AVATAR: Los perfiles de usuario

TR0 (versión 1.0; última modificación: 6/4/2005)

1. Introducción

Las preferencias distintivas de los usuarios en un entorno de recomendación personalizada como el de AVATAR, son claramente necesarias para asegurar recomendaciones de alta calidad. Para ello, en este documento vamos a describir diferentes aspectos relacionados con la estructura, organización y actualización de los perfiles de usuario.

En nuestro enfoque consideraremos dos tipos diferentes de preferencias:

1. Preferencias declaradas de forma explícita por los usuarios. Una vez que los espectadores se registran en el sistema para beneficiarse del carácter personalizado de las recomendaciones ofrecidas, éstos pueden declarar explícitamente qué tipo de programas televisivos le interesan (o no le interesan). Además de en esta etapa de configuración inicial del sistema, los usuarios pueden acceder en cualquier momento a su perfil y modificar la información que deseen.
2. Preferencias inferidas por el sistema. AVATAR permite reducir la carga de los usuarios, evitando tediosos procesos de búsqueda de contenidos interesantes, y evitando una interacción constante del usuario con el sistema para declarar explícitamente sus intereses televisivos. De esta forma, la herramienta es capaz de inferir preferencias de los usuarios, a partir de la información de realimentación ofrecida por los mismos. Así, sin interrumpir el visionado de los programas que el espectador esté disfrutando en un momento dado, AVATAR le puede recomendar de forma implícita programas, para que después éste pueda elegir, entre los contenidos sugeridos, aquellos que pueden ser de su agrado. Una vez elegidos los programas, el usuario podrá proceder a su visionado en el momento que desee.

La información almacenada en los perfiles de usuarios, deberá recoger tanto datos personales identificativos del usuario, como sus preferencias en lo referente a programas de TV. Por ello, dichos perfiles pueden ser descompuestos en dos partes claramente diferenciadas, tal como se aprecia en la Fig. 1.

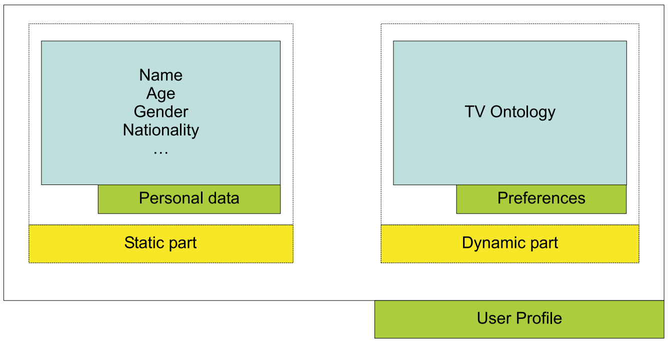

Figura 1. Estructura de los perfiles de usuario

- **Parte estática del perfil**: Almacenará datos personales de los usuarios, como el nombre, la profesión, la nacionalidad, el estado civil, la edad de sus hijos (en caso de tenerlos) o sus aficiones. Será un reducido conjunto de información la que nuestra herramienta solicite al usuario, evitando así que éste tenga que cubrir tediosos formularios. Dada la naturaleza de estos datos, esta parte cambiará con muy poca frecuencia y será almacenada una vez el usuario se registra en el sistema. Aún así, nuestra herramienta ofrece la posibilidad de modificar cualquier dato personal, para así mantener siempre registros convenientemente actualizados. Para ello, la herramienta ofrecerá un menú con diferentes opciones. Algunas de ellas serán explicadas al final de esta sección.
- **Parte dinámica del perfil**: Esta parte del perfil reflejará las preferencias de los usuarios, es decir, almacenará información sobre los programas que les gustan (o disgustan). Dado que define el contexto en el que se enmarcan las preferencias del usuario, a esta parte la llamaremos *contexto semántico personalizado* o $\mathcal{PSC}$. Por ello, es claro que cambiará frecuentemente, y deberá ser mantenida actualizada en todo momento. Esta parte será por tanto el elemento clave en el proceso de recomendación y, por ello, será descrita con más detalle en la Secc. 2.

Antes de centrarnos en una descripción detallada de los perfiles de usuario, es conveniente presentar en este punto algunas de las opciones que los espectadores de AVATAR tienen a su disposición en el menú de usuario.

1. *Update User Profile*: Los usuarios recurrirán a esta opción para modificar cualquier campo incluido en la parte estática del perfil personal.
2. *Provide Feedback Information*: Una vez que las recomendaciones son presentadas a los usuarios, éstos pueden aceptar o rechazar los contenidos sugeridos por la herramienta.
   El agente de realimentación (*feedback agent*), que hemos incluido en la arquitectura del sistema descrita en [1], se encarga de actualizar convenientemente el perfil del usuario, una vez recibida la información de realimentación proporcionada por el mismo. A esta información de los perfiles, una vez haya sido actualizada, accederán los agentes de recomendación para elaborar las siguientes sugerencias presentadas al espectador, tal como se muestra en la Fig. 2.

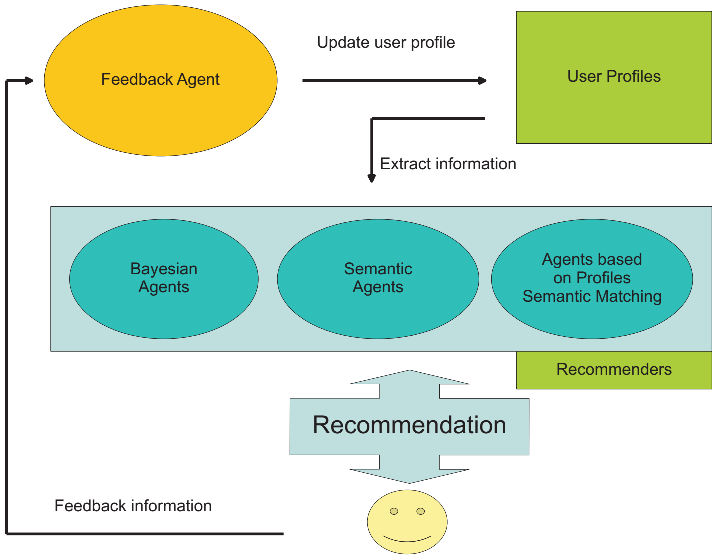

Figura 2. Interacción *feedback agent* - recomendadores

3. *Viewing of Recommended TV Programs*: Esta opción del menú de usuario permitirá seleccionar aquellos programas que formen parte de la recomendación ofrecida por la herramienta, para su posterior visionado.

2. Los Sistemas Recomendadores y el Modelado de Usuarios

Dos de las técnicas más empleadas para afrontar la elaboración de recomendaciones personalizadas, con arreglo a las preferencias de un usuario del sistema, son: (i) filtrado colaborativo (*collaborative filtering*) y (ii) métodos basados en contenido (*content-based methods*). Ambos presentan diferentes ventajas y también desventajas en un entorno de recomendación personalizada, de ahí que haya propuestas en el estado del arte que recurran a un enfoque híbrido, en el que se combinan ambos métodos con el objetivo de complementar las ventajas y desventajas de cada uno de ellos [4].

- El filtrado colaborativo [11] consiste en, dado un usuario objetivo, encontrar usuarios con preferencias similares a las de éste, para reutilizar recomendaciones previamente elaboradas para ellos. La principal desventaja de este método es lo que se ha dado en llamar el *ramp-up*: hasta que no haya un gran número de usuarios registrados en el sistema, cuyas preferencias sean conocidas para éste, las recomendaciones no serán adecuadas para el usuario objetivo. Es decir, el recomendador necesitará conocer información sobre muchos usuarios, de modo que sea fácil encontrar similaridades entre las preferencias del usuario objetivo, y las de los restantes usuarios del sistema. De lo contrario, la calidad de las sugerencias presentadas será demasiado baja, y en consecuencia, el grado de satisfacción de los usuarios muy reducido.

