos de la sesión de estudio para el estudiante: aquellos conceptos que quiere aprender en la sesión.

La respuesta de la recomendación se obtiene mediante un proceso en dos pasos: recuperación y ordenación (Figura 15). El estudiante realiza una consulta, a partir de la cual se recuperan del repositorio LOs que la satisfagan de manera aproximada, para después ordenar el conjunto recuperado en función de la calidad asignada a cada LO.

En la etapa de recuperación se hace una búsqueda en el repositorio de LOs guiada por los objetivos para la sesión de estudio actual junto con el perfil del estudiante. Este último ayudará a recuperar aquellos LOs que mejor se adapten a los conocimientos actuales del estudiante. Como resultado de esta etapa se obtiene un conjunto de LOs candidatos a mostrar al estudiante.

La siguiente fase, la etapa de ordenación, asigna una puntuación a cada LO que representa lo que llamaremos la calidad del LO. Esta puntuación permite ordenar los LOs recuperados en la etapa anterior para así mostrar los que mejor satisfacen las necesidades estudiante.

En las siguientes subsecciones se explican en detalle estas dos etapas.

3.4.1. Etapa de recuperación

La etapa de recuperación busca en el repositorio un conjunto inicial de LOs que satisfagan de una manera aproximada la consulta del estudiante. El proceso de recuperación primero intenta encontrar LOs indexados exactamente por los conceptos de la consulta, es decir, aquellos LOs que en la relación *covers* contengan los conceptos de la consulta. Si no hay LOs que satisfagan esta condición, o si estamos interesados en una localización más flexible, los LOs indexados por un subconjunto de los conceptos (iguales o similares) especificados en la consulta del estudiante serán recuperados. Para ello se utilizará la taxonomía de conceptos incluida en la ontología. La recuperación aproximada contempla la posibilidad de recuperar LOs que cubran conceptos hermanos de aquellos que están representados en la consulta o con un grado menor de parentesco. Teniendo la ontología representada por la Figura 10 una recuperación aproximada que implicase la subida de un nivel por la taxonomía de conceptos contemplaría la localización de LOs que cubran también los conceptos $C_{3,1}$ y/o $C_{3,2}$ si el estudiante solicitó en la consulta elementos que contuvieran $C_{3,3}$.

Este conjunto inicial de LOs es filtrado y sólo aquellos LOs que cubren conceptos de la ontología "listos para ser explorados" por el estudiante serán finalmente considerados en la etapa de ordenación. Este proceso de filtrado permite aumentar el nivel de personalización desde la primera etapa del proceso de recomendación, en la línea de mejoras que se pretendían realizar sobre la estrategia basada en contenido presentada en [31, 32].

En el proceso de recuperación tendremos, pues, que conocer cuándo un concepto perteneciente a una lección está "listo para ser explorado" por un estudiante dado. Este dato nos permitirá descartar aquellos LOs que no sean adecuados para el estudiante. Decimos que un concepto está "listo para ser explorado" por un estudiante si de acuerdo a su perfil y al itinerario de aprendizaje definido, cumple cualquiera de las siguientes condiciones.

- Es un concepto ya explorado por el estudiante, así que aparecerá en el perfil con el correspondiente nivel de competencia.
- Es un concepto que pertenece a una lección cuyos conceptos aún no han sido explorados pero está lista para ser descubierta: si l_1 precede a una lección l_2 en el itinerario de aprendizaje, el estudiante puede descubrir un concepto de l_2 si el nivel de competencia del estudiante para l_1 supera un "umbral de progreso". Si varias lecciones $l_1, l_2, \ldots, l_k$ preceden directamente a la lección l_x, un concepto de l_x puede ser descubierto si la media aritmética de los niveles de competencia de todas las predecesoras directas superan (o igualan) el "umbral de progreso".

De acuerdo a los conceptos de la taxonomía presentada en la Figura 10, al perfil del estudiante visto en la Figura 14 y al itinerario de aprendizaje representado en la Figura 11, los conceptos $C_{1,1}$, $C_{1,2}$, C_2, $C_{3,1}$, $C_{3,2}$ y $C_{3,3}$, así como los que están por encima de

ellos en la taxonomía, son conceptos "listos para ser explorados", ya que figuran en el perfil del estudiante con un nivel de competencia, es decir, estos conceptos ya han sido estudiados previamente por el estudiante actual. El concepto C_4 estará "listo para ser explorado" si el umbral de progreso definido así lo permite. C_4 pertenece a *Lesson4*, y según el itinerario de aprendizaje esta lección debe estudiarse después de *Lesson3*. El nivel de competencia alcanzado en esta última es 5. Si este valor supera o iguala el umbral de progreso, C_4 estará "listo para ser explorado".

En resumen, el objetivo del proceso de filtrado es descartar aquellos LOs indexados en la ontología por conceptos no alcanzables para el estudiante activo. Esta etapa de filtrado proporciona una manera de personalización *ligera* a largo plazo en esta primera fase de la estrategia de recomendación. De esta manera, ante una misma consulta formulada por dos estudiantes distintos, el conjunto de LOs recuperados podría variar significativamente en función del dominio de la materia que tenga cada uno.

Veamos un ejemplo de la etapa de recuperación. Supongamos que:

- en el repositorio tenemos los LOs representados en la Figura 13,
- el perfil de estudiante junto con el itinerario de aprendizaje y los valores de competencia de las lecciones que componen el itinerario computados a partir del perfil, se encuentran representados en la Figura 14,
- el umbral de progreso se establece en 5,
- y la consulta propuesta por el estudiante es $Q = \{C_{3,3}\}$.

Para facilitar el seguimiento del ejemplo, en la Figura 16 reunimos la Figura 13 y la Figura 14.

Con estos datos, y asumiendo un proceso de búsqueda aproximado que permitiera subir hasta un nivel en la taxonomía, el conjunto de LOs recuperados sería $CR = \{$L01, L03, L04$\}$. L01 y L03 forman parte del conjunto debido a la recuperación aproximada – no cubren el concepto $C_{3,3}$ de la consulta pero sí $C_{3,1}$ y $C_{3,2}$ que son conceptos hermanos de $C_{3,3}$. Además todos los conceptos que cubren son conceptos "listos para ser explorados", por lo que no son eliminados en el proceso de filtrado. L02 se descarta ya que, aunque cubre el concepto $C_{3,3}$, cubre, entre otros, el concepto $C_{5,1}$, que no está listo para ser explorado: según el itinerario de aprendizaje debería explorarse antes C_4 y éste tampoco está explorado, por lo cual $C_{5,1}$ todavía no está listo para ser explorado. Veamos ahora qué sucede con L04. Los conceptos cubiertos por este LO son $\{C_2, C_{3,3}, C_4\}$. Al cubrir el concepto de la consulta es candidato a formar parte de *CR*, pero además cubre dos conceptos más: C_2 que es un concepto ya explorado con un nivel de competencia igual a 8 (ver Figura 16); y C_4 que no es un concepto explorado. C_4 es un concepto de la ontología perteneciente a *Lesson4*, y según el criterio definido podremos explorar C_4 si el nivel de competencia de la lección predecesora supera el umbral de progreso. El nivel de competencia de *Lesson3* es 5, es decir, es igual que el umbral de

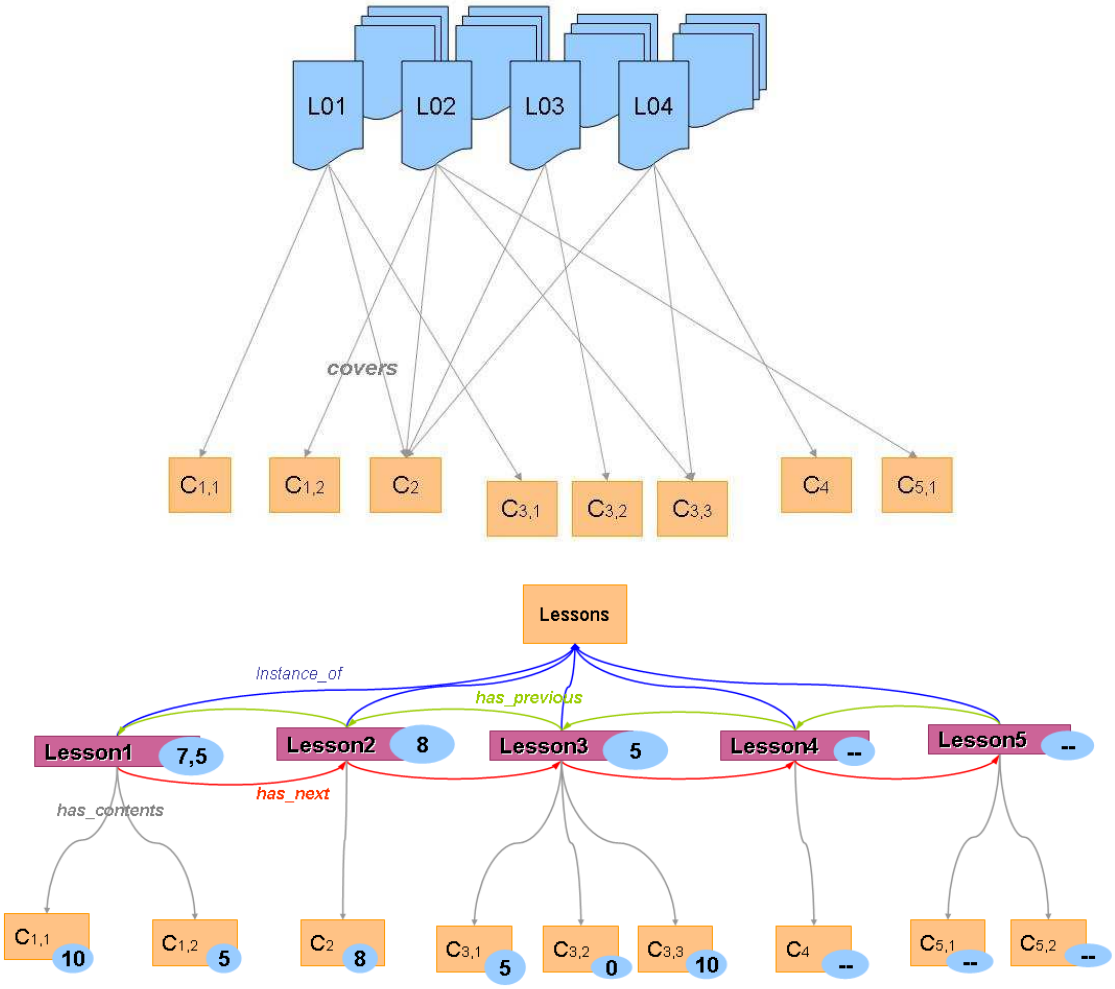


Figura 16. En la parte superior tenemos algunos LOs contenidos en el repositorio. En la parte inferior tenemos una vista parcial del perfil del estudiante.

progreso definido y, en consecuencia, el concepto C_4 está "listo para ser explorado" y L04 se incluye en *CR*.

Si esa misma consulta hubiese sido formulada por un estudiante que ya hubiese demostrado cierto nivel de conocimiento del concepto C_4, el conjunto *CR* incluiría L02 pues el concepto $C_{5,1}$ estaría listo para ser explorado con la estrategia definida. Vemos así, como hemos señalado antes, que el proceso de filtrado permite introducir una mínima personalización a largo plazo desde esta primera etapa de recuperación.

3.4.2. Etapa de ordenación

La etapa de ordenación asigna una valoración o relevancia, que llamaremos *calidad*, a cada LO obtenido en la etapa anterior y los ordena de acuerdo a dicho valor, obteniendo así el conjunto final que será mostrado al estudiante.

A fin de calcular la calidad de un LO *L* para un estudiante *S* que ha proporcionado una consulta *Q* hemos seleccionado una métrica de calidad definida como la suma ponderada de dos términos: la similitud (*Similitud*) entre *Q* y los conceptos cubiertos por *L*, y la utilidad pedagógica (*UP*) de *L* con respecto al estudiante *S*. Esta métrica de calidad se basa en la propuesta por [80] para la introducción de diversidad en las recomendaciones y ha sido adaptada como una manera de incluir personalización a largo plazo en la recomendación.

$$(1) \quad Calidad(L,S,Q) = \alpha \cdot Similitud(L,Q) + (1-\alpha) \cdot UP(L,S) \; donde \; \alpha \in [0,1]$$

A continuación pasamos a exponer cómo hemos resuelto en esta propuesta el cálculo de *Similitud(L,Q)* y *UP(L, S)*.

El cómputo de la similitud *Similitud(L,Q)* entre los conceptos recogidos en la consulta *Q* y los conceptos que *L* cubre requiere calcular la similitud entre dos conjuntos de conceptos. Existen diferentes métricas que pueden usarse en casos como éste. En concreto, una simplificación consiste en comparar los conceptos individuales que resultan de la conjunción de cada uno de los conjuntos de conceptos –el concepto que resulta de la conjunción de los conceptos de la consulta (*Q_conj_conceptos*) y el concepto que resulta de la conjunción de los conceptos que cubre *L* (*L_conj_conceptos*)– en lugar de comparar directamente dos conjuntos de conceptos. Hemos decidido utilizar la métrica de similitud definida en [35], donde *super(Q_conj_conceptos)* representa el conjunto de todos los conceptos contenidos en la ontología que son superconceptos de *Q_conj_conceptos* y *super(L_conj_conceptos)* contiene todos los conceptos contenidos en la ontología que son superconceptos de *L_conj_conceptos*:

$$(2) \quad Similitud(L,Q) = \frac{|super(Q\ conj\ conceptos) \cap super(L\ conj\ conceptos)|}{\sqrt{|super(Q\ conj\ conceptos)|} \cdot \sqrt{|super(L\ conj\ conceptos)|}}$$

El valor de *Similitud(L,Q)* se encuentra en el intervalo [0,1].

La *Similitud* calcula la relevancia de un LO debida a los objetivos de la sesión (reflejados en la consulta propuesta por el estudiante) que *L* satisface, lo que hemos llamado los objetivos a corto plazo. Cuanto mayor es el número de conceptos de la consulta que *L* permite aprender, mayor será el valor de la similitud. Cuanto más similares sean los conceptos que cubre *L* y los conceptos explicitados en la consulta, mayor será el valor de similitud. Como podemos observar, el conocimiento de la

similitud entre los conceptos representados en la ontología es crucial en nuestro contexto.

La utilidad pedagógica se refiere a lo adecuado que es el LO para el estudiante en función de su perfil. A fin de medir la utilidad pedagógica que el LO *L* muestra para un estudiante *S* dado, *UP(L, S),* hemos adoptado una estrategia instructiva que promueve completar las carencias de conocimiento del estudiante incluyendo conocimientos de refuerzo [74]. El objetivo es asignar una utilidad pedagógica mayor a *L* si éste cubre conceptos en los cuales el estudiante ha demostrado tener un bajo nivel de competencia. De esta manera, *L* puede ayudar al estudiante a reforzar sus conocimientos sobre esos conceptos y así alcanzar sus objetivos de aprendizaje a largo plazo. Este refuerzo de conocimiento podría asignar prioridad a LOs que el estudiante todavía no ha explorado, o puede repartirla de una manera igualitaria entre los LOs explorados y no explorados. En el primer caso la información almacenada en el perfil del estudiante acerca del histórico de navegación sería de gran importancia.

Hemos optado por calcular la utilidad pedagógica de acuerdo a la siguiente fórmula.

$$(3) \quad UP(L,S) = 1 - MAN(L,S)$$

donde *MAN (L, S)* es la media aritmética normalizada de los niveles de competencia alcanzados por el estudiante *S* en los conceptos cubiertos por *L*, de modo que *MAN (L, S)* estará en el intervalo [0, 1]. *UP (L, S)* también toma valores entre 0 y 1.

Siguiendo con el ejemplo utilizado en la subsección anterior, tenemos que *CR* = {L01, L03, L04}. Para conocer la calidad de cada uno de estos LOs de acuerdo a (1) tendremos que calcular para cada LO tres valores: *MAN(L,S)*, *UP(L,S)* de acuerdo a (3) y *Similitud(L,Q)* de acuerdo a (2). Ejemplificaremos el resultado con un LO y después en la Tabla 1 podemos ver el resultado de calcular estos valores previos para todos los LOs de *CR*.

$$MAN(L01,S) = \left(\frac{10+8+5}{3}\right)\cdot 0{,}1 = 0{,}76$$

$$UP(L01,S) = 1 - MAN(L01,S) = 1 - 0{,}76 = 0{,}24$$

$$Similitud(L01,Q) = \frac{|super(Q\ conj\ conceptos) \cap super(L01\ conj\ conceptos)|}{\sqrt{|super(Q\ conj\ conceptos)|}\cdot\sqrt{|super(L01\ conj\ conceptos)|}} = \frac{|\{Conceptos\ C_3\ C_{3,3}\} \cap \{Conceptos\ C_1\ C_{1,1}\ C_2\ C_3\ C_{3,1}\}|}{\sqrt{3}\cdot\sqrt{6}} = 0{,}471$$

Debemos señalar que para calcular el valor de *MAN(L,S)* cuando *L* cubre conceptos que todavía no han sido explorados (es decir, no tienen un nivel de competencia asociado) se ha adoptado el criterio de asignar a dichos conceptos un valor de nivel de competencia intermedio en la escala considerada. De esta manera los resultados de las

valoraciones no se verán influenciados por estos conceptos que carecen de nivel de competencia.

