

Prácticas 7-8

Inteligencia Artificial

Autores (Grupo 5):

José Javier Cortés Tejada

Pedro David González Vázquez

1. Descripción de la ontología

La ontología desarrollada en la práctica se compone de 4 clases principales, representando aplicaciones, desarrolladores de aplicaciones, los sistemas operativos de las mismas, así como usuarios a los cuales les serán realizadas las recomendaciones.

En cuanto a la clase *App*, esta se compone de los siguientes slots:

- *developer*: instancia de la clase *Developer* la cual indica el desarrollador de dicha aplicación.
- *download*: número de descargas de la aplicación.
- *name*: nombre de la aplicación.
- *operative_system*: instancia de la clase *Operative_system* la cual indica el sistema operativo para el que ha sido desarrollada dicha aplicación.
- *pegi*: este slot hace referencia al sistema de clasificación por edades establecido por Información Paneuropea sobre Juegos (PEGI), de tal manera que determina la edad mínima que los usuarios han de tener para usar dicha aplicación.
- *price*: precio de la aplicación.
- *score*: *rating* o puntuación de la aplicación, establecido entre los valores 0 y 5, como mínimo y máximo respectivamente.
- *type*: slot de tipo *symbol* el cual determina que nivel de la jerarquía de aplicaciones ocupa cada instancia de dicha clase, es decir, si tenemos una instancia de la clase *App* con *type*=*Action*, esta aplicación hereda tanto de la clase *App* como de la clase *Game*.

Esta clase actúa como superclase de toda la jerarquía de aplicaciones, de manera que en niveles inferiores tenemos varias clases (todas ellas heredan de forma directa de la clase *App*) como *Travel*, *Video* o *Social_network* que no tienen más clases en niveles inferiores u otras como *Book*, *Game* o *Music*, las cuales sí tienen un tercer nivel de jerarquía por debajo.

Por ejemplo, la clase *Game* se subdivide a su vez en varias subclases como *Action* o *Adventure*, de tal manera que cada instancia de estas clases será a su vez instancia de la superclase *Game* (ambas heredan de ella) y también lo será de la clase *App* (*Game* hereda de ella).

En cuanto a la clase *Operative_system*, solo consta del slot *name* (el mismo que la clase *App*) para indicar el nombre del sistema operativo, y el slot *version* el cual indica la versión del sistema operativo, de manera que podemos tener instancias de dicha clase con el mismo nombre pero con versiones distintas.

Respecto de la clase *Developer*, también tenemos el slot *name* (ya mencionado anteriormente), el cual indica el nombre de la desarrolladora, y un slot de cardinalidad múltiple, *develops*, instancia de la clase *App* donde cada desarrollador guarda las apps que ha creado

Por último, tenemos la clase *User* la cual organiza la información del usuario que vamos a necesitar para generar sus recomendaciones. Disponemos de los siguientes slots:

- *name*: indica el nombre del usuario (no es relevante para las recomendaciones, pero si no lo ponemos queda raro).
- *device*: instancia de la clase *Operative_system* que determina el dispositivo para el que van a realizarse las recomendaciones (es necesario saber su sistema operativo así como la versión en la que está con el fin de no recomendar aplicaciones que el usuario no va a poder usar).
- *age*: indica la edad del usuario.
- *pleasure*: slot de cardinalidad múltiple que indica las categorías de interés para el usuario.
- *recom*: slot de cardinalidad múltiple e instancia de la clase *App* el cual almacena las recomendaciones generadas por el sistema para dicho usuario.
- *wallet*: indica el presupuesto del que dispone el usuario para gastar en aplicaciones, el cual viene por defecto a 0.

Consultas

query1

Buscamos aplicaciones gratuitas que tienen un *rating* por valor de 5. Para ello, aplicamos los filtros que aparecen en la siguiente figura, es decir, filtramos por precio=0.0 y además score=5.0, todo ello sobre instancias de la clase App.