- En contraposición, los métodos basados en contenido [12] se basan en analizar las preferencias de un usuario dado, recomendando contenidos similares a dichas preferencias. La gran ventaja de esta estrategia es que se requiere un conjunto de entrenamiento relativamente pequeño —en comparación al tamaño de la base de datos necesaria en el caso del filtrado colaborativo— para elaborar las recomendaciones, ya que simplemente es necesario analizar las preferencias de un usuario del sistema, y éstas no han de ser comparadas con el resto de usuarios registrados en el recomenador. La mayor parte de los enfoques se basan en el empleo de técnicas basadas en el cálculo del "vecino más cercano", para elaborar las recomendaciones. Sin embargo, la principal limitación de este enfoque está relacionada con un parecido excesivo entre los contenidos sugeridos por el sistema, y los ya conocidos por el usuario antes de emplear el recomendador personalizado, lo que lo convierte en un método de escasa utilidad, si se usa como único método de recomendación.

Otro de los métodos más utilizados para llevar a cabo un proceso de recomendación personalizada, son aquellos basados en conocimiento (*knowledge-based methods*). En ellos, se modela tanto el conocimiento relativo a los contenidos que son objeto de recomendación, como las preferencias de los usuarios. Una vez llevado a cabo tal modelado, será necesario definir un "mapeo" entre ambos dominios, para relacionar los intereses de los usuarios con aquellos contenidos que puedan ser más atractivos para ellos. Además, esta técnica puede ser combinada con las anteriores para explotar las ventajas aportadas por ambas al proceso de personalización. Sirva como ejemplo de esto [3], donde se usan tanto estrategias basadas en conocimiento, como filtrado colaborativo.

Otra de las ventajas de los mecanismos basados en conocimiento, es que no sufren el problema del *ramp-up*, debido a que no se tienen en cuenta las preferencias del resto de usuarios del sistema, sino el "mapeo" entre los intereses de un usuario individual, y el conocimiento modelado en el sistema.

Por otra parte, en el filtrado colaborativo, sólo serán recomendados aquellos contenidos que hayan sido clasificados positivamente por usuarios con preferencias similares a las del usuario objetivo. Esto significa que si el sistema dispone de un nuevo contenido, éste no será ofrecido a un usuario hasta que haya sido declarado interesante por otro usuario similar al primero. Esta limitación no está asociada a los métodos basados en conocimiento, en los que cada contenido nuevo disponible en el sistema, podrá ser ofrecido al usuario, siempre que el "mapeo" entre el conocimiento de su perfil y el conocimiento que modela los contenidos a recomendar, así lo establezca.

En nuestro prototipo el problema de *ramp-up* se evita gracias a la inclusión de los agentes semánticos. Estos últimos se encargarán de recomendar contenidos, aunque éstos no hayan sido previamente clasificados por usuarios del sistema, gracias al descubrimiento —y posterior filtrado y clasificación— de las relaciones semánticas que hemos definido en el documento Doc2, disponible en la página del proyecto AVATAR.

Este tipo de técnicas son las más apropiadas en el contexto del sistema AVATAR. Así, la principal novedad del sistema que proponemos es el enfoque de razonamiento semántico sobre las descripciones de los programas de TV – descritos mediante los metadatos TV-Anytime – y sobre las preferencias personales de los usuarios. Para ello, se hace necesario contar con algún mecanismo de representación de conocimiento. Aprovechando la experiencia ganada en el campo de la Web Semántica [5,6], AVATAR empleará tecnologías típicas de este entorno, para así afrontar los procesos de inferencia y razonamiento semántico, previos a la recomendación.

En relación a la conceptualización de nuestro dominio de aplicación, AVATAR utilizará como base de conocimiento una ontología, conforme al lenguaje OWL (*Web Ontology Language*). De esta forma, mediante una jerarquía de clases y propiedades podemos representar formalmente diferentes aspectos relacionados con el dominio de la TV.

2.1. Modelado de Usuarios: Enfoques existentes

En el estado del arte, existen técnicas de modelado de usuarios basadas en conocimiento [7,8], que resultan de gran utilidad en un sistema como **AVATAR**, en virtud de lo explicado antes. En este tipo de mecanismos, es necesario que el sistema adquiera conocimiento sobre las preferencias de los usuarios (por ejemplo, mediante la formulación de cuestionarios), para después ser modeladas en una base de conocimiento. Una de las técnicas de modelado basadas en conocimiento de mayor relevancia es la conocida como "stereotyping", propuesta por Elaine Rich en 1979 [14]. En este tipo de técnicas es necesario en primer lugar, identificar grupos de usuarios, con una serie de características distintivas, relevantes en el dominio de aplicación concreto del sistema en cuestión. La colección de esas características que definen a los usuarios, es a lo que se llama estereotipo (*stereotype*). Dichas características deben ser formalizadas en un sistema de representación apropiado. Para ello, en la propuesta de Elaine Rich [13] se recurre a una organización jerárquica de estereotipos, de forma que se favorezcan los procesos inferenciales. Antes de seguir describiendo nuestro enfoque, nos centraremos en algunos aspectos relacionados con el uso de estereotipos.

Si bien en esta técnica hay ideas que resultan de gran interés en un sistema basado en razonamiento e inferencia como AVATAR, como la organización jerárquica de las preferencias de los usuarios, nuestro enfoque difiere en algunos aspectos. En concreto, AVATAR dispone de una base de conocimiento, una ontología sobre el dominio de la TV, en la que, como hemos dicho antes, se ha establecido una jerarquía de clases y propiedades describiendo diferentes programas de TV. Por tanto, esta estructura jerárquica resulta claramente útil a la hora de modelar los intereses televisivos de los usuarios, y será precisamente éste el punto común entre nuestro enfoque y el de Elaine Rich. Sin embargo, dicha ontología permitirá modelar las preferencias de usuarios individuales, esto es, en ningún momento estamos considerando las preferencias de grupos de usuarios [9], como ocurre en los estereotipos.

Sin embargo, la identificación de grupos de usuarios con intereses similares es claramente ventajoso en un entorno de personalización. Esta es la idea que subyace de un enfoque basado en filtrado colaborativo, donde se tienen en cuenta las preferencias de usuarios similares. Por esta razón, AVATAR llevará a cabo un proceso de detección de similaridad entre las preferencias de varios espectadores, de modo que sea posible sugerir a un usuario aquellos programas que hayan interesado a otros con gustos parecidos a los suyos. La similaridad entre las preferencias de los espectadores se evaluará tanto desde un punto de vista sintáctico, como semántico, tal como describimos en el documento Doc3, disponible en la página del proyecto

AVATAR. Además una de las métricas empleadas en esta propuesta de detección de similaridad semántica será lo relacionados que estén los contenidos enfrentados, de modo que si inferimos relaciones implícitas entre dos contenidos, consideraremos que son muy similares.

A la vista de lo que acabamos de explicar, podemos concluir diciendo que nuestra propuesta comparte ideas con los estereotipos de Rich, esto es, también parte de un modelado jerárquico de las preferencias de los usuarios. Sin embargo, en lugar de definir una nueva jerarquía de estereotipos, en la que se modelen intereses de grupos de usuarios, nuestro sistema reutiliza una ontología de contenidos televisivos, modelando usuarios individuales de forma dinámica, a medida que el sistema recibe información sobre las acciones llevadas a cabo por los usuarios (esto es, su comportamiento de visionado).