Podemos observar en la Tabla 1 que el LO que tiene mayor *UP* es L03. Esto es debido a que el nivel de competencia alcanzado en los conceptos que cubre es menor que para los conceptos que cubren L01 y L04. En particular, L03 cubre el concepto $C_{3,2}$ que tiene un nivel de competencia igual a 0, y según la estrategia definida para *UP,* este LO debe tener mayor utilidad pedagógica, para así completar las carencias de conocimiento demostradas en $C_{3,2}$. Si nos fijamos en los valores alcanzados para la *Similitud*, vemos que el que mejor resultado ha obtenido ha sido L04 ya que este LO es el único que cubre exactamente, entre otros, el concepto solicitado en la consulta $C_{3,3}$.

Tabla 1. LOs de *CR* junto con la media aritmética normalizada (*MAN*), la utilidad pedagógica (*UP*) y la similitud con la consulta.

LOs	MAN	UP	Similitud
L01	0.76	0.24	0.471
L03	0.4	0.6	0.577
L04	0.76	0.24	0.774

La medida de calidad resultante definida en (1), junto con la *Similitud* (2) y la utilidad pedagógica expuesta en (3), permite introducir un considerable grado de personalización en la etapa de ordenación. La calidad de un LO finalmente propuesto para un estudiante depende parcialmente de la utilidad pedagógica que el LO tiene para conseguir los objetivos a largo plazo del estudiante. La influencia final de la utilidad pedagógica y, como consecuencia, el nivel de personalización conseguido en la lista final de LOs recomendados puede ser controlado por medio del valor asignado al peso α utilizado en (1).

Valores bajos de α le dan la prioridad a la utilidad pedagógica en contra de la similitud con la consulta. En particular, $\alpha = 0$ representa el nivel más alto de personalización a largo plazo, y en este caso, la consulta (objetivos para la sesión) sólo se utiliza en la etapa de recuperación. Esto asegura que el recomendador propone LOs que cumplan con los objetivos de la sesión a un nivel mínimo, aunque el orden en el cual son propuestos al estudiante está totalmente influenciado por los objetivos a largo plazo que permiten alcanzar (su capacidad de refuerzo de conocimientos ya manejados). Por el contrario, valores altos de α dan la prioridad a la similitud con la consulta frente a la utilidad pedagógica.

Una vez que el valor de α utilizado en (1) se fija, el recomendador muestra un comportamiento con respecto al tipo de personalización que ofrece. Podemos obtener un comportamiento más flexible si, en un recomendador dado, α puede tomar diferentes

valores. De esta manera, el recomendador muestra una mayor adaptabilidad para los potenciales usuarios. Por ejemplo, el valor de α puede depender del tipo de estudiante que utiliza el recomendador. Valores altos de α puede ser apropiados para los *buenos* estudiantes, aquellos cuyos perfiles muestran un buen rendimiento. Estos estudiantes rara vez necesitaran reforzar conocimientos y el recomendador podría centrarse en sus objetivos de aprendizaje de la sesión dando prioridad a aquellos LOs que están más correlacionados con la consulta. Por el contrario, valores bajos de α pueden ser apropiados para estudiantes con bajo rendimiento, de tal manera que el recomendador fomente solventar sus carencias de conocimiento sin comprometer significativamente los intereses del estudiante reflejados en la consulta.

3.5. Estrategia de recomendación proactiva que explota la diversidad y la navegación por propuesta

En esta segunda estrategia [65] se ha explorado la forma de afrontar otra de las debilidades de la estrategia de recomendación basada en contenido descrita en [31, 32]: la carencia de diversidad en la recomendación. Al mismo tiempo, se ha aprovechado para ensayar una aproximación proactiva de las recomendaciones como alternativa o complemento a la naturaleza reactiva de las otras estrategias diseñadas. Veamos seguidamente cómo se han diseñado la naturaleza proactiva de la estrategia y la inclusión de la diversidad.

En esta estrategia, el estudiante, al comenzar una sesión de recomendación, recibirá un conjunto de LOs propuestos como *actividades del día*, es decir, un conjunto de recursos educativos que se proponen al estudiante para que continúe con su proceso de aprendizaje. Siendo el recomendador el que lleva la iniciativa queremos proporcionar una ayuda adecuada a aquellos estudiantes que tienen una actitud pasiva o que tienen poco conocimiento del dominio como para proponer una consulta al recomendador.

Después de esta primera recomendación el recomendador otorga al estudiante la posibilidad de refinar el conjunto de LOs propuestos para que así pueda mostrar sus preferencias con respecto a la recomendación. De esta manera el recomendador engancha al estudiante en un proceso conversacional guiado por las preferencias que éste vaya mostrando con respecto a cada recomendación.

En la Figura 17 podemos ver un esquema de funcionamiento de esta estrategia. El recomendador propone un conjunto de LOs de acuerdo al perfil del estudiante. Ante este conjunto el estudiante tiene dos opciones, seleccionar un LO para trabajar en él, o refinar la propuesta seleccionando un LO y diciendo que quiere "*más como éste*" (*more like this*). Si se da la primera opción llegamos al final del proceso de recomendación. Si, por el contrario, el alumno proporciona una preferencia para refinar la recomendación el recomendador, de nuevo, propondrá otro conjunto de LOs al estudiante como resultado

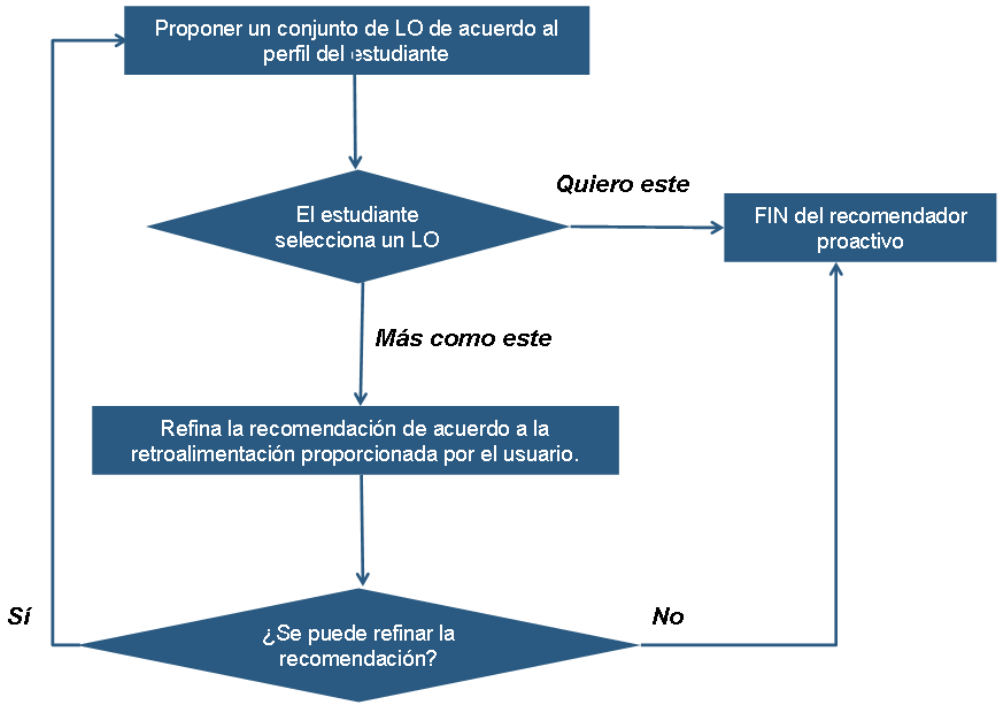


Figura 17. Diagrama de interacción usuario-recomendador para la estrategia proactiva.

del refinamiento del conjunto anterior. Este proceso conversacional termina cuando el estudiante selecciona un LO para practicar o bien si el recomendador no puede seguir refinando el conjunto propuesto.

Como vimos en el capítulo anterior, existen dos tipos de estrategias para los recomendadores conversacionales: navegación por pregunta o navegación por propuesta. Hemos elegido la alternativa de navegación por propuesta al considerarla más apropiada en dominios complejos donde el usuario puede no estar suficientemente preparado para contestar preguntas sobre sus requisitos ya que el conocimiento del dominio por parte del usuario puede ser insuficiente. Esto está en consonancia con la audiencia a la que va dirigida fundamentalmente esta estrategia: estudiantes poco motivados o con un conocimiento insuficiente del (vocabulario del) dominio y que tengan dificultades para plantear consultas.

La diversidad buscada para el conjunto recuperado se puede conseguir, en nuestro contexto, de distintas formas: con LOs pertenecientes a recursos educativos distintos (presentaciones, ejemplos resueltos, ejercicios para practicar, vídeos, etc...), LOs que

permitan reforzar o aprender conceptos distintos, LOs que pertenezcan a categorías diferentes de la taxonomía de Bloom (en función del estado cognitivo del estudiante), etc. Presentamos a continuación las decisiones tomadas en relación a la inclusión de diversidad.

Para promover la diversidad del conjunto recuperado, en cada fase de recomendación el repositorio de LOs se divide en grupos. A partir de estos grupos, el conjunto recuperado estará formado por LOs que pertenezcan o sean representantes de grupos distintos. Esta división se lleva a cabo de tres maneras distintas dependiendo de la fase de la estrategia en la que nos encontremos:

- En la primera fase, el repositorio se divide en LOs que cubran conceptos ya estudiados, que los identificaremos como *LOs de refuerzo*, y en LOs que cubran nuevos conceptos que pueden ser explorados por el estudiante, identificados como *LOs de descubrimiento*. Para identificar cuáles son los conceptos ya estudiados y cuáles los nuevos nos guiaremos por el itinerario de aprendizaje y el perfil de estudiante. De cada uno de los grupos se seleccionan uno o más LO representantes y esos serán los mostrados al estudiante.
- En la segunda fase, a la cual llegamos si el estudiante ha decidido seguir refinando las propuestas de la primera fase, se produce un recorrido por niveles en la taxonomía de conceptos de la ontología del dominio. Existe una etapa previa a este recorrido que considera la división del repositorio en grupos, cada uno de los cuales se corresponde con las lecciones definidas en el itinerario de aprendizaje. Después de que el estudiante haya seleccionado un LO de refuerzo o de descubrimiento, el recomendador genera una nueva propuesta utilizando LOs que cubren conceptos de cada una de las lecciones del grupo al que pertenecía el LO usado para refinar. El repositorio se divide así en tantos grupos como lecciones existan asociadas a refuerzo o asociadas a descubrimiento. En las posteriores etapas el recomendador genera una nueva propuesta utilizando LOs que cubren conceptos del nivel más alto de la taxonomía de conceptos dentro del grupo al que pertenecía el LO usado para refinar. El repositorio se divide así en tantos grupos como conceptos existan en el nivel de la taxonomía de conceptos del dominio en el que nos encontremos. De cada grupo se seleccionarán uno o más LOs representantes. Esta fase se ejecuta hasta que el estudiante elija para seguir refinando un LO que sea representante de un grupo de conceptos formado por un concepto hoja en la ontología, o hasta que el estudiante seleccione un LO para practicar.
- En la última fase, cuando ya no se puede alcanzar diversidad mediante los conceptos cubiertos por los LOs, el conjunto de LOs que son accesibles desde el concepto hoja alcanzado se divide atendiendo a las categorías de la versión revisada de la taxonomía de Bloom, y en cada grupo estarán aquellos LOs que cubran el concepto hoja, entre otros, y pertenezcan a la categoría correspondiente

de la taxonomía de Bloom. Como en ocasiones anteriores, el conjunto de LOs propuestos al estudiante estará formado por LOs pertenecientes a los distintos grupos. En el caso de que el estudiante esté seleccionando LOs para descubrir un nuevo concepto, se tenderá a seleccionar aquellos LOs que pertenecen a las categorías más bajas en la taxonomía de Bloom (es decir, *recordar* y *comprender*), debido a la idoneidad de estos para iniciar el aprendizaje de un nuevo concepto. Si, por el contrario, nos encontramos reforzando conceptos se tenderá a seleccionar LOs de las categorías Bloom superiores (i.e. *aplicar*, *analizar*, *evaluar* y *crear*).

Como ya hemos indicado, la navegación por propuesta permite que el estudiante muestre sus preferencias con respecto al conjunto de LOs propuestos indicando que desea *"más como éste"*. Para que el estudiante pueda realizar una elección coherente con la forma de trabajar de la estrategia es conveniente que cada LO propuesto lleve asociada una explicación de por qué o en calidad de qué se propone. En nuestro caso concreto se tratará del motivo por el cual dicho LO fue considerado representante del grupo correspondiente. Esto es: supongamos que un LO se añade al conjunto recuperado porque refuerza conceptos; si a continuación el estudiante selecciona este LO para seguir refinando la propuesta, la estrategia "entenderá" que quiere más LOs que le permitan reforzar conceptos. Esto también redundará, como ya se adelantaba en el Capítulo 2, en una mayor confianza y entendimiento de las recomendaciones que se hagan.

En las fases primera y segunda de esta estrategia se podrían seleccionar los LOs que representan a cada grupo y forman parte de la recomendación atendiendo a diferentes criterios entre los que podrían encontrarse, por ejemplo, aquellos que:

- Cubran un mayor número de conceptos.
- Demuestren tener una mayor utilidad pedagógica según lo descrito en la Sección 3.4.2.
- Cubran los conceptos con peor nivel de competencia en el perfil del alumno, dentro de los conceptos que se estén considerando en cada momento.
- No figuren en el historial de navegación del estudiante actual.
- Se adapten al perfil del estudiante en función de la categoría de la taxonomía de Bloom a la que pertenezca; etc.

Veamos un ejemplo de funcionamiento de esta estrategia, suponiendo que:

- ahora en el repositorio tenemos, entre otros, los LOs mostrados en la Figura 18,
- el umbral de progreso se establece en 5,
- siempre que sea posible el conjunto recuperado estará formado por cuatro LOs, y

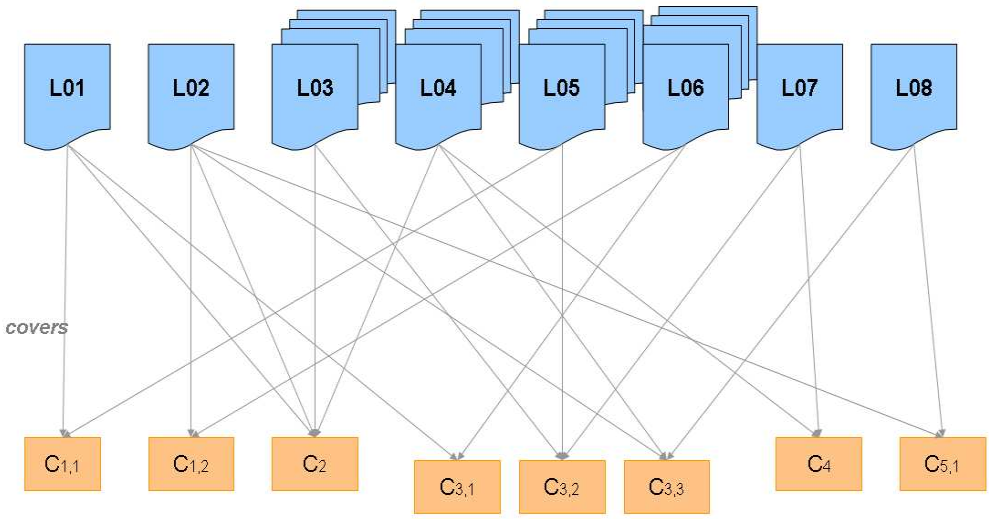


Figura 18. LOs junto con los conceptos que cubre cada uno

- el criterio de selección para un LO es que sea el que mayor número de conceptos cubre.

Para facilitar el seguimiento del ejemplo se ha vuelto a traer el contenido de la Figura 10 a la Figura 19. En la primera interacción la estrategia de recomendación divide los conceptos del itinerario de aprendizaje en conceptos de refuerzo y conceptos de descubrimiento (ver Figura 20 donde figura representado el perfil del estudiante de forma gráfica junto con los valores de competencia de las lecciones que componen el itinerario computados a partir del perfil). Lo primero que debe averiguar la estrategia es si este estudiante puede descubrir nuevos conceptos o no. En este caso concreto, en el perfil del estudiante vemos que ha alcanzado a estudiar todos los conceptos relativos a *Lesson3*. El nivel de competencia alcanzado para *Lesson3* es 5. Según el umbral de progreso y el itinerario de aprendizaje definido, tenemos que este estudiante está listo para descubrir nuevos conceptos referidos a *Lesson4*.