Query: query1

Query: query1 ☒ Match All ☐ Match Any

Class	Slot		Float
App	price	is	0.0
App	score	is	5.0

More Fewer Clear

OK Cancel

query2

Buscamos aplicaciones de menos de 5 € desarrolladas por Disney para niños de cualquier edad. En este caso aplicamos los filtros de la imagen inferior sobre la clase App. Filtramos

Query: query2

Query: query2 ☒ Match All ☐ Match Any

Class	Slot		Float
App	price	is less than	5.0
App	developer	contains	Disney
App	pegi	is	0

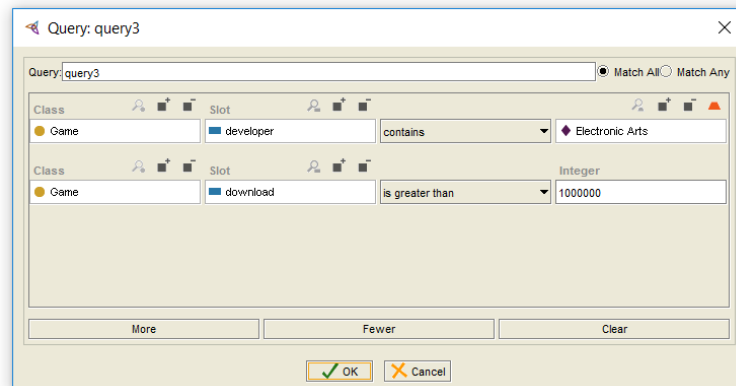
More Fewer Clear

OK Cancel

las aplicaciones por precio de manera que este sea menor que 5.0, que tenga un pegi=0 para que pueda ser usada por cualquier usuario y además filtramos de nuevo para que solo salgan aplicaciones que tienen a Disney como desarrollador.

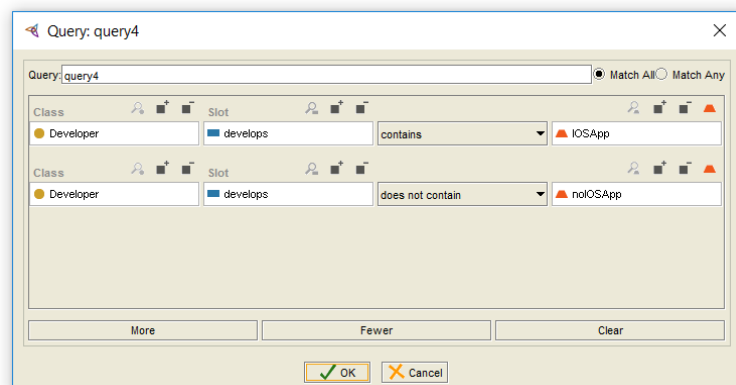
query3

En esta consulta buscamos juegos desarrollados por Electronic Arts que tienen más de un millón de descargar. Para ello filtramos directamente sobre la clase Game que hereda de App, pues solo nos interesan los juegos, y seleccionamos aquellos que tienen como Developer a dicha compañía y además su slot download debe ser mayor que un millón, como muestra la siguiente imagen.



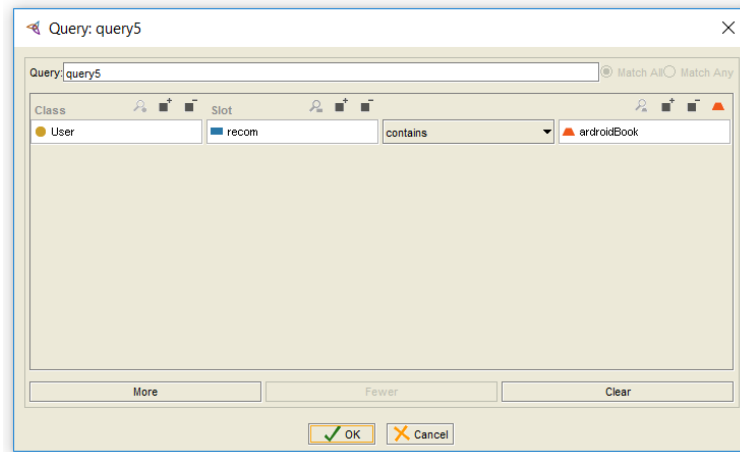
query4

En esta consulta se nos piden los desarrolladores que tienen aplicaciones de viajes solo para iOS. Para ello aplicamos los filtros sobre la clase Developer, de tal manera que nos quedamos con los desarrolladores que tienen aplicaciones en iOS (lo realizamos con la query IOSApp, que busca app desarrolladas para iOS) y desechamos aquellos con app creadas para otro SO (esto lo lleva a cabo la query noIOSApp, que filtra los developers con apps para otro SO que no sea iOS).