El modelado dinámico de usuarios ya ha sido tratado en trabajos previos, como [16]. En este artículo se presenta un sistema recomendador, llamado AlFresco, en el que se registran usuarios interesados en pinturas y frescos de origen italiano. El usuario puede interactuar con el sistema escribiendo frases, navegando por diferentes enlaces y usando una pantalla táctil. Este recomendador emplea un modelado dinámico de usuarios que se construye de forma incremental, a medida que el sistema conoce más información sobre las preferencias de éstos. El modelo aquí descrito consta de dos módulos diferentes:

- El primero de ellos almacena el conocimiento que tiene el sistema sobre un usuario.
- El segundo representa los intereses potenciales de dicho usuario. Para ello, se emplea una red de activación/inhibición, en la que cada nodo representa un área de interés, como una determinada época o un estilo pictórico. La activación de una nodo, causada por ejemplo, porque el usuario ha preguntado algo relacionado con un área de interés concreto, provoca (i) la activación de los nodos correspondientes a áreas relacionadas con la primera, y conectadas a ella mediante nodos de activación (*activation links*) y (ii) la inhibición de los nodos que representan áreas incompatibles (unidas mediante enlaces inhibidores o *inhibitory links*).

Una vez revisados algunos de los enfoques más relacionados con nuestra propuesta, en la siguiente sección describimos la estructura de los perfiles manejados por nuestra herramienta.

2.2. Modelado de Usuarios mediante Ontologías: Ventajas

Son dos las partes que podemos identificar en los perfiles del sistema AVATAR:

- *Parte dinámica del perfil*: Será una ontología OWL construida dinámicamente a medida que el sistema dispone de información adicional sobre el usuario. A partir de esto es fácilmente deducible el nombre que hemos asignado a nuestros perfiles: *perfiles-ontología*. La actualización de las preferencias definidas en esta parte del perfil, será clave para obtener recomendaciones personalizadas y de alta calidad.
- *Parte estática del perfil*: Serán un conjunto de propiedades *Datatype* que permitirán asignar valores a los campos definidos en la Secc. 1 y a otros que completan la parte de datos personales de los usuarios. Para representar a cada usuario utilizaremos la clase *User* y cada uno de ellos estará identificado unívocamente por un identificador numérico. Los dominios de las propiedades *Datatype* serán los tipos de datos de XML Schema enumerados en la Tabla 1.

  La inclusión de las ontologías como técnica para modelar usuarios presenta varias ventajas:

  - En primer lugar, hemos dicho que el proceso de razonamiento semántico requiere un mecanismo formal de representación del conocimiento. Dado que el razonamiento en AVATAR se lleva a cabo sobre las preferencias de los usuarios, es decir, sobre los programas que interesan a los mismos, es necesario que sus preferencias estén representadas formalmente. Este requisito se satisface gracias a la ontología OWL implementada. Esta representación formal permitirá aplicar herramientas existentes en el estado del arte para explorar la base de conocimiento del sistema. Este proceso

PROPIEDADES	DOMINIO	RANGO
hasID	clase *User*	xsd:nonNegativeInteger
hasName	clase *User*	xsd:string
hasAge	clase *User*	xsd:nonNegativeInteger
hasJob	clase *User*	xsd:string
hasLikings	clase *User*	xsd:string
hasMaritalStatus	clase *User*	xsd: string
hasChildrenAge	clase *User*	xsd:nonNegativeInteger

Tabla 1. Propiedades *Datatype* para representar la parte estática de los perfiles de usuario

conducirá a la extracción de relaciones semánticas entre intancias de las clases declaradas en la ontología, importantes de cara al proceso de recomendación. En trabajos previos [15] ya se ha tratado el descubrimiento de asociaciones entre conceptos que estaban ocultas en una base de conocimiento, conduciendo así a procesos inferencias muy interesantes de cara a búsquedas semánticas.

- La ontología permite almacenar jerárquicamente el conocimiento sobre las preferencias de usuarios, en lugar de utilizar listas planas que impiden expresar las relaciones existentes en un dominio de aplicación concreto [10]. Por ejemplo, si a un usuario le gustan las retranmisiones deportivas de Fórmula 1 (clase *F1 Broadcasting Programs*) y los programas informativos donde se resumen los partidos de fútbol de la jornada liguera (clase *Football Informative Programs*), es interesante almacenar en su perfil los dos tipos de programas que le interesan, pero bajo una misma categoría: en este caso, programas deportivos (clase *Sport_Programs*).
- Además la representación ontológica de perfiles permite hacer un matching semántico entre perfiles de usuarios, es decir, es posible determinar si dos perfiles son "similares', más allá de una simple comparación sintáctica. Por ejemplo, un usuario al que le gustan las retransmisiones de F1 es más "parecido" a otro que le gustan los programas sobre fútbol, que a un tercero al que le gustan los concursos. Esto se debe a que los dos primeros espectadores comparten preferencias pertenecientes a una clase común (en este caso *Sport Programs*), no así en el caso del primer y el tercer usuarios.

Nuestro enfoque propone un proceso de actualización dinámico de las preferencias de los usuarios, basado en que a medida el sistema conozca nuevas preferencias sobre un usuario particular, complete la información almacenada en su perfil. Para ello, bastará con añadir a la ontología de perfiles las clases correspondientes, con arreglo a la base de conocimiento del sistema, esto es, la ontología de TV, junto con las propiedades relativas a cada una de dichas clases. Este procedimiento se abordará en la siguiente sección.

3. *Perfiles-Ontología*: Sintaxis y Actualización

En primer lugar, es conveniente destacar que en el proceso de construcción de cada *perfil-ontología* podemos diferenciar dos elementos claves:

- La base de conocimiento del sistema, esto es, la ontología de TV OWL.
- El lenguaje de consulta **LIKO**, utilizado para explorar la ontología antes citada, y cuyos operadores recordamos en la Tabla 2.

Tabla 2. Operadores de consulta del lenguaje **LIKO**

Operators	**Input**	**Output**
$\triangleright$	Una clase de la ontología.	Subclases directas relativas a la clase de entrada.
$\triangleleft$	Una clase de la ontología.	Superclases directas relativas a la clase de entrada.
$>>$	Una propiedad de la ontología.	Subpropiedades directas relacionadas con la propiedad de entrada.
$<<$	Una propiedad de la ontología.	Superpropiedades directas referidas a la propiedad de entrada.
$>\Leftarrow$	Instancias de clases de la ontología.	Propiedades donde el dominio es una instancia de la clase de entrada.
$>\Rightarrow$	Instancias de class de la ontología.	Propiedades donde el rango es una instancia de la clase de entrada.
$>\odot$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades transitivas contenidas en la ontología.
$>\oplus$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades inversas contenidas en la ontología.
$>\hookleftarrow$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades simétricas contenidas en la ontología.
$>\ominus$	Una propiedad definida en la ontología.	Propiedad inversa de la propiedad de entrada.