En este caso, como resultado de esta primera interacción tenemos que el conjunto propuesto será CR = {L01, L03, L04, L07}.

L04 y L07 están incluidos por pertenecer al grupo de descubrimiento ya que cubren C_4. L01 se incluye debido a que permite reforzar conceptos y de estos es el que mayor número conceptos cubre. L03 también permite reforzar conceptos y después de L01 es uno de los que mayor número de conceptos cubre.

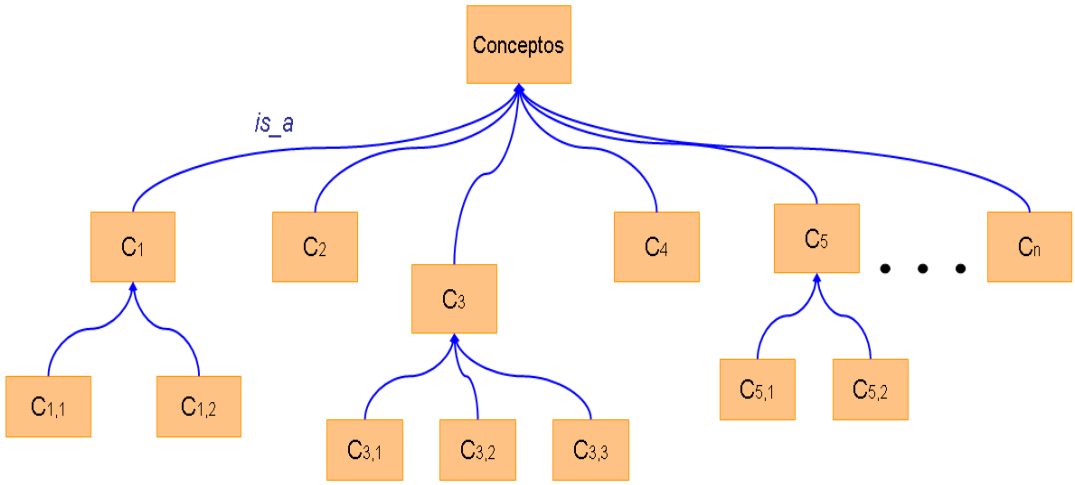


Figura 19. Ejemplo de una taxonomía de conceptos

Supongamos que el estudiante decide refinar *CR* y selecciona L01 porque pertenece al grupo de refuerzo. Así obtenemos una división en tres grupos (ver Figura 21) cada uno de los cuales se corresponde con los conceptos de más alto nivel en la taxonomía –C_1, C_2 y C_3– relacionados con los conceptos que cubren las tres lecciones ya estudiadas, *Lesson1*, *Lesson2* y *Lesson3* respectivamente. En este caso tenemos que

CR = {L01, L03, L05, L06}

L02, L04, L07 y L08 han sido descartados por cubrir conceptos que no sólo pertenecen a las lecciones ya estudiadas. L01 nos permite reforzar subconceptos del concepto C_1, L03 nos permite reforzar subconceptos de C_2 y L05 y L06 nos permiten reforzar subconceptos de C_3.

Supongamos que en la siguiente interacción el estudiante decide refinar la consulta solicitando más LOs como L03. Este LO cubre un concepto hoja, C_2. En este momento otros LOs del repositorio que, como L03, sirvan para cubrir C_2 son agrupados de acuerdo a las seis categorías de la versión revisada de la taxonomía de Bloom. *CR* estará formado por cuatro LOs que cubran al concepto C_2 y pertenezcan a grupos distintos. En este caso, ya que nos encontramos reforzando conceptos y según el compromiso adoptado anteriormente, sería conveniente utilizar LOs de grupos que pertenezcan a los niveles superiores de la taxonomía de Bloom, ya que se supone que las habilidades asociadas a los niveles inferiores de dicha taxonomía deberían haber sido alcanzadas previamente.

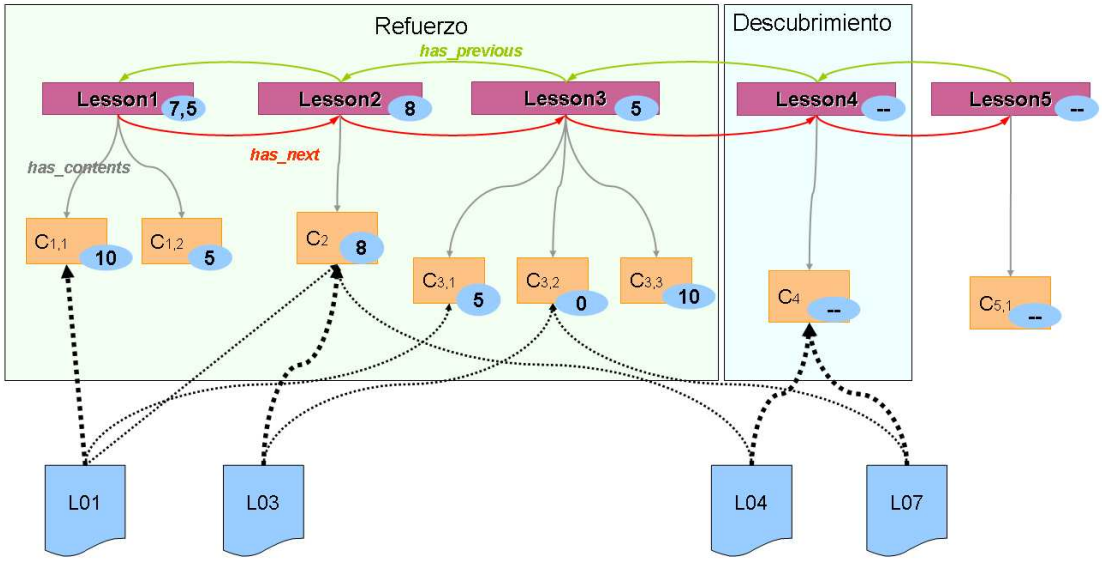


Figura 20. División del itinerario de aprendizaje en conceptos de refuerzo y conceptos de descubrimiento.

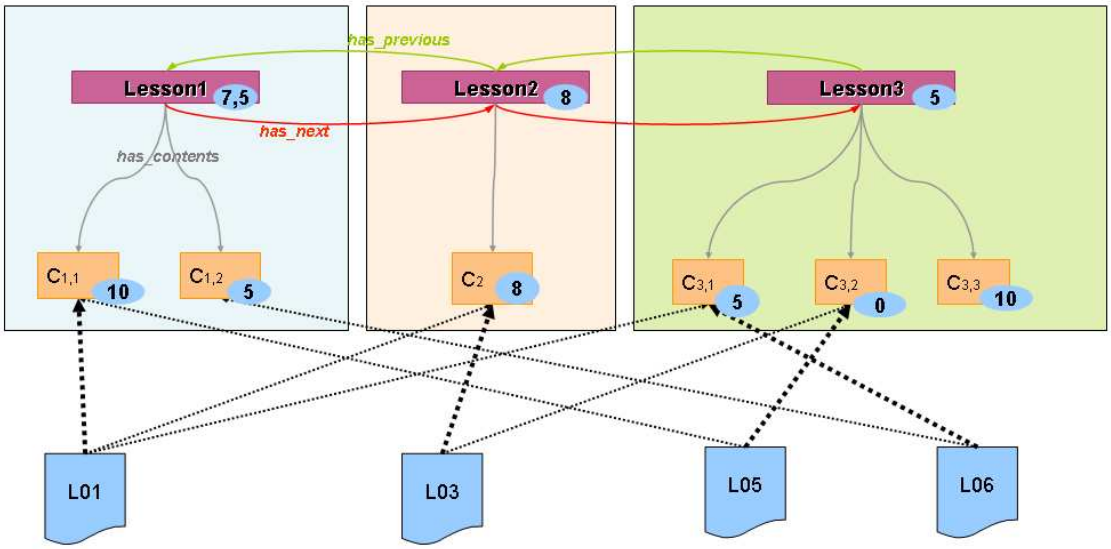


Figura 21. División del itinerario de aprendizaje en conceptos de refuerzo.

3.6. Conclusiones

En este capítulo hemos descrito dos aproximaciones o estrategias de recomendación de LOs que intentan afrontar los aspectos mejorables de la recomendación basada en contenido empleada en el enfoque híbrido descrito en [31, 32]. Antes de presentar cada estrategia se han descrito las fuentes de conocimiento necesarias: la ontología del dominio, los LOs y sus metadatos, y, por último, el perfil de estudiante.

Como hemos indicado, la estrategia reactiva presentada en la Sección 3.4 [66, 67] permite incorporar distintos niveles de personalización en el recomendador. El estudiante debe formular una consulta en la que reflejará cuáles son sus objetivos a corto plazo, es decir, qué es lo que quiere aprender en esa sesión de aprendizaje. Estos objetivos se corresponderán con conceptos de la ontología. Los objetivos a largo plazo se extraen del perfil del estudiante. El conjunto inicial de LOs obtenido a partir de la consulta es filtrado y sólo aquellos LOs que cubren conceptos de la ontología "listos para ser explorados" por el estudiante serán finalmente considerados en la etapa de ordenación. Este proceso de filtrado permite incorporar cierto nivel de personalización desde la primera etapa del proceso de recomendación (la etapa de recuperación), en la línea de mejoras que se pretendían realizar sobre la estrategia basada en contenido presentada en [31, 32]. El segundo proceso que constituye esta estrategia, el proceso de ordenación, emplea una medida de calidad que combina similitud con la consulta y utilidad pedagógica de los LOs recuperados. La inclusión de la utilidad pedagógica permite reforzar aún más la consecución de objetivos a largo plazo, y, en definitiva, la personalización o adaptación al estudiante. Para valorar la utilidad pedagógica se ha seguido una estrategia instructiva que, como forma de conseguir objetivos a largo plazo, promueve suplir las deficiencias de conocimiento valorando mejor a aquellos LOs recuperados que permitan reforzar el conocimiento de los conceptos en los que el estudiante ha demostrado un bajo nivel de competencia. Evidentemente, esta estrategia de refuerzo, como forma de conseguir incluir objetivos a largo plazo, sería intercambiable en el marco definido por otras estrategias instructivas sin más que cambiar la forma de valorar la utilidad pedagógica en la medida de calidad de la estrategia.

La influencia final de la utilidad pedagógica en la valoración de la calidad del LO, según la medida de calidad definida en (1), y, como consecuencia, el nivel de personalización conseguido en la lista final de LOs recomendados, puede ser controlado por medio del valor asignado al peso α utilizado en (1). Una vez que el valor de α se fija, el recomendador muestra un comportamiento con respecto al tipo de personalización que ofrece. Ahora bien, es posible obtener un comportamiento más flexible si, en un recomendador dado, se permite que α tome diferentes valores. De esta

manera, el recomendador sería capaz de mostrar una mayor adaptabilidad a los potenciales usuarios.

La estrategia proactiva [65], descrita en la Sección 1.1, promueve la diversidad del conjunto recuperado y emplea la navegación por propuesta como modelo de interacción con el estudiante, afrontando así las otras dos potenciales deficiencias de la estrategia inicial basada en contenido. Esta navegación engancha al estudiante en un proceso conversacional en el que podrá refinar el conjunto propuesto. El hecho de que la estrategia opere de manera proactiva ayuda a estudiantes poco motivados o que no tiene un conocimiento suficiente del dominio como para proponer una consulta, a que continúen su proceso de aprendizaje.

Para fomentar la diversidad la estrategia actúa en 3 fases. En una primera fase se introduce diversidad atendiendo al carácter de refuerzo o de descubrimiento de nuevo conocimiento que puede asociarse a cada LO del repositorio, para lo cual es necesario tener en cuenta el perfil del estudiante. En una segunda fase, la diversidad se consigue proponiendo LOs que pueden servir de representantes de distintas partes de la taxonomía de conceptos del dominio. En cada iteración de esta fase se trabaja a un nivel de la taxonomía. Cuando no se puede conseguir más diversidad de esta forma por haber llegado a las hojas de la taxonomía del dominio, la clasificación que según la taxonomía de Bloom se ha realizado de los LOs es empleada como forma de conseguir diversidad.

Estas dos estrategias basadas en contenido pueden sustituir la existente en el recomendador híbrido inicial. Cualquiera de estas dos estrategias podría encajar dentro de ese recomendador híbrido para, a través de una segunda estrategia de CF, refinar el conjunto propuesto mediante las valoraciones otorgadas por los estudiantes a los distintos LOs.

Capítulo 4

RECOMENDACIÓN DE OBJETOS DE APRENDIZAJE EN EL CONTEXTO DE LA ENSEÑANZA DE LA PROGRAMACIÓN

Muchos docentes coinciden en que el aprendizaje de la Programación es una parte fundamental en el aprendizaje de la Informática y una tarea difícil más allá del paradigma de programación empleado [54]. Por eso, en los últimos años, muchos esfuerzos se han centrado en complementar las estructuras convencionales de los cursos de Programación (basados, en gran medida, en clases magistrales y trabajos prácticos de laboratorio) con nuevas aproximaciones pedagógicas y herramientas que sacan provecho de las Tecnologías de la Información y la Comunicación (TIC).

Los primeros sistemas de dedicados a la enseñanza de la Programación basados en ejemplos tenían una interfaz sencilla que seleccionaba ejemplos relevantes y proporcionaba búsquedas a partir de palabras claves que aparecían en el enunciado del problema o en el código del mismo [25]. Estas herramientas no tenían en cuenta el conocimiento actual del estudiante, así que podían recuperar ejemplos que trataban conceptos que el estudiante todavía no sabía, o incluso conceptos que todavía no estaba preparado para aprender. Otros entornos dejaban en manos del estudiante la responsabilidad de buscar los ejemplos permitiéndole explorar el repositorio completo a su gusto. WebEx [15], un sistema para explorar ejemplos de programación comentados desarrollados en el lenguaje C, es un ejemplo de este tipo de entornos. Recientes investigaciones en recursos educativos han señalado la necesidad de la adaptación en los entornos educativos. Más recientemente, NavEx [92], una evolución de WebEx, clasifica ejemplos de acuerdo al estado actual de los conocimientos del estudiante y su

historia pasada de interacciones con el sistema. Este sistema aplica navegación adaptada para: (a) distinguir nuevos ejemplos de ejemplos que han sido parcial o totalmente explorados y (b) categorizar ejemplos que están "listos para" o "todavía no preparados para" ser explorados de acuerdo al estado actual de los conocimientos del estudiante. Algunos repositorios educativos utilizan navegación social para resolver el problema de encontrar el siguiente ejemplo a explorar. Este es el caso de Knowledge Sea [16], una plataforma de acceso electrónico a documentos sobre el lenguaje de programación C que incorpora navegación social. El motor de búsqueda de Knowledge Sea utiliza un vector de valoraciones para ordenar los LOs recuperados de acuerdo a su relevancia expresada mediante indicaciones visuales.

En este capítulo veremos cómo aplicar las estrategias de recomendación descritas en el Capítulo 3 a un repositorio de LOs de Programación. En la Sección 1.1 contaremos la evolución del repositorio describiendo el estado y contenidos actuales. Después explicaremos cómo se aplica el conocimiento necesario para el recomendador para el caso concreto de la enseñanza de la Programación (Sección 1.1). En las Secciones 4.3 y 0 mostraremos un ejemplo de uso para las dos estrategias basadas en contenido explicadas en el capítulo anterior. Finalizaremos el capítulo con unas pequeñas conclusiones (Sección 4.5).

4.1. Un repositorio de recursos educativos de Programación a disposición de los alumnos de la Universidad Complutense de Madrid

En el entorno universitario español es usual la implantación de TIC, básicamente en forma de Campus Virtuales (CVs), como una vía para extender los servicios y funciones del campus universitario en un conjunto de espacios y herramientas en Internet que sirven de apoyo al aprendizaje, la investigación y la gestión docente [51].

El CV de la Universidad Complutense de Madrid (UCM) ha sido el mecanismo usado para introducir las TIC en la docencia de asignaturas de la materia "Introducción a la programación" existentes en diversas titulaciones de dicha universidad. La motivación nace no sólo de los cambios provocados por el Espacio Europeo de Educación Superior (EEES) para mejorar el proceso de aprendizaje de los estudiantes, sino también por las reiteradas peticiones de los propios estudiantes acerca de la disponibilidad de material que facilite su autoaprendizaje.

Un equipo docente formado por un grupo de profesores del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la UCM, con los que la autora del presente trabajo ha colaborado en los últimos tiempos, ha trabajado en los tres últimos años en sendos Proyectos de Innovación y Mejora de la Calidad Docente con el objetivo de desarrollar un repositorio de recursos didácticos

para asignaturas de Programación y la puesta a disposición del mismo a través del CV de la UCM. Estos recursos están pensados principalmente para asignaturas como: "Introducción a la Programación" y "Laboratorio de Programación I" de la Ingeniería en Informática, la Ingeniería Técnica en Informática de Sistemas y la Ingeniería Técnica en Informática de Gestión; "Fundamentos de Programación" de la Licenciatura en Física y de la Ingeniería en Electrónica; y "Programación" de la Licenciatura en Física. Este servicio multidisciplinar está disponible desde finales del curso 2005-2006 para una población de estudiantes variable según el año académico, llegando en alguna ocasión a superar los 700 estudiantes. En la Figura 22 tenemos un ejemplo de algunos de los recursos disponibles para los estudiantes durante el curso 2007-2008.

A lo largo de ese tiempo, ha ido evolucionando la forma de acceso a los recursos, siempre sujeto a las limitaciones que WebCT, el Sistemas de Gestión de Contenidos de Aprendizaje (LCMS, del inglés *Learning Content Manangement Systems*) empleado, impone. En un primer momento el material estaba accesible para que los alumnos navegaran libremente por el repositorio explorando los contenidos que quisieran. Así, tras obtener unos primeros recursos educativos en la que se satisfacía la necesidad de material práctico resuelto, el equipo docente involucrado introdujo una importante mejora didáctica: incorporar material práctico no resuelto, acompañado en ocasiones por guías para su resolución y la comprobación de su correcto funcionamiento. No obstante, esta ampliación de contenidos no se consideró suficiente para facilitar el autoaprendizaje. Trabajos como el presentado en [45] resaltan la necesidad de adaptarse al estilo y la capacidad de aprendizaje de cada estudiante, que cambia a lo largo del tiempo dependiendo del contexto en el que se desenvuelva. La VCP evolucionó para implementar un curso en el CV de la UCM que proporcionara un modelo secuencial de aprendizaje en el que hay que superar la realización de tests para avanzar por los contenidos. El resultado a día de hoy es un curso online multidisciplinar que soporta un modelo de aprendizaje activo basado en casos prácticos resueltos y a resolver.

Figura 22. Algunos de los contenidos almacenados en el CV.

Aunque los estudiantes valoran la ayuda que estos recursos les proporcionan, cerca del 70% de los estudiantes ha reconocido que echan en falta facilidades más sofisticadas para el acceso a los mismos y más del 40% coincidía en la conveniencia de que fuese algún tipo de acceso guiado por el conocimiento del estudiante [34].

Por este motivo, y tal y como se ha indicado en el capítulo anterior, dentro del Grupo de Aplicaciones de Inteligencia Artificial (GAIA) de la UCM se iniciaron trabajos en el campo de la recomendación de LOs que facilitasen el acceso a aquellos mejor adaptados a las necesidades del estudiante con el objetivo de que dichos trabajos tuviesen una aplicación directa en nuestras tareas docentes y, en particular, desembocaran en el desarrollo de recomendadores de LOs creados a partir de los recursos de Programación y que fueran usados posteriormente por nuestros alumnos para reforzar y facilitar su aprendizaje.

Fruto de estos trabajos se definió una aproximación híbrida de recomendación de LOs descrita en el capítulo anterior [31, 32], que combina en cascada un enfoque basado en contenido con un enfoque colaborativo. Las estrategias basadas en contenido presentadas en este trabajo refinan y mejoran la empleada en la aproximación híbrida inicial.

4.2. Aplicación a un repositorio de LOs en el ámbito de la programación: Las fuentes de conocimiento

En esta sección describiremos el proceso de aplicación de las dos aproximaciones generales descritas en el Capítulo 3 a un repositorio de LOs de contenidos educativos para la enseñanza de la Programación, en lo que respecta a las necesidades de conocimiento: la ontología (Subsección 4.2.1), LOs (Subsección 4.2.2) y el perfil del estudiante (Subsección 4.2.3).

4.2.1. Ontología de Programación

La ontología nos permite dotar al sistema de conocimiento sobre el dominio concreto de aplicación, conocimiento que, como vimos en el capítulo anterior, juega un papel importante en las estrategias de recomendación definidas. Para esta aplicación de las estrategias de recomendación definidas el conocimiento de la ontología será relativo al aprendizaje-enseñanza de la Programación. La ontología ha sido desarrollada en OWL mediante la herramienta Protégé 3.3.1[4].

Según la ontología definida en la Sección 3.3.1 tenemos dos clases importantes que habrá que adaptar a este dominio de aplicación:

- Una clase que representa los conceptos asociados a la enseñanza-aprendizaje de la Programación: la clase *ProgrammingConcept*. En la Figura 23 podemos ver una vista parcial de la jerarquía de conceptos definida. Cada individuo de esta clase tiene la propiedad *belongs_to* que nos indica a qué lenguaje de programación está asociado el concepto.

[4] http://protege.stanford.edu/

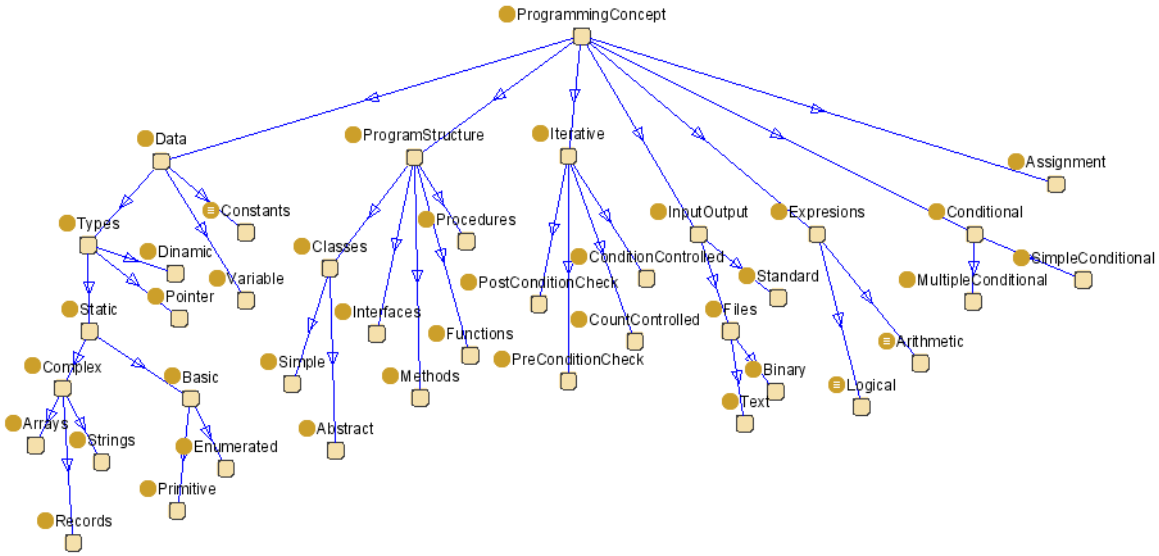


Figura 23. Vista parcial de los conceptos de programación definidos en la ontología.

- Las lecciones que componen el itinerario de aprendizaje. La clase *Lessons* será la encargada de representar las lecciones definidas. Cada lección se creará como un individuo de dicha clase que tendrá las siguientes propiedades:
    - *has_contents*: hace referencia a los conceptos de Programación que se trabajan en esa lección. En términos del perfil del estudiante, se refiere a los conocimientos que el estudiante ha de dominar/conocer al terminar el estudio de dicha lección.
    - *has_next* y *has_previous*: son las propiedades que permiten definir el secuenciamiento entre lecciones. Este secuenciamiento nos permitirá construir el itinerario de aprendizaje. Estas propiedades nos indican la lección que precede y sigue, respectivamente, a la lección actual.

Junto con estas dos clases, se han definido otras para representar conocimiento adicional sobre el dominio. En particular:

- Una clase que identifica los distintos itinerarios (o cursos) de enseñanza de Programación que pueden ser diseñados (*Programming Course*). Entendemos como un curso de programación la secuencia de todas las lecciones que conforman un itinerario de aprendizaje completo para la enseñanza de la Programación. Cada individuo de la clase tendrá dos propiedades:

  - *has_language*: referida al lenguaje de programación empleado en un determinado itinerario de aprendizaje.
  - *has_lessons*: las lecciones que componen el itinerario de aprendizaje.
- Una clase que identifica el lenguaje de programación que puede ser empleado en los contenidos (*Language*). Cada instancia de esta clase identificará a un lenguaje de programación. Un individuo de *Language* puede estar relacionado con un individuo de *ProgrammingCourse* mediante la relación *has_language*. Así mismo, un concepto de la jerarquía de conceptos se relaciona con un determinado lenguaje mediante la propiedad *belongs_to*. De esta manera podremos verificar si un curso está bien diseñado. Un curso con un lenguaje de programación concreto sólo puede tener lecciones que cubran conceptos definidos para ese lenguaje concreto. En la Figura 24 tenemos un ejemplo de esta última propiedad. Por un lado están dos conceptos de Programación representados como instancias de *Conditional* en la ontología (*switch* y *case*). Estos dos conceptos de Programación corresponden a lenguajes de programación diferentes: el primero pertenece al lenguaje C y el segundo al lenguaje Pascal, ambos representados como instancias de *Language*.

En la Figura 25 podemos ver de manera gráfica las relaciones definidas entre las distintas clases de la ontología.

4.2.2. Objetos de aprendizaje

Como apuntábamos en el capítulo anterior entendemos un LO como un recurso (generalmente digital) que puede ser usado y reutilizado como soporte a la enseñanza y al aprendizaje y que además es una experiencia de aprendizaje que contiene un objetivo y una actividad de aprendizaje. En nuestro repositorio tenemos LOs correspondientes a distintos tipos de recursos educativos relacionados con la Programación, como, por ejemplo:

- *Presentaciones*: notas teóricas sobre los conceptos de programación.
- *Ejercicios resueltos*: ejemplos de ejercicios resueltos para que el estudiante pueda analizar las soluciones.
- *Ejercicios para resolver*: ejercicios de programación propuestos para practicar. Existen distintos tipos de ejercicios:
  - *Ejercicios de desarrollo completo*: a partir del enunciado de un problema, el estudiante debe proponer una solución al mismo mediante el desarrollo completo de un programa.
  - *Ejercicios "fill in the gaps"*: ejercicios para completar trozos de código.

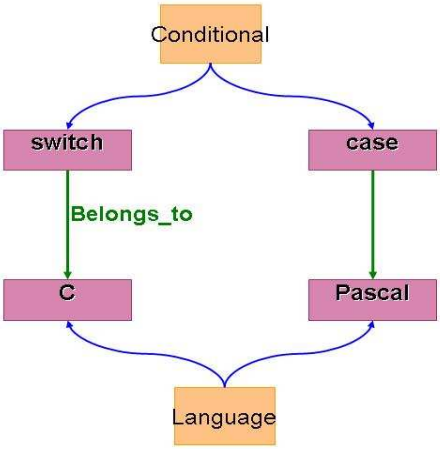


Figura 24. Vista de dos conceptos con el lenguaje de programación al que pertenecen.

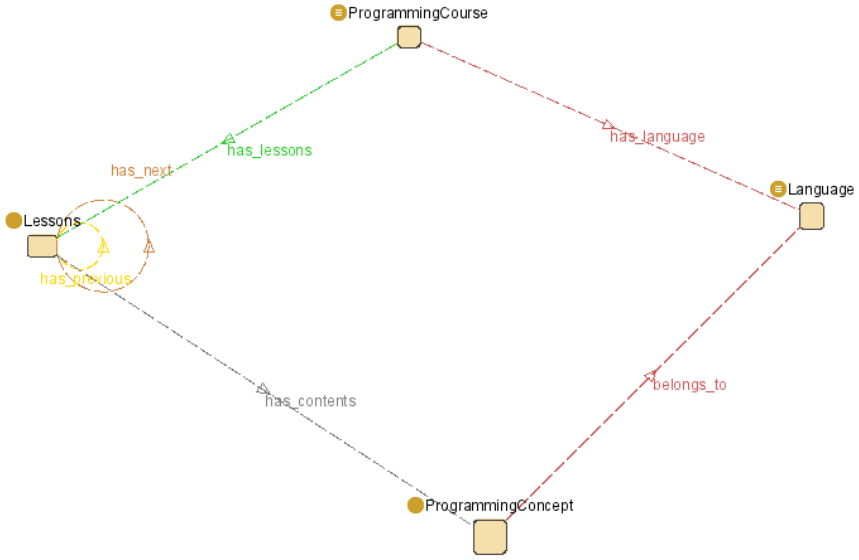


Figura 25. Esquema de las relaciones establecidas entre todas las clases de la ontología.

- *Ejercicios "encuentra el error":* ejercicios con algún error de compilación que el estudiante debe ser capaz de descubrir y solucionar.
- *Cuestiones*: breves preguntas para autoevaluar conocimientos.

Parte del trabajo de aplicación de las estrategias de recomendación al dominio de la Programación ha implicado realizar una adaptación de los recursos educativos existentes en el repositorio al estándar LOM. Para ello hemos utilizado un modelo XML

propuesto en [43] marcando así con metadatos los distintos LOs contenidos en el repositorio. Estos metadatos nos permitirán generar el conocimiento necesario sobre los distintos LOs que se verá reflejado en la ontología y será utilizado posteriormente para razonar sobre ellos. En el Anexo 1 aparecen algunos ejemplos de LOs.

La información englobada dentro de la categoría *General* nos proporciona información básica del LO (título, descripción, idioma del LO). Está información se le mostrará al estudiante cuando elija un LO para practicar. Pero también contiene un elemento importante para el recomendador: el metadato *keyword*. Las *keywords* indican qué conceptos de programación son cubiertos por un LO. Las *keywords* nos sirven de punto de enlace entre los LOs contenidos en el repositorio y los conceptos de programación definidos en la jerarquía de conceptos de la ontología de Programación.

El resto de categorías fueron explicadas en el capítulo anterior en la aproximación general, y son iguales para cualquier LO aunque no sea del ámbito de la Programación.

Por cada LO contenido en el repositorio se ha de crear una instancia en la ontología que lo represente. Este proceso se realiza de manera automática mediante una aplicación desarrollada en Java y que hace uso de la librería OntoBridge[5], analizando la categoría *keywords* de los metadatos de cada LO contenido en el repositorio. El contenido de este metadato nos dirá qué conceptos de los definidos en la ontología del dominio son cubiertos por ese LO. A continuación se completa la propiedad *covers* con los conceptos identificados.

4.2.3. El perfil del estudiante

El perfil de estudiante refleja el valor de competencia alcanzado en cada concepto en el que el estudiante ya ha sido evaluado, junto con el historial de LOs visitados.

En el sistema el perfil de estudiante se encuentra en un fichero en el que se almacena:

- Los objetivos involucrados en el proceso de aprendizaje reflejados como conceptos que debe aprender. Cada concepto tendrá asociado un valor numérico indicando el nivel de competencia alcanzado. Si un concepto no tiene asociado ningún valor numérico indica que dicho concepto todavía no ha sido explorado por el estudiante actual. El nivel de competencia se establece en el intervalo [0, 10].
- El historial de navegación. En el mismo fichero se escribirá el identificador asociado a cada LO ya visitado por el estudiante.

A partir de los niveles de competencia alcanzados en los conceptos es posible calcular el nivel de competencia de las lecciones que los engloban. En concreto,

[5] OntoBridge es una librería para Java, desarrollada por el grupo GAIA de la Universidad Complutense de Madrid, con el fin de manejar ontologías de una forma sencilla.

$$(1)\quad Calidad(L,S,Q) = \alpha \cdot Similitud(L,Q) + (1-\alpha)\cdot UP(L,S)\ donde\ \alpha \in [0,1]$$

$$(2)\quad Similitud(L,Q) = \frac{|super(Q\ conj\ conceptos) \cap super(L\ conj\ conceptos)|}{\sqrt{|super(Q\ conj\ conceptos)|}\cdot\sqrt{|super(L\ conj\ conceptos)|}}$$

$$(3)\quad UP(L,S) = 1 - MAN(L,S)$$

Figura 26. Resumen de las fórmulas utilizadas en la estrategia de recomendación reactiva.

calcularemos el nivel de competencia de una lección como la media aritmética de los niveles de competencia de los conceptos que forman parte de dicha lección.

Aunque desde el punto de vista del perfil del estudiante la actualización del mismo es un proceso relevante, desde el punto de vista de las estrategias de recomendación dicho proceso es algo ajeno a las mismas. De ahí que no se trate en este trabajo.