Una vez que el sistema identifica un programa que interesa (o no) al usuario el proceso de actualización del perfil se lleva a cabo mediante los pasos indicados a continuación:

1. Dicho programa será una instancia de una de las clases definidas en la ontología. En primer lugar, será necesario extraer dicha clase, para así identificar a qué categoría de contenidos pertenece dicho este programa (por ejemplo, “Action_Movies”).
2. En segundo lugar, se aplicará *recursivamente* el operador **LIKO** $\triangleleft$ sobre la clase extraída en el paso anterior. El hecho de que dicho contenido se aplique recursivamente se debe a que este operador devuelve la superclase *directa* correspondiente a la clase indicada[1], sin embargo, dicha clase puede tener otras superclases en niveles superiores en la jerarquía de conocimiento establecida en la ontología (por ejemplo, es posible establecer una jerarquía como la indicada a continuación: películas → películas de acción → películas de acción del oeste). Este proceso recursivo finalizará cuando **LIKO** devuelva el elemento raíz de la ontología de TV, esto es, la clase “TV Contents”.
3. Cada una de las clases descubiertas en el paso anterior, se añade a la jerarquía dinámica establecida en la ontología de perfiles. De este modo, no se van anotando las preferencias de los usuarios como si de listas planas se tratara, sino que se recurre al conocimiento del dominio de la TV, para mantener una estructura jerárquica y organizada. Pongamos en este punto un ejemplo para entenderlo mejor.
   Supongamos que un usuario declara explícitamente que le gustan las retransmisiones de fútbol. El sistema aplica **LIKO** sobre la clase “Football Broadcasting” (*Football Broadcasting* $\triangleleft$) y obtiene la clase “Sport_Broadcasting”. De nuevo se aplica el citado operador **LIKO** (*Sport_Broadcasting* $\triangleleft$), para obtener la clase “Sport_Programs”. Las tres clases descubiertas – *Sport_Programs* → *Sport_Broadcasting* → *Football Broadcasting* – se añaden a la jerarquía de conocimiento dinámica del perfil, y se actualizan los pesos de interés correspondientes a las mismas, tal como se explicará en la Secc. 3.2.
   Imaginemos que a continuación el sistema tiene información sobre un programa deportivo de carácter informativo, que ha sugerido al usuario y que éste ha elegido. En este caso, se aplicará de nuevo el

[1] Si A es la superclase de B, se dice que A “subsume” a B, de acuerdo a lo que se llama *principio de “subsumption”*.

lenguaje **LIKO**, y se irá completando la jerarquía que se había empezado a contruir anteriormente. Así, en el perfil del usuario figurará la jerarquía mostrada en la Fig. 3.

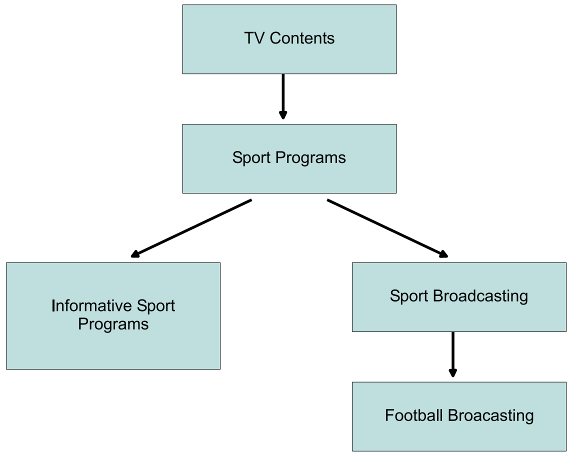

Figura 3. Estructura de los perfiles de usuario

Es decir se va replicando la ontología OWL en el perfil de usuario, pero de forma dinámica, con arreglo a la información de la que dispone el sistema sobre el usuario en cuestión.

4. Además de almacenar en el perfil la jerarquía de clases correspondiente a las preferencias de los usuarios, también será necesario guardar propiedades (*Object* y *Datatype*) que definan características de las instancias de estas clases. Este proceso se detalla en la siguiente sección, donde nos centraremos en la sintaxis que emplearemos para referirnos a la información almacenada en el *perfil-ontología* del usuario.

3.1. Sintaxis de los Perfiles de Usuario

En lo que se refiere a la información almacenada en los perfiles de usuario, podemos diferenciar dos posibles situaciones:

- En primer lugar, el sistema puede conocer simplemente la clase a la que pertenecen los programas que gustan (o no) al usuario. Por ejemplo, el espectador al registrarse en el sistema, puede declarar explícitamente que le gustan las películas de acción. En este caso, teniendo en cuenta lo explicado en la Secc. 3, es claro que, si hay clases que "subsumen" a ésta, será necesario aplicar el lenguaje **LIKO** para extraer y almacenar la jerarquía de contenidos apropiada. En este caso se almacenará la sucesión de clases de la jerarquía, concatenadas mediante el símbolo que identifica la "subsumption", esto es, $\sqsupseteq$. De cara a una recomendación personalizada, este es el peor caso, ya que la cantidad de información que el sistema conoce sobre las preferencias del usuario es muy limitada, lo que dificulta los procesos de inferencia y personalización.
- En segundo lugar, es posible que el sistema no sólo conozca una clase genérica, sino una o varias instancias de dicha clase, junto con propiedades que definen características concretas de éstas. Para reflejar esta situación en el *perfil-ontología*, se recogerá la clase en cuestión, concatenándole mediante puntos (.) todas aquellas propiedades que definen cada una de las instancias que vamos a almacenar.

En relación a este último punto es conveniente destacar que en la base de conocimiento OWL del sistema es posible identificar unas estructuras que hemos denominado *secuencias de propiedades* (PS en adelante), definidas a partir de las propiedades de la ontología.

Así, podemos definir PS como un conjunto de entidades (clases o valores concretos) unidas entre sí mediante propiedades (*Object* o *Datatype*, respectivamente). Dado que en la ontología aparecen instancias de estas clases, también es posible definir a partir de PS una instancia de la misma, que representaremos en adelante como ps. Además, tanto para las secuencias como para sus instancias, es posible definir su longitud como el número de propiedades que las integran. De ello se desprende que *Length* (PS) = *Length* (ps).

Por otra parte, las secuencias de propiedades definidas en la ontología de TV siempre tienen un inicio que es la instancia a la que están asociadas, y deben cumplir que la instancia que figura como rango en cada una de sus propiedades, sea la misma que aparece como dominio en la siguiente, siendo posible de esta forma construir la secuencia.

Con el objeto de almacenar las propiedades que definen cada instancia contenida en el perfil, usaremos en éste parte de las secuencias que acabamos de describir. Así, las secuencias de los *perfiles-ontología* deben cumplir dos condiciones:

- Primero, la última propiedad de cada secuencia en el perfil debe ser una propiedad *Datatype*.
- Por otra parte, en lugar de almacenar la secuencia completa definida en la ontología OWL para cada programa (instancia) contenido en el perfil, emplearemos únicamente secuencias de longitud uno o de longitud dos. Teniendo en cuenta que la longitud de la secuencia es el número de propiedades que contiene, y que la última siempre debe ser de tipo *Datatype*, es fácil ver que los perfiles almacenan para cada programa, o bien propiedades *Datatype* definiendo valores concretos de éstos (longitud uno), o bien instancias relacionadas con el programa mediante una propiedad *Object*, junto a propiedades *Datatype* asociadas a dichas instancias (longitud dos).

En relación a este último punto, hemos optado por incluir en el perfil aquellas instancias que estén directamente relacionadas con el programa, ya que de esta forma el usuario tendrá opción de declarar su nivel de interés —como veremos más adelante— tanto en los programas que le han gustado/disgustado, como en las instancias asociadas directamente a ellos. Esta es la razón que nos ha llevado a incorporar secuencias de longitud uno y dos en nuestros *perfiles-ontología*.

De esta forma, las secuencias del perfil serán siempre subsecuencias de las definidas en la ontología OWL. Esto impide sobrecargar inútilmente el perfil y por otra parte, no supone una pérdida de conocimiento de cara al proceso de razonamiento, ya que para llevar a cabo las inferencias necesarias en nuestro sistema, bastará con localizar las instancias del perfil en la ontología y analizar las secuencias completas.