4.3. Aplicación de la estrategia de recomendación reactiva

A continuación vamos a mostrar la aplicación de la estrategia de recomendación reactiva descrita en el capítulo anterior a nuestro repositorio de LO de Programación. Como ya se mencionó, diferentes valores del parámetro α empleado en la medida de calidad (1) (ver Figura 26) provocan que el recomendador proporcione mayor o menor nivel de personalización. Para mostrar este hecho hemos desarrollado un prototipo que realiza distintas recuperaciones para diferentes valores del parámetro α. Más adelante veremos los diferentes resultados obtenidos mediante variaciones de este parámetro.

Recordemos que esta estrategia, al ser reactiva, opera bajo demanda del estudiante, el cual ha de formular una consulta en la que indicará qué conceptos quiere estudiar en la actual sesión de aprendizaje. Estos conceptos representarán sus objetivos a corto plazo, que junto con los objetivos a largo plazo representados en su perfil ayudarán a la estrategia a seleccionar aquellos LOs que mejor se adapten al estudiante.

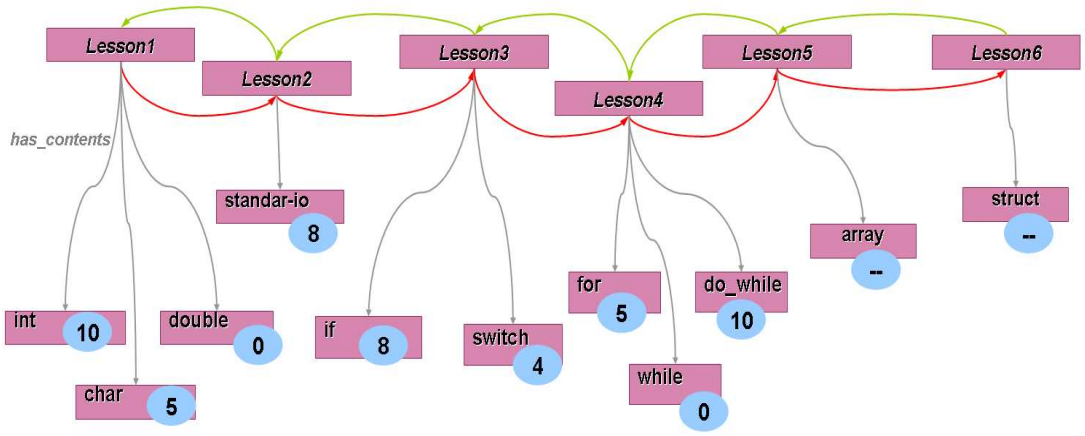


Figura 27. Representación gráfica del perfil de un estudiante. Aunque las lecciones no forman parte del perfil, se han incluido aquí para presentar el itinerario de aprendizaje definido

El perfil del estudiante que utilizaremos en el ejemplo de aplicación está representado gráficamente en la Figura 27. En esta figura vemos algunos de los conceptos que figuran en el perfil junto con el nivel de competencia alcanzado en cada uno de ellos. Los conceptos que no tienen nivel de competencia (*array* y *struct*) todavía no han sido explorados por el estudiante. El umbral de progreso considerado para seguir avanzando en el itinerario de aprendizaje ha sido establecido en 5. En la Tabla 2 tenemos algunos LOs contenidos en el repositorio junto con los conceptos que cubre cada uno de ellos.

Tabla 2. Algunos LOs contenidos en el repositorio junto con los conceptos que cubre cada uno de ellos.

LOs	Conceptos
L01	if - for
L02	if - while - struct
L03	switch - for
L04	switch -while - array
L05	if - while
L06	if - for
L07	if - for - while
L08	if - for - do_while
L09	if - for - array
L10	if - for - switch

La interfaz de este pequeño prototipo la podemos ver en la Figura 28. En la parte izquierda aparece un árbol que nos permite navegar por la taxonomía de conceptos. Mediante el mismo se puede seleccionar lo que se desea estudiar en la sesión actual. En la parte superior derecha se muestran los conceptos que se van formando la consulta. en la parte inferior se muestran los resultados del proceso de recomendación. Este prototipo ha sido utilizado para realizar pruebas, de ahí que podamos escoger el valor del parámetro α usado para obtener el conjunto propuesto. En función de este valor nos mostrará los resultados obtenidos.

De acuerdo a la figura, vemos que se propone la consulta $Q = \{if, for\}$.

El conjunto final (C_{Final}) que se recomienda a partir de Q y con $\alpha = 1$ es:

C_{Final} = {L06, L01, L10, L09}

Ese valor de α nos indica que la métrica de calidad es igual que la similitud entre el LO y la consulta. Vemos que C_{Final} contiene L01 y L06. Estos dos LOs tienen exactamente los mismos conceptos solicitados en la consulta, por ello son presentados en primer lugar (véase Figura 28). Luego aparecen L10 y L09 que cubren los dos conceptos de la consulta y uno más. De esta manera, las primeras recomendaciones se diferencian poco unas de otras. Esto es un ejemplo del tema de la sobreespecialización o la falta de diversidad de las recomendaciones citado en el Capítulo 2.

Si $\alpha = 0.5$, la similitud con la consulta y la utilidad pedagógica tienen el mismo peso en (1) (véase Figura 29). En este caso el conjunto final es:

C_{Final} = {L07, L01, L05, L06}

Los LOs que están altamente relacionados con la consulta (L01 y L06) siguen siendo propuestos. Sin embargo, son reordenados de tal manera que la prioridad es para L07 que, al mismo tiempo, cubre conceptos en los que el estudiante ha demostrado tener un bajo nivel de competencia (los conceptos *while* y *for*). Podemos observar que L05, que también cubre conceptos con un bajo nivel de competencia (*while*), es recomendado. En este caso vemos que no se ha comprometido de una manera significativa el interés que los LOs tienen para el estudiante a corto plazo.

Si $\alpha = 0.25$, la utilidad pedagógica tiene un mayor peso que la similitud con la consulta. En este caso, el recomendador proporcionar LOs que promuevan completar las carencias de conocimiento del estudiante incluyendo conceptos que es necesario reforzar. Los resultados los tenemos en la Figura 30. En este caso, el conjunto de LOs obtenido es el siguiente:

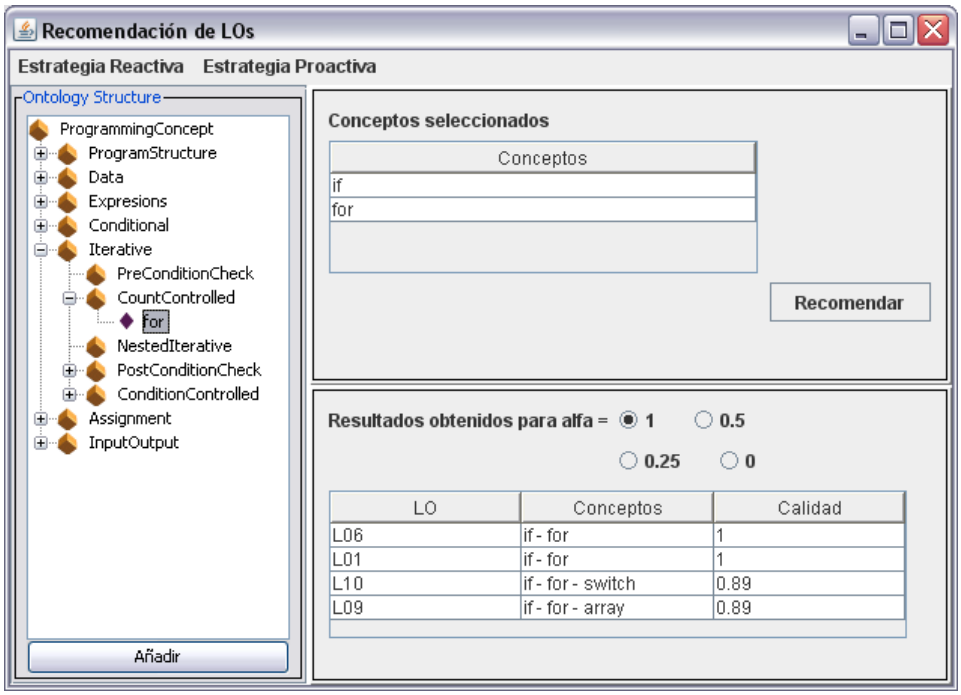


Figura 28. Interfaz de la estrategia de recomendación reactiva en el prototipo de prueba. Se muestran los resultados para la consulta propuesta y α = 1.

C_{Final} = {L07, L04, L05, L03}

En el ejemplo, tenemos que el concepto *switch* tiene un nivel de competencia igual a 4; L03 y L04 son LOs que cubren ese concepto. Por otro lado, el concepto *while* tiene un nivel de competencia igual a 0; L04, L05 y L07 cubren esta carencia. Con este valor de α los LOs recomendados atienden mejor las necesidades de refuerzo que en el caso anterior. El primer LO propuesto sigue satisfaciendo los objetivos a corto plazo representados en la consulta.

Por último, si consideramos el valor α = 0 sólo la utilidad pedagógica es tenida en cuenta. En este caso el conjunto propuesto es (véase Figura 31):

C_{Final} = {L04, L05, L07, L03}

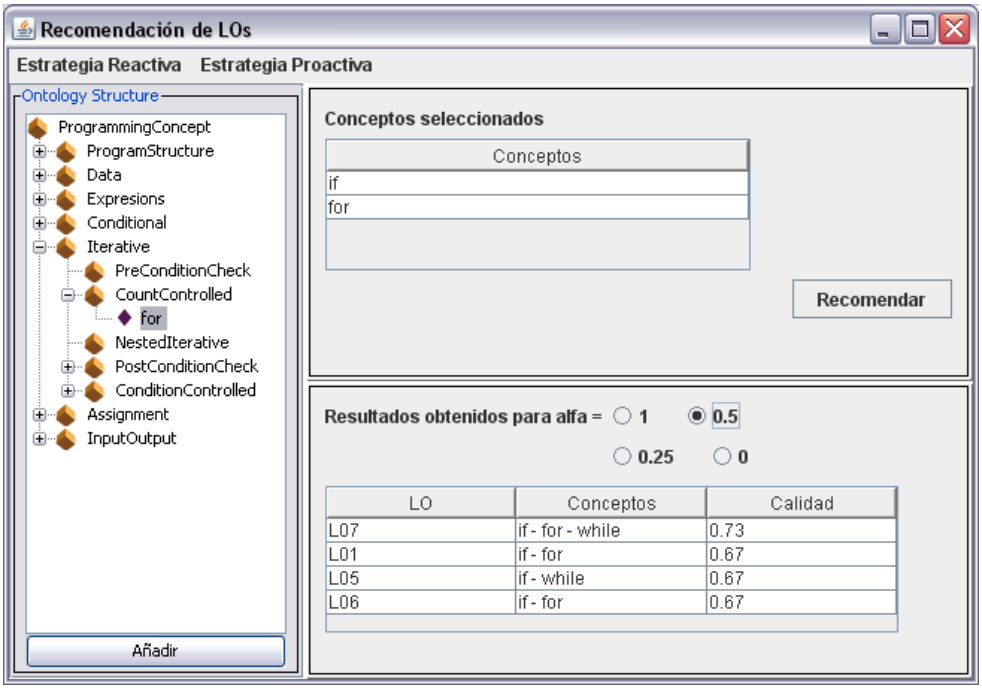


Figura 29. Resultados de la estrategia de recomendación reactiva con α = 0,5.

Los resultados son parecidos a los obtenidos con α = 0.25, pero el conjunto está ordenado de diferente manera. En este caso el LO que ha obtenido mejor calidad es L04. Esto es debido a que los conceptos cubiertos por este LO (*switch*, *while*) son los que más bajo nivel de competencia reflejan en el perfil, además cubre un concepto nuevo para el estudiante (*array*), y por todas estas características es el LO que tiene mayor utilidad pedagógica. Quizá este LO es el más diferente con la consulta entre todos los recuperados, ya que no cubre ninguno de los dos conceptos solicitados sino que cubre *hermanos* de estos. Este LO fue recuperado porque en la etapa de recuperación utilizamos una recuperación aproximada.

Lo mostrado aquí es un estudio mínimo inicial de las situaciones que se pueden producir variando el valor de *α*. Es conveniente realizar más pruebas con distintas partes de la taxonomía y distintos perfiles de estudiante para intentar localizar aquellos valores de *α* que puedan dar lugar a una mejor configuración de la recomendación.

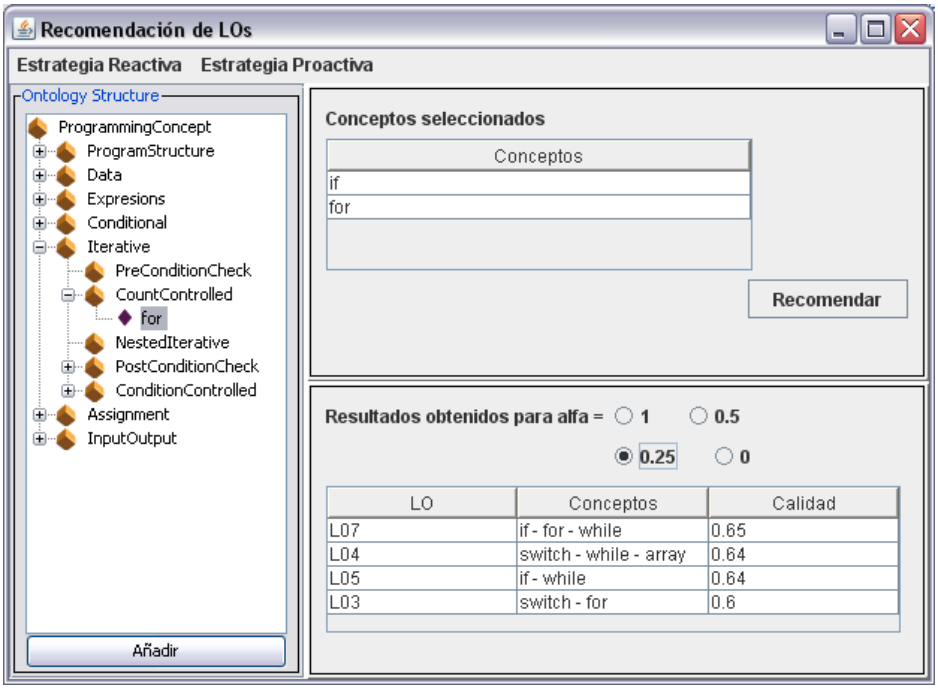


Figura 30. Resultados de la estrategia de recomendación reactiva con $\alpha = 0{,}25$.

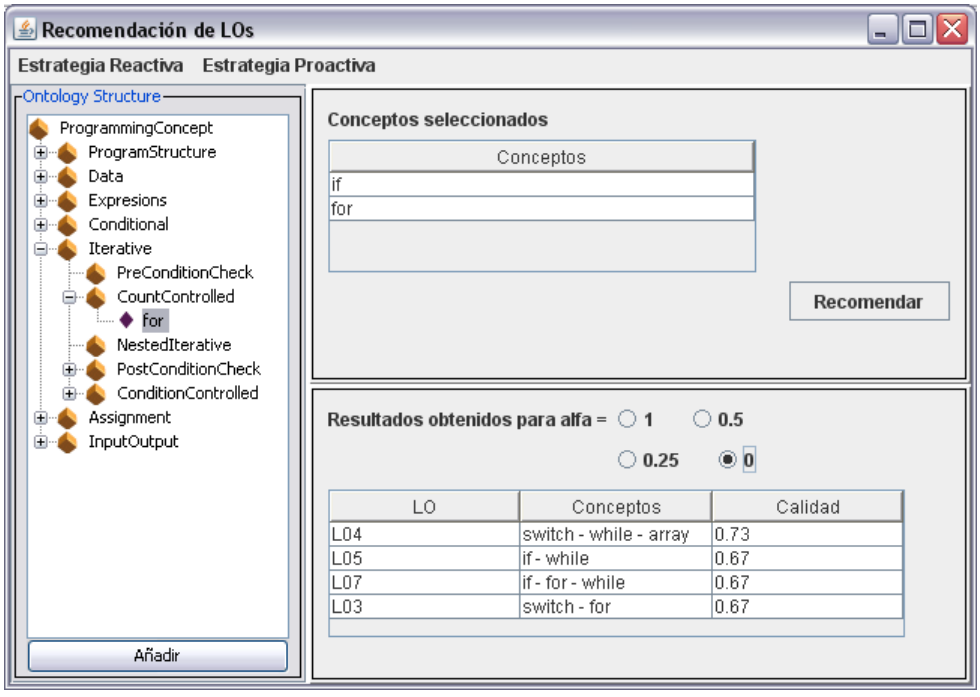


Figura 31. Resultados de la estrategia de recomendación reactiva con $\alpha = 0$

4.4. Aplicación de la estrategia de recomendación proactiva

En esta sección vamos a mostrar la aplicación de la estrategia de recomendación proactiva descrita en el capítulo anterior a nuestro repositorio de LOs de Programación. Recordemos que esta estrategia propone un conjunto inicial de LOs al estudiante sin necesidad de que éste proponga una consulta. Como ya se mencionó esta estrategia promueve la diversidad del conjunto recuperado. Para mostrar cómo opera esta estrategia hemos desarrollado un pequeño prototipo que realiza una recuperación inicial a partir de la información contenida en el perfil del estudiante, otorgando la posibilidad de refinar dicho conjunto.

El perfil de estudiante que utilizaremos para el ejemplo está representado gráficamente en la Figura 32. Igual que en el ejemplo genérico descrito en el capítulo anterior, el criterio considerado para seleccionar LOs representante(s) de cada grupo en las distintas fases de esta estrategia es que cubran el mayor número de conceptos posible. En la Tabla 3 tenemos algunos de estos LOs contenidos en el repositorio.

Tabla 3. Algunos LOs contenidos en el repositorio junto con los conceptos que cubre cada uno de ellos.

LOs	Conceptos
L20	int – char – switch – while – do_while – standard-io
L23	int – char – switch – for - array
L24	double – switch – while - array
L25	double – if – while – struct
L26	double – char – standard-io – if – switch - while
L27	int – char - if – do_while
L28	int –if – array
L29	if - for – char
L30	if - for – double
L31	char – switch - while
L32	int – if – for

En la Figura 33 aparecen las *"Actividades del día"* (i.e. la primera recomendación propuesta por la estrategia). Para este ejemplo asumimos que siempre que sea posible se mostrarán 4 LOs. Para elegir los representantes de cada grupo en cada una de las fases se sigue el criterio de elegir entre los que cubren más conceptos.

Para poder hacer la primera propuesta, lo primero que tiene que conocer la estrategia es si este estudiante puede descubrir nuevos conceptos. Según el perfil mostrado vemos que ha alcanzado a explorar hasta *Lesson4* y que el nivel de competencia alcanzado en esta lección es 6 (la media aritmética de los conceptos que forman parte de ella). Por lo tanto este estudiante está listo para descubrir nuevos conceptos contenidos en la

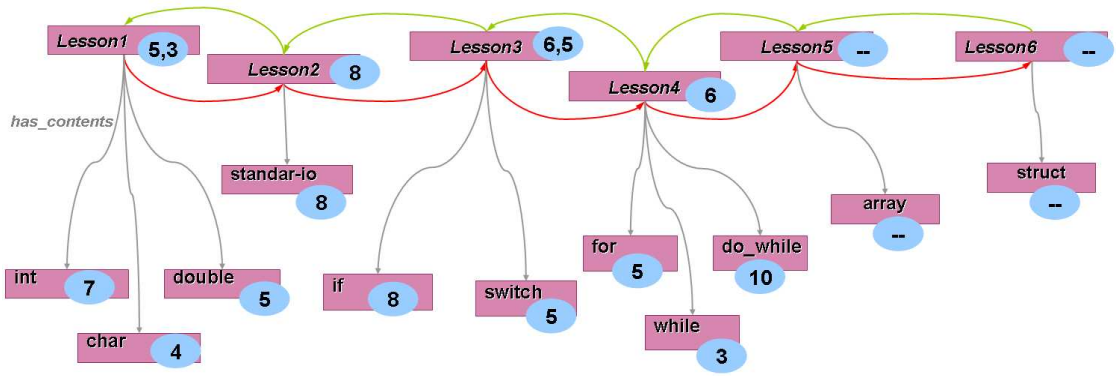


Figura 32. Representación gráfica del perfil de un estudiante. Aunque las lecciones no forman parte del perfil, se han incluido aquí para presentar el itinerario de aprendizaje definido.

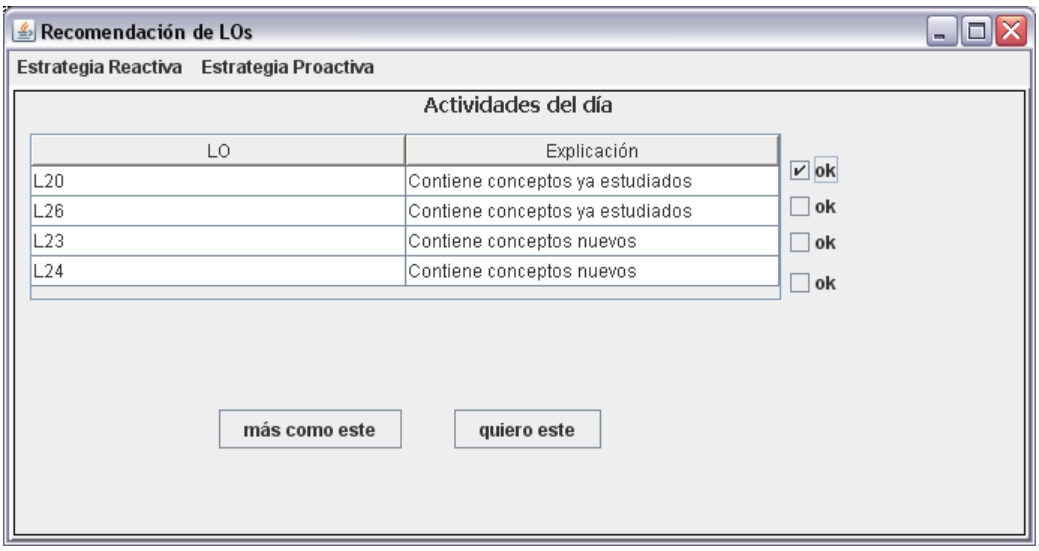


Figura 33. Interfaz de la estrategia de recomendación proactiva. Vemos cada LO junto con la explicación de por qué fue seleccionado

siguiente lección. En particular podrá descubrir los conceptos de *Lesson5*. Con esta información la estrategia divide el repositorio de LOs en dos grupos: refuerzo y descubrimiento. La recomendación resultante son 4 LOs de los cuales dos nos indican que contiene conceptos ya estudiados y los otros dos conceptos nuevos.

Después de esta primera propuesta el estudiante tiene dos opciones: seleccionar un LO para practicar (*quiero este)*, o refinar la recomendación propuesta indicando sus

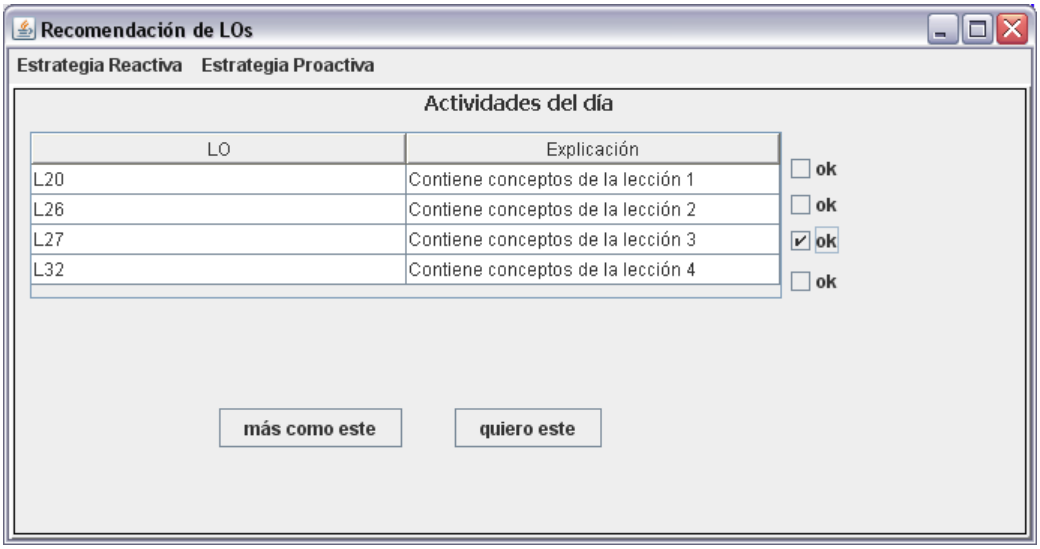


Figura 34. Conjunto de LOs propuestos a partir de un refinamiento.

preferencias con respecto al conjunto propuesto comenzando así un proceso conversacional *(más como este)*. Como ya indicamos, es conveniente proporcionar al menos una sencilla explicación sobre por qué ha sido seleccionado cada LO por parte de la estrategia.

Si se decide refinar la propuesta indicando que se desean más como L20, a este nivel se estaría indicando que se quieren más LOs que contengan conceptos ya estudiados. En la siguiente fase la estrategia definiría 4 grupos, que se corresponden con las lecciones ya estudiadas (*Lesson1, Lesson2, Lesson3 y Lesson4)*. Como resultado obtenemos los cuatros LOs de la Figura 34. Observamos que en esta ocasión las explicaciones proporcionadas por la estrategia han cambiado. Ahora indica la lección a la que pertenece cada LO, ya que éste ha sido el criterio aplicado para proporcionar diversidad a la recomendación. De nuevo estamos ante la situación de decidir entre refinar la consulta o seleccionar un LO para practicar. Imaginemos que se sigue refinando eligiendo L27, lo que está indicando que se quieren otros LOs que contengan conceptos sobre *Lesson3*. En particular esta lección engloba todos los conceptos de estructuras condicionales.

La nueva propuesta estará formada por LOs de los 2 grupos en los que se divide el repositorio (*SimpleConditional y MultipleConditional*, véase Figura 23)[6]. El resultado de la nueva propuesta lo vemos en la Figura 35.

[6] Realmente en ese paso estaríamos en el nivel de la taxonomía correspondiente a *Conditional*, pero al ser un único concepto y un único grupo el que se consideraría se ha tomado el criterio de seguir bajando hasta encontrar un nivel en el que se puedan considerar varios grupos (o hasta llegar a un nodo hoja).

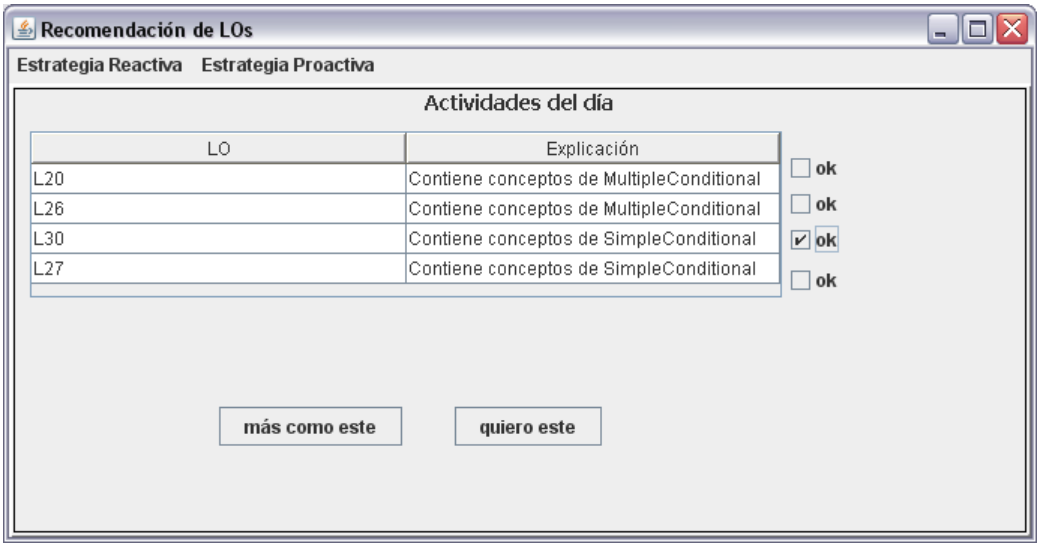


Figura 35. Conjunto de LOs propuestos a partir de un refinamiento

Supongamos que se decide refinar la consulta indicando que se desean más como L30. En ese caso, estamos en una hoja de la taxonomía. Se entra así en la tercera fase de la estrategia. Los LOs que cubren el concepto *SimpleConditional* se dividen en seis grupos, correspondientes a cada una de las categorías de la versión revisada de la Taxonomía de Bloom. El conjunto propuesto está formado por los LOs de la Figura 36. Todos ellos cubrirán el concepto *SimpleConditional* (representado por su instancia *if*). El primero de ellos, L26, pertenece a la categoría *Crear*; es un LO que permite al estudiante diseñar el resultado de un problema. Corresponde a un *Ejercicio de desarrollo completo*. L32 pertenece a la categoría *Aplicar*; es un LO que pide que el estudiante utilice sus conocimientos para determinar/resolver/calcular una actividad propuesta. Se trata de un *Ejercicio de "encuentra el error"*. L29 pertenece a la categoría *Comprender*[7]; es un LO que explica algún contenido. Se trata de una *Presentación* de contenidos teóricos. Por último L30 pertenece a la categoría *Analizar*; es un LO que permite interpretar lo expuesto en él. Se trata de un *Ejemplo de código*. En este momento se ha llegado al final de la estrategia proactiva.

[7] En la Sección 3.5 explicamos que en la última fase si nos encontrábamos reforzando conceptos eligiríamos LOs pertenecientes a las categorías más altas de la versión revisada de la taxonomía de Bloom. En este caso se ha elegido este LO porque no había ningún otro LO que permitiese cubrir ninguna de las 4 categorías más altas.

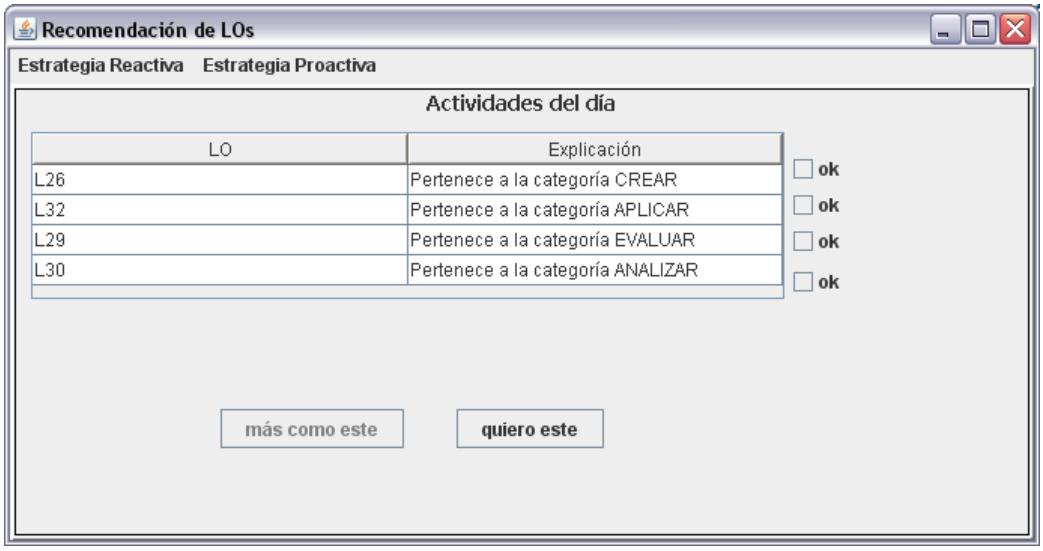


Figura 36. Conjunto de LOs propuestos junto con la categoría de la versión revisada de la Taxonomía de Bloom a la que pertenecen.

4.5. Conclusiones

En este capítulo se ha descrito la aplicación de las dos estrategias de recomendación basadas en contenido a un repositorio de contenidos de Programación.

El primer paso realizado para la aplicación de las dos estrategias fue adaptar las fuentes de conocimiento:

- Diseñar una ontología en OWL que contuviera todo el conocimiento del dominio necesario para las estrategias de recomendación.
- Adaptar los recursos educativos contenidos en el repositorio existente al estándar Learning Object Metadata (LOM) y desarrollar un programa en Java que permitiese conectar los LOs con la ontología definida. Para ello se ha hecho uso de OntoBridge.

Con el fin de ejemplificar las dos estrategias se han desarrollado unos pequeños prototipos en Java. Estos prototipos, si bien aún deben ser refinados, nos han permitido enfrentarnos a la implementación concreta de las estrategias genéricas presentadas en el Capítulo 3.