Para reflejar las secuencias de propiedades en nuestro perfil hemos incluido los símbolos descritos a continuación:

- Cada instancia (programa) del perfil puede tener concatenadas varias secuencias. Para ello, intercalaremos el símbolo $\sqcap_P$ entre cada una de dichas secuencias. La primera de ellas, como hemos dicho al comienzo, se añade a la instancia en cuestión mediante un punto ($\cdot$).
- Para almacenar los valores concretos de las propiedades *Datatype* que finalizan cada secuencia empleamos el constructor

  Datatype property [*Value*]

  siendo *Datatype property* la propiedad en cuestión y *Value*, el valor tomado por la misma (un tipo de datos concreto).
- Para expresar varias propiedades *Datatype* referidas a una misma instancia, intercalaremos el símbolo ; entre los constructores *Datatype property* [*Value*] indicados antes. Así, por ejemplo

Actors. (hasName [C.Eastwood] ; hasAge [75])

serían dos secuencias de propiedades de longitud 1, asociadas a una instancia de la clase *Actors* contenida en el perfil.

Finalmente expresamos formalmente las secuencias almacenadas en el *perfil-ontología* del usuario, considerando además los constructores y símbolos de la sintaxis propuesta. Éstas son representadas mediante la forma extendida *Backus-Naur* a continuación (donde *op_name* identifica las propiedades *Object* y *dp_name*, las *Datatype*), e ilustradas en la Fig. 4.

$$
\begin{array}{ll}
<\mathit{user\_profile\_preferences}> & ::= \{<\mathit{classes\_chain}>\} \\
<\mathit{classes\_chain}> & ::= <\mathit{class\_name}>\ [\text{"}\sqsupseteq\text{"}<\mathit{class\_name}>\ \{<\mathit{classes\_chain}>\}]\ [<\mathit{class}>] \\
<\mathit{class}> & ::= <\mathit{class\_name}>\ [\text{"."}<\mathit{property}>\ \{\text{"}\sqcap_p\text{"}<\mathit{property}>\}\ ] \\
<\mathit{property}> & ::= <\mathit{op\_name}>\text{"."}<\mathit{dp\_name}>\text{"["}\mathit{value}\text{"]"}\{\ ;\ <\mathit{dp\_name}>\text{"["}\mathit{value}\text{"]"}\} \\
<\mathit{class\_name}> & ::= \mathit{letter}\{\mathit{letter}|\mathit{digit}|\text{"\_"}\} \\
<\mathit{op\_name}> & ::= \mathit{letter}\{\mathit{letter}|\mathit{digit}|\text{"\_"}\} \\
<\mathit{dp\_name}> & ::= \mathit{letter}\{\mathit{letter}|\mathit{digit}|\text{"\_"}\}
\end{array}
$$

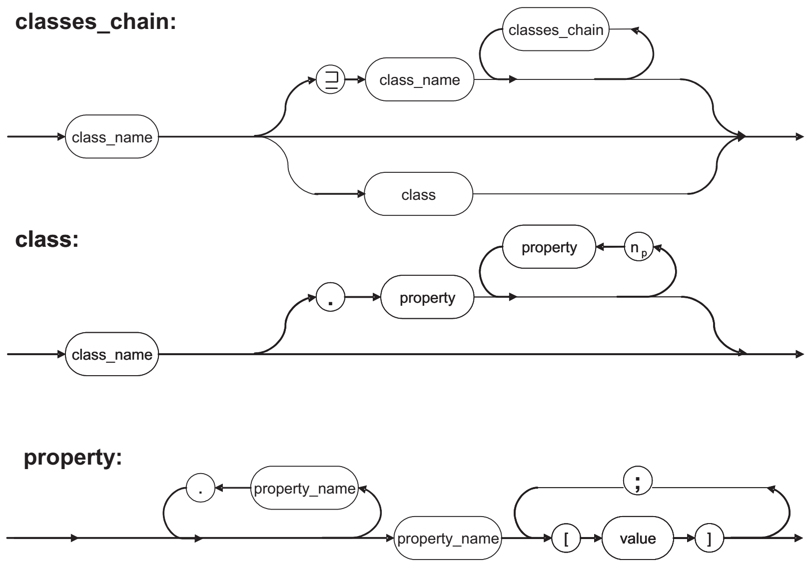

Figura 4. Organización de los perfiles de usuario expresada mediante la forma extendida *Backus-Naur*

A la vista de la figura es fácil apreciar lo que hemos comentado anteriormente. En el perfil de usuario siempre habrá como mínimo un clase. Si ésta no es lo suficientemente general en la jerarquía de conocimiento establecida por la ontología, será necesario concatenarle mediante el símbolo $\sqsupseteq$ las clases oportunas (según el principio de "subsumption"). Además de dicha sucesión de clases, en el perfil de usuario también pueden figuran diferentes secuencias de propiedades (siempre formadas por uan propiedad *Datatype* o bien, por una *Object* y una o varias *Datatype* separadas por ;), que se concatenan a estas clases mediante puntos.

Para concluir esta sección, mostramos un ejemplo de perfil de usuario en AVATAR. Supongamos un usuario de nacionalidad española, médico, casado y sin hijos. Su perfil contiene información sobre (i) un partido de fútbol del Liverpool, (ii) la comedia "La máscara", protagonizada por el actor canadiense Jim Carrey y (iii) un informativo sobre deportes, presentado por el deportista Carl Lewis. Una vez consultada la jerarquía de conocimiento definida en la ontología de TV[2], el perfil de este espectador será el mostrado en la Tabla 3.

hasName [Javier López Paz]	hasID [215]	hasJob [Doctor]	*Datos personales* (*parte estática*)
hasGender [Male]	hasMaritalStatus [Married]		
hasNationality [Spanish]	hasChildrenAge [None]	...	

Sport_Programs $\sqsupseteq$ Sport_Broadcasting $\sqsupseteq$ Soccer_Broadcasting. (hasTeam [Liverpool])
Sport_Programs $\sqsupseteq$ Informative_Sport_Programs. (hasPresenter.hasName [C. Lewis])

Movies $\sqsupseteq$ Comedy_Movies. (hasTitle [The Mask]) $\sqcap_P$ (hasStarringActor. (hasName [J.Carrey] ; hasNationality [Canadian]))

Tabla 3. Ejemplo de un perfil de usuario en AVATAR

Como ya dijimos antes, el sistema AVATAR no sólo necesita conocer las preferencias de los usuarios, sino también el nivel de interés asociado a éstas. Por esta razón, la próxima sección describe las métricas usadas en AVATAR para actualizar los perfiles y para elaborar recomendaciones personalizadas.

3.2. Proceso de Actualización de los Perfiles de Usuario en AVATAR

Cada uno de los agentes de recomendación incluidos en la arquitectura propuesta para nuestro sistema [2], ofrece recomendaciones personalizadas a los usuarios ordenadas según su relevancia y con arreglo a la información almacenada en sus perfiles. Una vez estos contenidos hayan sido presentados a los usuarios, el sistema debe inferir datos de las acciones llevadas a cabo por ellos (por ejemplo, los programas han sido aceptados o rechazados). Para traducir el comportamiento de visionado del usuario a su perfil, AVATAR actualizará una lista de índices cuyos valores determinarán el orden de los programas incluidos en recomendaciones futuras.

En adelante supondremos que AVATAR ha mostrado una primera recomendación al espectador, donde figurarán diferentes programas (instancias de las clases definidas en la ontología de TV implementada). A partir de esta recomendación, el sistema extraerá las clases relativas a las instancias contenidas en la misma. Cada una de estas clases (a las que llamaremos C_i en adelante) debe ser añadida al *perfil-ontología* de este usuario, junto con la jerarquía de clases que las "subsumen", de acuerdo al conocimiento de la ontología. Además, los índices asociados a estas clases deben ser calculados y añadidos al perfil (se crea una propiedad *Datatype* para cada uno de los índices que definiremos en esta sección).