La estrategia reactiva presentada en la Sección 4.3 [66, 67] permite incorporar distintos niveles de personalización en el recomendador. Valores altos del parámetro α

en la medida de la calidad de un LO otorgan la prioridad a aquellos LOs que son más similares a la consulta del estudiante. Estos valores podrían ser apropiados para estudiantes que han obtenidos buenos resultados en la parte del itinerario de aprendizaje que llevan recorrido. Por el contrario, valores bajos de α otorgan la prioridad a LOs que muestran valores altos de *UP*. Estos valores podrían ser adecuados en las primeras etapas del itinerario de aprendizaje (en la exploración de los primeros conceptos) o cuando un estudiante ha obtenidos bajos resultados en el itinerario que lleva recorrido. Es necesario adoptar un compromiso en el peso que se dé a la similitud y a la utilidad pedagógica a la hora de calcular la similitud, es decir, en la elección del valor del parámetro α. Aquí se ha mostrado un estudio mínimo inicial de las situaciones que se pueden producir variando el valor de α. Es conveniente realizar más pruebas con distintas partes de la taxonomía y distintos perfiles de estudiante para intentar localizar aquellos valores de α que puedan dar lugar a una mejor configuración de la recomendación.

En cuanto a la estrategia proactiva presentada en la Sección 0 [65], ésta promueve la diversidad. Se ha realizado el ensayo de esta estrategia utilizando el número de conceptos cubiertos como criterio para elegir los LOs representantes de cada grupo dentro de cada iteración. Sería conveniente ensayar algunas de las otras alternativas propuestas en el Capítulo 3 y ver cuál puede conducir a situaciones de mayor diversidad.

Capítulo 5

CONCLUSIONES Y TRABAJO FUTURO

Como indicábamos al comienzo de la presente memoria, en los últimos años ha crecido la cantidad de recursos disponibles para los estudiantes en repositorios de contenidos educativos. La cantidad de estos recursos a los que puede acceder un estudiante provoca, en caso de no disponer del soporte necesario, que éste sufra una sobrecarga de información dificultando así el proceso de toma de decisiones sobre qué contenido es el más adecuado para él. Para aliviar este problema se hace necesario el desarrollo de herramientas que faciliten la localización de aquellos recursos que sean de interés para el estudiante y que se adapten al conocimiento individual, a los objetivos y/o a las preferencias del mismo.

En este Proyecto Fin de Máster se ha estudiado el uso de los sistemas de recomendación como medio para el acceso personalizado a los repositorios de contenidos educativos. Este estudio ha dado lugar a la propuesta de dos estrategias de recomendación aplicables en este tipo de repositorios. Se enmarca así este proyecto dentro de una reciente línea de trabajo dentro del contexto investigador que aborda el traslado de técnicas de recomendación al contexto educativo. Este trabajo es una continuación natural de los trabajos que dentro de esta nueva corriente se están realizando en el seno del Grupo de Aplicaciones de Inteligencia Artificial (GAIA) de la Universidad Complutense de Madrid.

De manera más detallada, las principales aportaciones del trabajo realizado, vinculadas a los objetivos planteados, han sido las siguientes:

- Se ha presentado una revisión del estado actual en la investigación sobre los sistemas recomendadores. Esta revisión nos ha permitido establecer cuáles son las principales características de estos sistemas, sus beneficios y algunos de los principales problemas que nos podemos encontrar en ellos. El resultado ha sido un marco teórico de caracterización de los sistemas recomendadores.
- Se ha analizado detenidamente el trabajo previo propuesto en [31, 32], dentro del campo del traslado de técnicas de recomendación al contexto educativo. El resultado de dicho análisis ha sido la detección de líneas de mejora del citado trabajo. En particular se consideraron líneas interesantes de expansión las tres siguientes, cada una de ellas relativa a un aspecto de diseño recogido en el marco de caracterización de recomendadores mencionado anteriormente:
    - Inclusión de un mayor nivel de personalización.
    - Planteamiento de una estrategia de interacción usuario-recomendador proactiva, como complemento a la naturaleza reactiva de la existente.
    - Diseño de una estrategia de selección que afronte el problema de la sobreespecialización.
- Se ha propuesto una estrategia de recomendación basada en contenido reactiva que pretende aliviar los problemas de *débil* personalización encontrados en [31, 32]. Esta estrategia permite incorporar distintos niveles de personalización mejorando así la capacidad de adaptación al estudiante [66, 67] Esta adaptación se ha realizado explorando un modelo de personalización *fuerte* que incorpora la consecución de objetivos a corto y a largo plazo.
- Se ha propuesto una segunda estrategia de recomendación basada en contenido de naturaleza proactiva [65], para aliviar las dificultades de acceso con las que se encuentran los alumnos con bajos conocimientos o con una actitud pasiva en estrategias como la descrita en [31, 32]. Se emplea la navegación por propuesta como modelo de interacción con el estudiante, Esta estrategia también promueve la inclusión de diversidad en la recomendación, afrontando así el inconveniente de la sobreespecialización.
- Se han identificado las necesidades de conocimiento específicas para estas dos de estrategias, a saber: una ontología del dominio, los LOs o recursos educativos (indexados a través de los contenidos de la ontología), y el perfil del estudiante.
- Se ha iniciado el proceso de aplicación de ambas estrategias a un repositorio de LOs ligados a la enseñanza de la Programación. Esta aplicación nos ha conducido a:
    - Desarrollar una ontología, en OWL, que contiene conceptos del dominio e itinerarios de aprendizaje sobre los mismos.

- Adaptar al estándar Learning Object Metadata (LOM) los recursos educativos de Programación contenidos en un repositorio existente.

A nivel de difusión de resultados, el presente trabajo ha dado lugar a las publicaciones que a continuación se enumeran:

- Ruiz-Iniesta, A., Jiménez-Díaz, G., Gómez-Albarrán, M.: Recommendation in Repositories of Learning Objects: A Proactive Approach that Exploits Diversity and Navigation-by-Proposing. In: 9th IEEE International Conference on Advanced Learning Technologies, pp. 543-545, (2009)
- Ruiz-Iniesta, A., Jiménez-Díaz, G., Gómez-Albarrán, M.: User-adaptive Recommendation Techniques in Repositories of Learning Objects: Combining Long-term and Short-term Learning Goals. In: Fourth European Conference on Technology Enhanced Learning, LNCS 5794, pp. 645-650, (2009)
- Ruiz-Iniesta, A., Jiménez-Díaz, G., Gómez-Albarrán, M.: Promoting Strong Personalization in Content-based Recommendation Systems of Learning Objects. In: XI International Symposium on Computers in Education (2009) (en prensa)

5.1. Trabajo futuro

A la vista de los resultados obtenidos queda mucho trabajo por hacer. A continuación señalamos algunas tareas como posible trabajo futuro.

En relación con la estrategia reactiva es conveniente realizar más pruebas con distintas partes de la taxonomía y distintos perfiles de estudiante para intentar localizar aquellos valores del parámetro usado en la medida de calidad definida que puedan dar lugar a una mejor configuración de la recomendación. Así mismo convendría experimentar con distintas medidas de cálculo de la similitud con la consulta y de valoración de la utilidad pedagógica de los LOs.

En cuanto a la estrategia proactiva, se está ensayando la misma utilizando el número de conceptos cubiertos como criterio para elegir los LOs representantes de cada grupo dentro de cada iteración del proceso conversacional. Sería conveniente ensayar algunas de las otras alternativas propuestas en el Capítulo 3 y analizar cuál puede conducir a situaciones que mejoren más la diversidad. Por ejemplo, el uso de un criterio que tuviese en cuenta, al igual que en la estrategia reactiva, la utilidad pedagógica de los LOs para un estudiante dado.

Una mejora en cuanto a la estrategia proactiva sería la incorporación de una realimentación más detallada por parte por el estudiante. En lugar de que en cada refinamiento sólo seleccione un LO indicando que quiere "*más como este"*, que además indiqué *por qué prefiere ese LO*. Estas explicaciones podrían estar basadas en el resto de características que definen un LO según LOM. En el modelado de los LOs existe

más información disponible a parte de la que a día de hoy se utiliza. Por ejemplo existen categorías que incluyen información sobre el tipo de recurso que es, sobre su nivel de dificultad, etc. También se podría sacar mayor partido a la clasificación que se hace con respecto a la Taxonomía de Bloom. Si permitimos que el estudiante explique sus preferencias con respecto a un LO en base a estas informaciones se obtendría un perfil de preferencias. Este perfil sería una fuente de conocimiento adicional para la estrategia y así podrían tenerse en cuenta cosas como "prefiere ejercicios fáciles", o "ejemplos resueltos", o "le gustan los contenidos clasificados en una cierta categoría de Bloom", en el criterio de selección de los LOs propuestos. Estas preferencias también podrían ser útiles de cara a la estrategia reactiva a la hora de calcular la utilidad pedagógica de los LOs recuperados.

En cuanto al perfil del estudiante, actualmente no utilizamos la información sobre el historial de navegación que incluye. Un posible refinamiento de la métrica de calidad empleada en la estrategia reactiva podría ser tener en cuenta este hecho y penalizar estos LOs con el objetivo de que no vuelvan a ser recomendados al estudiante. Esta información también puede emplearse en la estrategia proactiva.

La integración de las estrategias basadas en contenido propuestas en la estrategia híbrida inicial sería otra tarea a realizar. Recordemos que esta estrategia tiene dos componentes: un módulo de recomendación basado en contenido y otro de CF. Estos dos componentes actúan en cascada, haciendo que las recomendaciones del primero sean refinadas por el segundo. Esta integración permitiría refinar el conjunto de LOs propuesto por cualquiera de estas dos estrategias mediante las valoraciones asignadas a cada LO por los estudiantes.

Por último, sería necesario evaluar la aplicación de estas estrategias en una comunidad real de estudiantes, como por ejemplo nuestros estudiantes de asignaturas de Programación, que puedan dar información sobre el interés de las recomendaciones realizadas.

Anexo 1

EJEMPLOS DE OBJETOS DE APRENDIZAJE

Este anexo recoge algunos ejemplos de recursos contenidos en el repositorio junto con el fichero XML que los modela. Todos los recursos contenidos en el repositorio están en formato HTML (*HyperText Markup Language*).

El primer LO que vamos a mostrar lo tenemos en la Figura 1. En la parte superior de la imagen tenemos parte del fichero XML que modela el LO según el estándar LOM y en la parte inferior tenemos el HTML con el enunciado del ejercicio. Es un *Ejercicio para resolver* de los de *desarrollo completo* que cubre los conceptos de Programación *if* y *while.* Este LO está clasificado dentro de la categoría *Crear* de la versión revisada de la taxonomía de Bloom.

En la Figura 2 tenemos otro ejemplo de LO. Al igual que antes, en la parte superior de la imagen está el modelo XML y en la parte inferior el enunciado en HTML. En este caso el LO es un *Ejemplo resuelto* que cubre el concepto de Programación *if*. Este LO pertenece a la categoría *Analizar* de la versión revisada de la taxonomía de Bloom.

Por último tenemos el LO de la Figura 3. Este LO es un *Ejercicio para resolver* de modalidad "cuestión". En particular, este LO cubre los conceptos de Programación *if* y *switch*. Pertenece a la categoría *Analizar* de la versión revisada de la taxonomía de Bloom.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOM">
    <general>
        <identifier>
                <string>P222MAGM03</string>
        </identifier>
        <title>
            <string>Cambio a base factorial</string>
        </title>
        <language><string>es</string></language>
        <description>
            <string>Desarrollar un subprograma en base a un algoritmo dado.
                El objetivo es realizar cálculos mediante bucles.
            </string>
        </description>
        <!-- Many other keywords snipped -->
        <keyword>
            <string>while</string>
        </keyword>
          <keyword>
            <string>if</string>
        </keyword>
        <keyword>
            <string>cpp</string>
        </keyword>
        <keyword>
            <string>pascal</string>
        </keyword>
        <coverage>
                <string>programación</string>
        </coverage>
        <structure>
                <source>LOMv1.0</source>
                <value>atomic</value>
        </structure>
        <aggregationLevel>1</aggregationLevel>
    </general>
    <lifeCycle>
        <version>
                <source>LOMv1.0</source>
                <value>1.0</value>
        </version>
        <status>
                <source>LOMv1.0</source>
                <value>final</value>
```

Cambio a base factorial

Título	**(P222MAGM03)** *Cambio a fase factorial*
Autor	*Marco Antonio Gómez Martín*
Paquete temático	*Bucles*
Variante	✓ (P222MAGM02) **Ejercicio base** *Cambio a base 2*
Nivel de dificultad	Medio

Enunciado

Los sistemas de numeración posicionales son aquellos donde el valor de un dígito cambia dependiendo de su posición.

El sistema en base 10 que utilizamos comunmente pertenece a este tipo de numeración. Como es sabido, el dígito "5" aporta más o menos cantidad al número dependiendo de su posición. En concreto, el valor del dígito más a la derecha es multiplicado por 10^0, el dígito siguiente es multiplicado por 10^1, el siguiente por 10^2, y así sucesivamente.

La *base factorial* es una notación posicional donde los dígitos son multiplicados por el *factorial* de la posición que ocupan (no consideraremos la "posición cero" por comodidad, ya que 0! = 1!).

De esta forma, el dígito más a la izquierda tiene peso 1!, el siguiente 2! y así sucesivamente.

Ejemplos:

Base factorial	Desarrollo	Base 10
320	3 * 3! + 2 * 2! + 0 * 1!	22
301	3 * 3! + 0 * 2! + 1 * 1!	19
43100	4 * 5! + 3 * 4! + 1 * 3! + 0 * 2! + 0 * 1!	558

Implementa un programa que solicite al usuario un número en *base 10* y lo escriba en *base factorial*.

Figura 1. Vista parcial del modelo XML (parte superior) para el LO mostrado (parte inferior).

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOM">
    <general>
        <identifier>
                <string>E7ML05</string>
        </identifier>
        <title>
            <string>Método de ordenación recursiva Quicksort</string>
        </title>
        <language><string>es</string></language>
        <description>
            <string>Desarrollar un programa a partir un algoritmo.
                        El objetivo es ordenar un vector mediante recursividad.
            </string>
        </description>
        <!-- Many other keywords snipped -->
       <keyword>
            <string>if</string>
        </keyword>
        <keyword>
            <string>recursión</string>
        </keyword>
        <keyword>
            <string>pascal</string>
        </keyword>
                <keyword>
            <string>cpp</string>
        </keyword>
        <coverage>
                <string>programación</string>
        </coverage>
        <structure>
                <source>LOMv1.0</source>
                <value>atomic</value>
        </structure>
        <aggregationLevel>1</aggregationLevel>
    </general>
    <lifeCycle>
        <version>
                <source>LOMv1.0</source>
                <value>1.0</value>
        </version>
```

Método de ordenación recursiva Quicksort

Paquete temático	Recursividad
Nivel de dificultad	Alto

Enunciado

Implementar el método de ordenación recursiva Quicksort.

Algoritmo

Pasos:

1. Elegir un elemento del vector como pivote.
2. Dividir el vector original en un subvector *v* con los elementos menores que el pivote y otro *w* con los elementos mayores.
3. Ordenar ambos subvectores utilizando quicksort.
4. Devolver el vector resultante de concatenar: *v*, *pivote*, *w*.

Algoritmo de división:

1. Elegir el elemento pivote (ej: el primero del vector).
2. Recorrer el vector desde la izquierda y la derecha a la vez.
    1. Buscar desde la derecha un elemento menor o igual que el pivote.
    2. Buscar desde la izquierda un elemento mayor que el pivote.
    3. Intercambiar ambos elementos.
3. Hasta que coincidan o se hayan cruzado ambas posiciones.
4. Intercambiar el pivote y el elemento en el que finalizó el recorrido desde la derecha

```
/*-------------------------------------------------------------*/
/* Procedimiento quicksort
   Descripcion: metodo de ordenacion de un vector de enteros por el quicksort
   Entradas:
        vector: el vector a ordenar
        limInf: el limite inferior del algoritmo
        limSup: el limite superior del algoritmo
*/
void quicksort(int *vector, int limInf, int limSup)
{
   int izda, k, dcha;
   int med, temp;

   izda = limInf;
   dcha = limSup;

   med = vector[(izda+dcha)/2];

   do
   {
        while (vector[izda] < med && izda < limSup ) izda++;
                while (med < vector[dcha] && dcha > limInf ) dcha--;
                if (izda <= dcha)
                {
                        temp = vector[izda];
                        vector[izda]= vector[dcha];
                        vector[dcha]= temp;
                        izda++;
                        dcha--;
                }
   } while (izda <= dcha);

   if (limInf < dcha) quicksort(vector,limInf,dcha);
   if (izda < limSup) quicksort(vector,izda,limSup);

}
```

Figura 2. Vista parcial del modelo XML (parte superior) para el LO mostrado (parte inferior).

```
<?xml version="1.0" encoding="utf-8" ?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOM">
    <general>
        <identifier>
                <string>P5CondicionalesSwitchCPP</string>
        </identifier>
        <title>
            <string>Pregunta de Autoevaluación Tema de Condicionales - Switch</string>
        </title>
        <language><string>es</string></language>
        <description>
            <string>Pregunta de respuesta corta.
                El objetivo es analizar un trozo de código y escribir la respuesta correcta.
            </string>
        </description>
        <!-- Many other keywords snipped -->

          <keyword>
            <string>if-sentence</string>
        </keyword>

          <keyword>
            <string>switch-sentence</string>
        </keyword>

          <keyword>
            <string>condicionales</string>
        </keyword>
        <keyword>
            <string>cpp</string>
        </keyword>
        <coverage>
                <string>programación</string>
        </coverage>
        <structure>
                <source>LOMv1.0</source>
                <value>atomic</value>
        </structure>
        <aggregationLevel>1</aggregationLevel>
    </general>
    <lifeCycle>
        <version>
                <source>LOMv1.0</source>
```

PREGUNTA Nº 5

Tipo de pregunta: RESPUESTA CORTA

Enunciado

Indica el valor almacenado por el siguiente programa en la variable *cifrado* cuando el usuario introduce en letra el valor A:

```
char letra, cifrado;

cin>>letra;

if (UpCase(letra)>='A'&& UpCase(letra)<='Z'){

   switch (UpCase(letra)){

    case 'Z': case 'A':

        cifrado='A';

     default:

        cifrado++;

   }

}
```

Figura 3. Vista parcial del modelo XML (parte superior) para el LO mostrado (parte inferior).