Es conveniente destacar que estos índices se calcularán tanto para una clase C_i, como para cada una de las instancias de ésta ($Instance_k^i$). Así, las expresiones que introduciremos en esta sección, permitirán calcular los índices de cada una de las instancias, de forma que después se sumarán estas contribuciones para obtener el índice asociada a la clase a la que pertenecen.

En el caso de las secuencias de longitud dos definidas en el perfil, conviene hacer alguna aclaración en relación a la actualización de los índices asociados a las instancias contenidas en las mismas. Hemos

[2] En este ejemplo, el programa presentado por Carl Lewis (*Informative_Sport_Programs*) y el partido de fútbol (*Soccer_Broadcasting*) son instancias de clases "subsumidas" por la clase *Sport_Programs*.

dicho que en este tipo de secuencias, cada programa del perfil está relacionado con otras instancias mediante propiedades *Object*. Por esta razón, es necesario tener en cuenta dos situaciones:

1. En caso de que las instancias de la recomendación ya estuviesen incluidas en el perfil (por ejemplo, un actor o un presentador que han participado en programas ya conocidos por el usuario), el peso que figure asociado a éstas para cada índice (en adelante identificado mediante el sufijo *old* asociado al índice correspondiente) deberá ser actualizado, añadiéndole una nueva contribución calculada mediante las expresiones que presentaremos en esta sección. A la expresión final de dicho índice la identificaremos en las expresiones mostradas mediante el sufijo *new*.
2. Por otra parte, si se modifica el índice relativo a una instancia del perfil, se actualizará también el índice correspondiente de las instancias relacionadas con ella mediante las mencionadas propiedades *Object*.

Por ejemplo, imaginemos que en el perfil figura una instancia correspondiente a una película de acción con un determinado actor protagonista. Si en futuras sugerencias, el usuario rechaza programas en los que ha participado este actor, el índice relativo a la instancia que identifica al actor deberá ser reducido (factor 2), pero también reduciremos el peso asociado a la película de acción donde éste interviene (factor 1), y por consiguiente, el peso de la clase *Action-Movies*.

Una vez que el índice de cada clase ha sido calculado, los valores serán normalizados en el rango [-1,1]. Los valores positivos identifican clases que interesan al usuario, mientras que los negativos se refieren a clases que no han sido de interés para él.

Los índices que hemos nombrado hasta ahora son presentados a continuación.

3.2.1 Índice de interés (*Degree Of Interest* ó DOI)

Este factor mide el nivel de interés del usuario, referido a cada clase y a cada instancia definida en su *perfil-ontología*. La expresión matemática que permite calcular el índice DOI asociado a una clase (por ejemplo, c_i) se calcula sumando el valor de este índice para cada una de las instancias almacenadas en el perfil del usuario ($Instance_k^i$), tal como refleja la Ec. 1.

$$\mathbf{DOI}(\mathbf{C_i}) = \sum_{k=1}^{N} DOI_{new}(Instance_k^i) \qquad (1)$$

En la Ec. 1, **N** es el número de instancias de la clase C_i contenidas en el *perfil-ontología* del usuario, **Instance$_\mathbf{k}$** es la instancia k-ésima de dicha clase y **DOI**$_{new}$ **(**$Instance_k^i$**)** es nivel de interés de cada una de dichas instancias.

Así, el sistema puede cuantificar el interés de un usuario por ejemplo en un actor (instancia de la clase *Actors*) o en una determinada película de acción (instancia de *Action-Movies*). AVATAR ofrece al usuario una interfaz donde éste puede acceder a las clases definidas en su perfil (por ejemplo, *Action-Movies* o *Actors*), las instancias de estas clases (Rocky II o Clint Eastwood), y las propiedades referidas a dichas clases. Además, los índices de las instancias del perfil pueden ser fijadas manualmente por el usuario, o bien, ser actualizaas por el sistema de forma automática. Este enfoque permite al recomendador corregir posibles errores en las sugerencias ofrecidas, consiguiendo así una rápida convergencia entre las preferencias reales del usuario y las inferidas por AVATAR.

Para calcular el campo *DOI*$_{new}$ ($Instance_k^i$) mostrado en la Ec. 1, tres índices deben ser considerados:

- *Índice de realimentación (Index Of Feedback ó IOF)*: Este factor identifica la información de realimentación que el usuario puede proporcionar al sistema. Para cada programa incluido en la recomendación, el valor del índice es 1 si el usuario selecciona este programa para su posterior visionado, y -1 en caso contrario. En caso de que el usuario hubiese proporcionado información de realimentación adicional (diciendo que el programa es muy interesante, interesante, poco interesante, etc.), este índice podría tomar valores mayores o menores, en función del interés declarado por el espectador.

- *Índice de visionado (Index Of Viewing ó IOV)*: Este índice es la relación entre el tiempo que un usuario ha visto un programa, y su duración total. Este factor se calcula sólo cuando un programa ha sido seleccionado por el usuario; en caso contrario su valor será 1. Imaginemos el siguiente escenario para validar su utilidad: AVATAR presenta al usuario una lista de programas personalizados. Él selecciona uno de ellos porque piensa que podría ser interesante. Sin embargo, cuando han transcurrido pocos minutos, el usuario suspende el visionado del programa ya que descubre que realmente este contenido no le interesa. Mediante el índice IOV, cuya expresión figura en la EC. 2, este error podría evitarse ya que el peso asignado a este programa sería reducido.

$$\mathbf{IOV}(Instance_k^i) = \frac{\text{Viewing Time}\,(Instance_k^i)}{\text{Duration}\,(Instance_k^i)} \tag{2}$$

- *Antigüedad de visionado* (*Antiquity of Viewing* ó AOV): Este índice representa el tiempo que ha transcurrido (en días) desde que el usuario selecciona un programa recomendado para ser visto, hasta que finalmente lo ve. Al igual que ya dijimos en el caso del índice IOV, sólo será calculado cuando un programa sea aceptado por el usuario, en caso contrario su valor será 1. Este factor está basado en lo siguiente: el espectador verá primero aquellos programas que sean más interesantes para él, de entre todos los contenidos almacenados para ser vistos.

Teniendo en cuenta los índices presentados hasta ahora, el factor DOI será calculado mediante la Ec. 3 presentada a continuación.

$$\mathbf{DOI}_{new}(Instance_k^i) = DOI_{old}(Instance_k^i) + \underbrace{\frac{IOV(Instance_k^i) * IOF(Instance_k^i)}{AOV(Instance_k^i)}}_{K_1} \tag{3}$$

La fórmula anterior permite actualizar el índice de interés relativo a la instancia $Instance_k^i$, siendo ésta uno de los programas que figuran en la recomendación ofrecida al usuario. El campo $DOI_{old}(Instance_k^i)$ será el valor que tenga asignado en el perfil la instancia $Instance_k^i$ antes del proceso de actualización, y valdrá 0, en caso de que dicha instancia no estuviese ya definida en éste.

Como ya dijimos antes, este programa en la ontología de TV estará relacionado con otras instancias mediante propiedades *Object*. El índice de interés correspondiente a estas últimas instancias (representadas en la Ec. 4 como *Related-Instances*$_j$) se calculará sumando al peso que ya tenía asignado en el perfil (en caso de que esta instancia figure en el perfil), la contribución K_1 que hemos definido en la Ec. 3, tal como se muestra en la siguiente expresión:

$$\mathbf{DOI}_{new}(Related-Instances_j) = DOI_{old}(Related-Instance_j) + K_1 \tag{4}$$

De esta forma, el proceso de actualización propuesto para el índice de interes incluye: (i) calcular el valor de este índice para cada instancia presente en la recomendación (mediante la Ec. 3), (ii) actualizar los intereses de las instancias relacionadas con las primeras mediante propiedades *Object* (aplicando la Ec. 4) y (iii) calcular el interés asociado a cada clase, sumando los pesos de todas las instancias de esta clase definidos en el *perfil-ontología* del usuario (Ec. 1).

Una vez presentadas las ecuaciones que permiten calcular los índices de interés, presentamos un escenario en relación a este índice. Supongamos que una clase determinada definida en el *perfil-ontología* tiene asociado una nivel de interés de valor elevado (valores muy cercanos a 1), dado que el usuario ha seleccionado muchas instancias de esta clase entre los programas recomendados por el sistema. AVATAR sugiere nuevas instancias de esta clase (o bien recomienda instancias que tienen en el perfil un alto valor de nivel de interés), y sin embargo el usuario en este momento las rechaza. En esta situación es claro que el interés relativo a dicha clase debe ser reducido en gran medida. De no ser así, dicho índice podría seguir siendo más elevado que el interés del resto de clases del perfil, lo que conduciría a incluir en recomendaciones futuras nuevas instancias de la misma que podrían ser rechazadas de nuevo por el espectador. Esto llevaría a disminuir el grado de satisfacción de los usuarios.

Para prevenir este tipo de problemas, nuestro prototipo usará el siguiente criterio: si el índice DOI asociado a una clase es mayor que 0.75 y una instancia de dicha clase es rechazada por el usuario, este índice tomará un valor igual a la media de los índices de interés positivos, cuyo valor sea menor que el de la clase en cuestión, declarados en el perfil del usuario. Si no existe ningún interés positivo menor que el de la clase en cuestión, el proceso de actualización se llevará a cabo mediante las ecuaciones presentadas antes. Este enfoque tiene sentido ya que, en este caso, en el perfil contendría clases con índices DOI muy elevado (mayor que 0.75 en el peor caso, y mayor que el índice analizado) y, por tanto serían estas clases las que aportarían sus instancias a la recomendación final. De esta forma, las instancias de la clase rechazada por el espectador, no serían incluidas en la lista de programas elaborada por el sistema.

3.2.2 Índice de confianza (*Confidence Index*)

Este índice cuantifica el éxito o fracaso obtenido por AVATAR en las recomendaciones ofrecidas en el pasado. El índice de confianza se calculará de la misma forma que el factor DOI. La única variación será la contribución que se añadirá a las instancias de cada clase para actualizar el valor correspondiente, identificada como K_2 en la Ec. 5, correspondiente a la instancia $Instance_k^i$.

$$\mathbf{Confidence}_{new}(Instance_k^i) = Confidence_{old}(Instance_k^i) + \underbrace{\frac{IOF(Instance_k^i)}{Order(k)}}_{K_2} \tag{5}$$

En la Ec. 5, **Order(k)** representa el orden del programa k-ésimo en la recomendación presentada al usuario, mientras IOF ($Instance_k^i$) es el índice de realimentación descrito antes.

De la misma forma que hemos comentado en el caso del índice DOI, también se actualizarán los índices de confianza de las instancias relacionadas con $Instance_k^i$ mediante propiedades *Object*. En este caso la contribución que se añadirá al peso ya asignado a estas instancias (recordar Ec. 4) será K_2, en lugar de K_1.

A la vista de la Ec. 5, es fácil ver que si el usuario elige contenidos de TV situados en las primeras posiciones de la recomendación (orden = 1,2, ...), el índice de confianza será incrementado fuertemente, mientras que si los programas seleccionados ocupan los últimos lugares, el incremento de este índice será muy pequeño. Sin embargo, si los programas clasificados en las primeras posiciones son rechazados, el índice de confianza para estas instancias, y para sus clases, serán decrementados en gran medida (recordar que en este caso $IOF(Instance_k^i)$ valdría -1). De la misma manera, el decremento será menor si los contenidos rechazados ocupasen las últimas posiciones.

3.2.3 Índice de relevancia (*Relevance index*)

Este índice es usado por AVATAR para ordenar y presentar los programas recomendados, de forma que aquellas clases contenidas en el perfil del usuario con valores elevados, aportarán más instancias a la lista de contenidos sugeridos, y por tanto ocuparán las primeras posiciones de la recomendación.

Este índice de relevancia se calcula combinando los índices de interés (DOI) y confianza. Al igual que los índices presentados hasta ahora, el factor de relevancia está asociado a cada clase y a cada instancia del perfil. Para calcular la relevancia de una clase, al igual que en los dos índices que acabamos de presentar, sumamos la contribución de cada instancia. El cómputo de cada una de éstas consiste en multiplicar su nivel de interés —una vez actualizado, esto es, DOI_{new}— por el valor absoluto de su confianza ($Confidence_{new}$), tal como indica la Ec. 6.

$$\mathbf{Relevance}_{new}(Instance_k^i) = Relevance_{old}(Instance_k^i) + DOI_{new}(\mathit{Instance}_k) * |Confidence_{new}(\mathit{Instance}_k)| \quad (6)$$

Finalmente, mostramos un compendio de escenarios donde se muestra la evolución del índice de relevancia, frente a las acciones llevadas a cabo por un usuario (programas aceptados o rechazados).

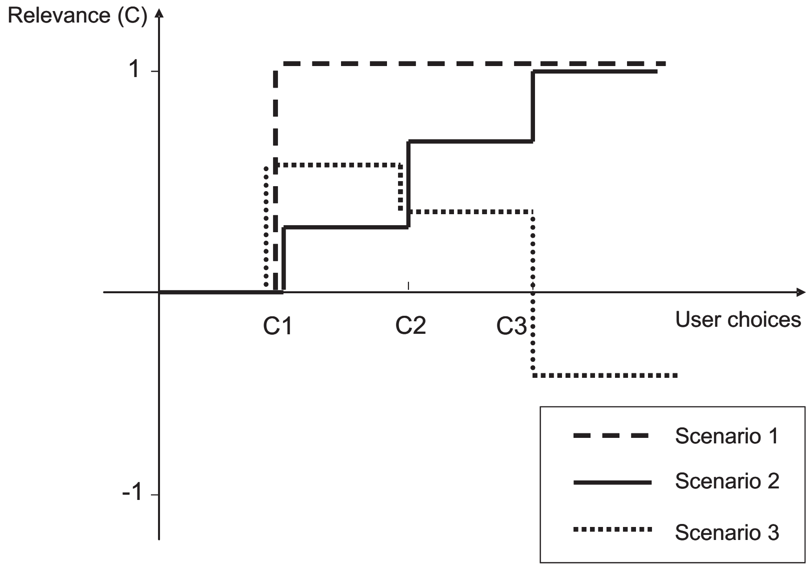

Figura 5. Evolución del índice de relevancia frente a las selecciones del usuario

En el eje horizontal de la Fig. 5 se muestran tres selecciones (C1, C2 y C3) de un mismo usuario. En el eje vertical, representamos el índice de relevancia de una clase genérica C. Cabe destacar que C1, C2 y C3 serían ventanas temporales y no instantes concretos, ya que el usuario debería tener tiempo para llevar a cabo sus acciones (selección, rechazo, visionado, etc.) antes de actualizar los índices y obtener los resultados, que comentamos en los tres escenarios mostrados a continuación.

* **Escenario 1**: En este caso, asumimos que el usuario selecciona todos los programas relativos a la clase C en la primera selección (C1), ocupando además estos programas las primeras posiciones de la recomendación ofrecida. Este mismo comportamiento se repetirá en las restantes elecciones, de modo que el índice de relevancia es máximo y de valor 1.