Referencias

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, 7 (1), pp. 39-59, (1994)

2. Abel, A., Lenne, D., Moulin, C., Benayache, A.: Using Two Ontologies to Index e-Learning Resources. In: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 549- 552, (2004)

3. Balabanovic, M.: An adaptive Web page recommendation service. In: Proceedings of the 1st International Conference on Autonomous Agents, pp 378-385 (1997)

4. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, pp. 714-720, (1998)

5. Bernaras, A., Laresgoiti, I., Corera, J.: Building and Reusing Ontologies for Electrical Network Applications. In: Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 96), pp. 298-302, (1996)

6. Bilgic, M., Mooney, R.: Explaining Recommendations: Satisfaction vs. Promotion. In: Beyond Personalization: A Workshop on the Next Stage of Recommender Systems Research at the International Conference on Intelligent User Interfaces, pp. 13-18, (2005)

7. Billsus, D., Pazzani, M.: A personal news agent that talks, learns and explains. In: Proceedings of the Third International Conference on Autonomous Agents (Agents'99), pp. 268-275, (1999)

8. Billsus, D., Pazzani, M.J.: A Hybrid User Model for News Story Classification. In: Proceedings of the 7th International Conference on User Modeling, pp. 99-108, (1999)

9. Billsus, D., Pazzani, M.J., Chen, J.: A Learning Agent for Wireless News Access. In: Proceedings of the 5th International Conference on Intelligent User Interfaces, pp. 33-36, (2000)

10. Bobrow, D., Kaplan, R., Kay, M., Norman, D., Thompson, H., Winograd, T.: Gus, a frame driven dialog system. Artificial Intelligence, 8, pp. 155-173, (1997)

11. Bradley, K., Rafter, R., Smyth, B.: Case-Based User Profiling for Content Personalisation. In: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 62-72, (2000)

12. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43-52, (1998)

13. Bridge, D.: Towards Conversational Recommender Systems: A Dialogue Grammar Approach. In: Proceedings of the Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the 6th European Conference in Case-Based Reasoning, pp. 9-22, (2002)

14. Bridge, D., Ouml, Ker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. The Knowledge Engineering Review, 20 (03), pp. 315-320, (2005)

15. Brusilovsky, P.: WebEx: Learning from Examples in a Programming Course. In: Proceedings of the World Conf. of the WWW and Internet, pp. 124-129, (2001)

16. Brusilovsky, P., Farzan, R., Ahn, J.-w.: Comprehensive Personalized Information Access in an Educational Digital Library. In: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, pp. 9-18, (2005)

17. Burke, R.: Hybrid Web Recommender Systems. In: Heidelberg, S.B. (ed.): The Adaptive Web, Vol. 4321, pp. 377-408, (2007)

18. Burke, R., Hammond, K., Young, B.: Knowledge-based navigation of complex information spaces. In: Proceedings of the 13th National Conference on Artificial Intelligence., pp. 462-468, (1996)

19. Burke, R.D.: Knowledge-based Recommender Systems. Encyclopedia of Library and Information Systems, 69, Sup. 32, (2000)

20. Burke, R.D., Hammond, K.J., Yound, B.C.: The FindMe approach to assisted browsing. IEEE Expert, 12 (4), pp. 32-40, (1997)

21. Coyle, L., Cunningham, P., Hayes, C.: A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers. In: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, pp. 505-518, (2002)

22. Doyle, M., Cunningham, P.: A dynamic approach to reducing dialog in on-line decision guides. In: Proceedings of the 5th European Workshop on Case-Based Reasoning, pp. 49-60, (2000)

23. Drachsler, H., Hummel, H., Koper, R.: Recommendations for learners are different: Applying memory-based recommender system techniques to lifelong learning. In: Proceedings of the 1st Workshop on Social Information Retrieval for Technology-Enhanced Learning & Exchange, pp. 18-26, (2007)

24. Drachsler, H., Hummel, H.G.K., Koper, R.: Personal Recommender Systems for Learners in Lifelong Learning Networks; the requirements, techniques and model. International Journal of Learning Technology, 3(4) , pp. 404-423, (2008)

25. Faries, J.M., Reiser, B.J.: Access and use of previous solutions in a problem solving situation. In: Proceedings of 10th Annual Conference of the Cognitive Science Society, pp. 433-439, (1988)

26. Farzan, R., Brusilovsky, P.: Social navigation support in a course recommender system. In: Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, pp. 91-100, (2006)

27. García-Salcines, E., Romero-Morales, C., Ventura-Soto, S., Castro-Lozano, C.: Sistema recomendador colaborativo usando minería de datos distribuida para la mejora continua de cursos e-learning. IEEE-RITA, 3 (1), pp. 19-30, (2008)

28. Gascueña, J.M., Fernandez-Caballero, A., Gonzalez, P.: Domain Ontology for Personalized E-Learning in Educational Systems. In: Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, pp. 456-458, (2006)

29. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. 35 (12), pp. 61-70, (1992)

30. Gomes, P., Antunes, B., Rodrigues, L., Santos, A., Barbeira, J.: Using Ontologies for eLearning Personalization. 3rd ELearning Conference - Computer Science Education, (2006)

31. Gómez-Albarrán, M., Bautista-Blasco, S., Carrillo-de-Albornoz, J.: Personalized Access and Students' Coauthoring in Repositories of Learning Objects: The Case of a Repository of Programming Examples. In: Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies, pp. 693-695, (2008)

32. Gómez-Albarrán, M., Jiménez-Díaz, G.: Recommendation and Students' Authoring in Repositories of Learning Objects: A Case-Based Reasoning Approach. International Journal of Emerging Technologies in Learning, 4, pp. 35-40, (2009)

33. Gómez-Albarrán, M., Jiménez-Díaz, G., López-Fernández, M., Gómez-Martín, M.A., Díaz-Esteban, A., Hernández-Yáñez, L., Ruiz-Iniesta, A.: Evolución de un espacio de trabajo multidisciplinar para el aprendizaje de la programación basado en casos prácticos: de los repositorios a los cursos adaptativos en el Campus Virtual de la UCM. In: V Jornada del Campus Virtual - UCM (2009)

34. Gómez-Albarrán, M., Jiménez-Díaz, G., López-Fernández, M., Gómez-Martín, M.A., Díaz-Esteban, A., Hernández-Yáñez, L.A., Ruiz-Iniesta, A.: Example-supported learning of programming concepts: from free-access to knowledge-controlled routing in repositories deployed in a Virtual Campus. In: XI International Symposium on Computers in Education (2009) (en prensa)

35. González-Calero, P., Díaz-Agudo, B., Gómez-Albarrán, M.: Applying DLs for Retrieval in Case-Based Reasoning. In: Proceedings of the International Workshop on Description Logics, pp. 51-55, (1999)

36. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition, 5 (2), pp. 199-220, (1993)

37. Han, E.-H., Karypis, G.: Feature-based recommendation system. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 446-452, (2005)

38. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, pp. 241-250, (2000)

39. Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 194-201, (1995)

40. Holt, P., Dubs, Jones, M., and Greer, J.: The State of Student Modelling: Student Modelling. In: J.E. Greer & G.I. McCalla (Eds.), Student modelling: The Key to Individualized Knowledge-based Instruction, pp. 3-38, (1994)

41. Hummel, H., Van den Berg, B., Berlanga, A., Drachsler, H., Jansenn, J., Nadolski, R., Koper, R.: Combining social-based and information-based approaches for personalised recommendation on sequencing learning activities. International Journal of Learning Technology, 3 (2), pp. 152-168, (2007)

42. Draft Standard for Learning Object Metadata, IEEE 1484.12.1-2002, 15 July 2002

43. Draft Standard for Learning Technology— Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata., IEEE P1484.12.3, Draft 8, 2005

44. Kamal, A., Wijnand van, S.: TiVo: making show recommendations using a distributed collaborative filtering architecture. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 394-401, (2004)

45. Kolb, D.A.: Experiential Learning: Experience as the Source of Learning and Development. Prentice-Hall. (1984)

46. Krathwohl, D.R.: A revision of bloom's taxonomy: An overview. Theory into Practice, 41 (4), pp. 212-218, (2002)

47. L'Allier James, J.: Frame of Reference: NETg's Map to the Products, Their Structure and Core Beliefs. NetG, (1997)

48. Lagoze, C., Arms, W., Gan, S., Hillmann, D., Ingram, C., Krafft, D., Marisa, R., Phipps, J., Saylor, J., Terrizzi, C., Hoehn, W., Millman, D., Allan, J., Guzman-Lara, S., Kalt, T.: Core services in the architecture of the national science digital library (NSDL). In: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 201-209, (2002)

49. Lemnitzer, L., Mossel, E., Simov, K.I., Osenova, P., Monachesi, P.: Using a Domain-Ontology and Semantic Search in an E-Learning Environment. In: Magued, I. (ed.): Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education. Springer, pp. 279-284, (2008)

50. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, Jan-Feb, pp. 76-80, (2003)

51. Marquès, P.: Impacto de las TIC en la enseñanza universitaria. Departamento de Pedagogía Aplicada, Facultad de Educación, UAB. (2000), (Última revisión: 27/08/08). Disponible en: http://dewey.uab.es/PMARQUES/ticuniv.htm.Último acceso: 30-ago-2009.

52. McSherry, D.: Diversity-Conscious Retrieval. In: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, pp. 219-233, (2002)

53. Melville, P., Mooney, J.R., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: 18th National Conference on Artificial intelligence, pp. 187-192, (2002)

54. Milne, I., Rowe, G.: Difficulties in learning and teaching programming—views of students and tutors. Education and Information Technologies, 7, pp. 55–66, (2003)

55. Miller, B., Albert, I., Lam, S.K., Konstan, J., Riedl, J.: MovieLens Unplugged: Experiences with a Recommender System on Four Mobile Devices. In: ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 30-32, (2003)

56. Mobasher, B., Jin, X., Zhou, Y.: Semantically Enhanced Collaborative Filtering on the Web. In: In B. Berendt, e.a.e. (ed.): Web Mining: From Web to Semantic Web., Vol. 3209. Springer pp. 57-76, (2004)

57. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the 5th ACM Conference on Digital Libraries, pp. 195-204, (2000)

58. O' Mahony, M., Smyth, B.: A Recommender System for On-line Course Enrolment: An Initial Study. ACM Conference on Recommender Systems, pp. 133-136, (2007)

59. Orwant, J.: Heterogeneous learning in the Doppelgänger user modeling system. User Modeling and User-Adapted Interaction, 4 (2), pp. 107-130, (1994)

60. Pazzani, M.J.: A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 13 (5-6), pp. 393-408, (1999)

61. Pazzani, M.J., Billsus, D.: Content-Based Recommendation Systems. The Adaptive Web, Vol. 4321. Springer Berlin / Heidelberg. pp. 325-341, (2007)

62. Quinlan, J.R. (ed.): C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc (1993)

63. Quinlan, J.R.: Induction of decision trees. Machine Learning ,1, pp. 81-106, (1986)

64. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Group Lens: An open architecture for collaborative filtering of netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, pp. 175-186, (1994)

65. Ruiz-Iniesta, A., Jimenez-Diaz, G., Gomez-Albarran, M.: Recommendation in Repositories of Learning Objects: A Proactive Approach that Exploits Diversity and Navigation-by-Proposing. In: 9th IEEE International Conference on Advanced Learning Technologies, pp. 543-545, (2009)

66. Ruiz-Iniesta, A., Jimenez-Diaz, G., Gomez-Albarran, M.: User-adaptive Recommendation Techniques in Repositories of Learning Objects: Combining Long-term and

Short-term Learning Goals. In: Fourth European Conference on Technology Enhanced Learning, LNCS 5794, pp. 645-650, (2009)

67. Ruiz-Iniesta, A., Jiménez-Díaz, G., Gómez-Albarrán, M.: Promoting Strong Personalization in Content-based Recommendation Systems of Learning Objects. In: XI International Symposium on Computers in Education (2009) (en prensa)

68. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285-295, (2001)

69. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems In: Heidelberg, S.B. (ed.): The Adaptive Web, Vol. 4321 pp. 291-324, (2007)

70. Schmitt, S.: simVar; a similarity-influenced question selection criterion for e-sales dialogs. Artificial Intelligence Review, 18 (3-4), pp. 195-221, (2002)

71. Shimazu, H.: ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. Artificial Intelligence Review, 18, pp. 223-244, (2002)

72. Shimazu, H.: ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), pp. 1443-1450, (2001)

73. Shimazu, H., Shibata, A., Nihei, K.: ExpertGuide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces. Applied Intelligence, 14, pp. 33-48, (2001)

74. Siemer, J., Angelides, M.C.: Towards an Intelligent Tutoring System Architecturethat Supports Remedial Tutoring, 12 (6), pp. 469-511, (1998)

75. Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: CHI '02 extended abstracts on Human factors in computing systems, pp. 830-831, (2002)

76. Smyth, B.: Case-Based Recommendation. The Adaptive Web, Vol. 4321. Springer Berlin / Heidelberg pp. 342-376, (2007)

77. Smyth, B., Cotter, P.: Personalized Electronic Program Guides for Digital TV. AI Magazine, 22 (2), pp. 89-98, (2001)

78. Smyth, B., Cotter, P.: Surfing the Digital Wave. In: Proceedings of the 3rd International Conference on Case-Based Reasoning and Development, pp. 561-571, (1999)

79. Smyth, B., Keane, M.T.: Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Vol. 102. Elsevier Science Publishers Ltd. pp. 249-293, (1998)

80. Smyth, B., McClave, P.: Similarity vs. Diversity Case-Based Reasoning Research and Development, Vol. 2080. Springer Berlin / Heidelberg. pp. 347-361, (2001)

81. Stauffer, K.: Student Modeling and Web-based Learning Systems. Technical Report, Athabasca University, (1996)

82. Swartout, B., Ramesh, P., Knight, K., Russ, T.: Toward Distributed Use of Large-Scale Ontologies. In: Proceedings of Spring Symposium on Ontological Engineering of AAAI, pp. 138-148, (1997)

83. Tanaka-Ishii, K., Frank, I.: Multi-agent explanation strategies in real-time domains. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp. 158-165, (2000)

84. Thompson, C.A., Göker, M.H., Langley, P.: A Personalized System for Conversational Recommendations. Artificial Intelligence Research, 21, pp. 393-428, (2004)

85. Tintarev, N., Masthoff, J.: A Survey of Explanations in Recommender Systems. In: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop, pp. 801-810, (2007)

86. Upendra, S., Pattie, M.: Social information filtering: algorithms for automating word of mouth. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210-217, (1995)

87. Vollrath, I., Wilke, W., Bergmann, R.: Case-based reasoning support for online catalog sales. IEEE Internet Computing, 2 (4), pp. 47-54, (1998)

88. Wei, K., Huang, J., Fu, S.: A Survey of E-commerce Recommender Systems. In: Proceedings of the International Conference on Service Systems and Service Management, pp. 1-5, (2007)

89. Wiley, D.A.: Learning Object design and sequencing theory. Department of Instructional Psychology and Technology. Brigham Young University (2000)

90. Yang, W., Wang, Z., You;, M.: An improved collaborative filtering method for recommendations' generation. In: In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, pp. 4135-4139, (2004)

91. Youngji, K., Sooho, O., Yongtae, W.: A Case-Based Recommender System Using Implicit Rating Techniques. In: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 522-526, (2002)

92. Yudelson, M., Brusilovsky, P.: NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples. In: Proceedings of 12th International Conference on Artificial Intelligence in Education, AIED 2005, pp. 710-717, (2005)