* **Escenario 2**: Ahora el usuario elige todas las instancias de la clase C contenidas en la recomendación, sin embargo no ve los contenidos elegidos en su totalidad. De esta forma, el índice IOV reduce el valor asignado a DOI y, por consiguiente, el valor de la relevancia asignado a esta clase será menor que 1. En la Fig. 5 se aprecia que la relevancia se incrementa después de cada selección del espectador. Esto se debe a que el usuario ve cada vez una mayor proporción de los programas elegidos (el tiempo de visionado es cada vez mayor). Por esta razón, el sistema continúa sugiriendo instancias de la clase C, ya que ésta parece ser de interés para el usuario.

* **Escenario 3**: El usuario selecciona primero algunas instancias de la clase C (en C1), y rechaza los restantes. Así, la relevancia de dicha clase alcanza un nivel medio. En la siguiente recomendación, AVATAR sugiere de nuevo instancias de esta clase y ahora el usuario rechaza uno de los programas sugeridos, causando un decremento en este índice. Sin embargo, C continúa perteneciendo a las preferencias positivas de este espectador y, por tanto AVATAR sugiere otra vez programas perteneciente a esta clase. Debido a que en esta ocasión continúa el rechazo, la relevancia se sigue reduciendo, alcanzando niveles negativos y quedando fuera de las sugerencias presentadas.

Para concluir esta sección relacionada con la actualización de los perfiles de usuario, destacar que los índices asignados a una clase deben ser propagados a través de la jerarquía de clases definida en el *perfil-ontología* del usuario. Una posibilidad sería propagar a cada superclase el promedio de los valores relativos a cada una de las subclases que dependen de ésta, de modo que el sistema puede disponer de una medida global de interés para cada una de las categorías de programas de TV definidas en su perfil.

4. Conclusiones

En este artículo hemos presentado una técnica de modelado de usuarios en el contexto del sistema AVATAR, un recomendador de contenidos de TV personalizados. Dado que AVATAR está basado en un enfoque de razonamiento semántico sobre las descripciones de los programas de TV y sobre las preferencias de los espectadores, es necesario recurrir a algún mecanismo, que permita una representación del conocimiento del dominio de la TV. A tal efecto, nuestro enfoque propone extender el uso de tecnologías propias de la Web Semántica al contexto de la TV. Así, hemos implementado una ontología —conforme al lenguaje OWL— en la que se ha establecido una jerarquía de clases y propiedades, conceptualizando nuestro dominio de aplicación, y en la que también se han declarado instancias concretas de dichas clases. Esta base de conocimiento será necesaria para llevar a cabo procesos inferenciales, que permitan descubrir relaciones semánticas entre las instancias definidas en la ontología.

Nuestra propuesta de modelado de usuarios se basa en reutilizar la base de conocimiento disponible en el sistema a la hora de representar las preferencias e intereses de los espectadores de TV. Así, para cada perfil de usuario será necesario construir una ontología OWL de forma incremental, a medida que el sistema disponga de información acerca del comportamiento de visionado de los usuarios. Dicho comportamiento será un compendio de varios elementos, entre los que se incluyen la información inferida de los programas elegidos o rechazados por el espectador, ante las sugerencias ofrecidas, y las preferencias declaradas explícitamente.

Además de la reutilización de la jerarquía de clases y propiedades definidas en la ontología de TV del sistema, AVATAR maneja una serie de índices para mantener permanentemente actualizados los perfiles de los usuarios, elemento que sin duda, es clave para asegurar recomendaciones personalizadas de alta calidad. Dichos índices tendrán en cuenta aspectos relevantes en un entorno recomendativo personalizado, como (i) el interés del usuario en las diferentes clases definidas en su perfil-ontología, así como en las instancias de las mismas (índice *index of interest*); (ii) la cuantificación del éxito o fracaso del sistema en recomendaciones pasadas (índice *confidence*); (iii) la proporción entre el tiempo que un usuario ve un programa y la duración total del mismo (índice *index of viewing*), entre otros.

En relación a los índices que hemos nombrado antes, nuestra propuesta también contempla la posibilidad de que el usuario pueda modificar directamente su perfil, accediendo a cada clase declarado en la ontología que lo representa, y a sus instancias, y declarando explícitamente su interés en las mismas. Este enfoque favorecerá una rápida corrección de posibles errores que el sistema pueda introducir a la hora de representar el interés de los usuarios. Además también permitirá acelerar la convergencia entre las preferencias reales del espectador, y las inferidas por el recomendador.

Nuestra propuesta de modelado es especialmente útil a la hora de aplicar una técnica de uso extendido en el contexto de los sistemas recomendadores, esto es, el filtrado colaborativo. Para poder recomendar programas a un usuario dado, que hayan sido de interés para otros espectadores con preferencias similares a las suyas, se hace necesario recurrir a mecanismos que permitan establecer una comparativa entre sus perfiles. La aplicación de ontologías en el modelado de usuarios permitirá aplicar un enfoque de filtrado colaborativo enriquecido, en el que se incluyen capacidades de inferencia semántica. Esto último permite ir más alla de una simple comparación sintáctica entre los contenidos de los perfiles enfrentados, empleando el conocimiento representado en los perfiles y en la ontología OWL.

Referencias

1. Blanco Y. et al. A Multi-Agent Open Architecture for a TV Recommender System: A Case Study using a Bayesian Strategy. In *Proc. of the IEEE Sixth International Symposium on Multimedia Software Engineering*, 2004.
2. Blanco Y. et al. AVATAR: An advanced Multi-Agent Recommender System of Personalized TV Contents by Semantic Reasoning. In *Proc. of 5th International Conference on Web Information Systems Engineering*, volume 3306 of *Lectures Notes in Computer Science*, pages 415–421. Springer-Verlag, 2004.
3. Burke R. Integrating Knowledge-based and Collaborative Filtering in Recommender Systems. In *Proceedings of the Workshop on AI and Electronic Commerce (AAAI)*, 1999.
4. Burke R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
5. Daconta M. et al. *The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management*. John Wiley & Sons, 2003.
6. Geroimenko V. et al. *Visualizing the Semantic Web*. Springer Verlag, 2003.
7. Johansson P. *User Modeling in Dialog Systems*. 2002.
8. Kobsa A. Utilizing Knowledge: The Components of the SB-ONE Knowledge Representation Workbench. In *J. Sowa, ed.: Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann, 1991.
9. Masthoff J. Selecting News to Suit a Group of Criteria: An Exploration. In *Proc. of the 4th Workshop on Personalization in Future TV - Methods, Technologies, Applications for Personalized TV*, 2004.
10. Middleton S. *Capturing Knowledge of User Preferences with Recommender Systems*. PhD thesis, Department of Electronic and Computer Science. University of Southampton., 2003.
11. Resnick P. et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of Conference on Computer Supported Cooperative Work*, 1994.
12. Ricci F. et al. ITR: a Case-Based Travel Advisory System. In *Proceedings of the 6th European Conference on Case Based Reasoning*, 2002.
13. Rich E. *Building and exploiting user model*. PhD thesis, Department of Computer Science. Carnegie-Mellon University, Pittsburgh, 1979.
14. Rich E. User modelling via Stereotypes. *Cognitive Science 3*, pages 329–354, 1979.
15. Rocha C. et al. A hybrid approach for searching in the semantic web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 374–383. ACM Press, 2004.
16. Stock O. et al. Human-computer interaction through natural language and hypermedia in AlFresco. *ACM SIGCHI Bulletin*, 28(3):102–107, 1996.