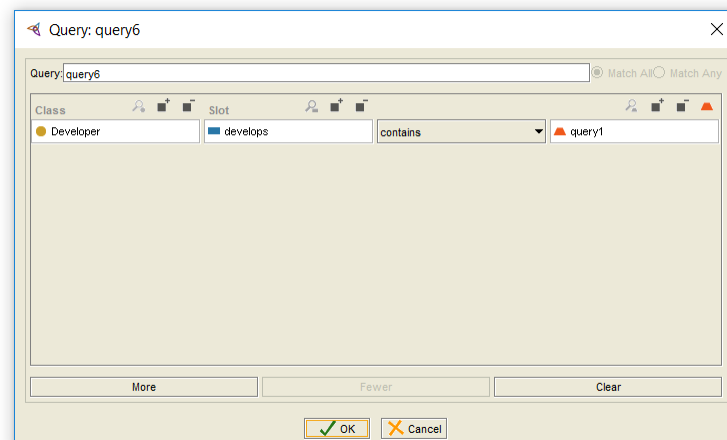
query5

Buscamos usuarios a los que se han recomendado libros del SO Android. Para ello, seleccionamos a los usuarios cuyo slot *recom* contiene algún resultado de la query *androidBook*, que busca apps de libros desarrolladas para el SO Android.



query6

Buscamos desarrolladores con aplicaciones gratuitas con un *rating* igual a 5. En este caso nos valemos de la query1 que busca este tipo de apps y basta con buscar desarrolladores cuyas apps desarrolladas coincidan con el resultado de la query1, como muestra la siguiente imagen.



Código Jess

El archivo con código jess se divide en dos secciones, una primera parte donde tenemos los *deffact* que introducen datos en el sistema sobre aplicaciones, desarrolladores y sistemas operativos y una segunda parte donde se realizan las recomendaciones.

De la primera parte solo interesa comentar un par de reglas, pues la mayoría son simplemente *deffact* que crean instancias en *protégé*, luego nos centramos en las reglas *load-app-developer* y *set-game-action-instance* ya que son las únicas que suscitan interés en este apartado. En ella rellenamos el *multislot* de los desarrolladores donde tenemos las apps que han creado y a la vez hacemos que esas apps rellenen su slot *developer* con el correspondiente desarrollador. Para ello tomamos la información almacenada en el *jess template* *Jess_app* y tomamos las apps y los desarrolladores que coincidan con los campos de dicho template para luego usar *slot-set*, con el que hacemos que ?DEV pase a ser el desarrollador de la aplicación ?APP y también *slot-insert* con el que insertamos la app ?APP en la lista de aplicaciones creadas por el desarrollador ?DEV.

La regla *set-game-action-instance* es el modelo que siguen el resto de reglas usadas para asignar a una app un nivel en la jerarquía. Para ello nos valemos del slot *type* que nos indica a que clases pertenecen las app con ese valor, en este caso nos quedamos con las aplicaciones que sean juegos de acción, de manera que las instancias que tengan este valor en dicho slot serán añadidas tanto a la lista de instancias de la clase *Game* y de la clase *Action*, pues ésta segunda hereda de la primera (se omite la explicación de varias reglas que son idénticas en cuando al funcionamiento, ya que se ve a simple vista que hacen lo mismo pero sobre otras clases).

De cara a la segunda parte, nos vamos a centrar solo en una regla (tomamos la regla *recom-comic-user*, que recomienda comic a los usuarios, aunque podríamos aplicar la explicación a cualquier otra) que define el esquema de las reglas usadas para las recomendaciones, pues todas son similares pero aplicadas a distintas clases. En este caso estamos recomendando aplicaciones a los usuarios, para ello tomamos dichos usuarios, las aplicaciones que vamos a recomendar y que además estén catalogadas como *Comic* (en este caso pues estamos en la regla que recomienda comic). Además filtramos las apps para eliminar las que no nos interesen, bien por que no cumplen la restricción de edad del usuario (la edad del usuario es menor que el *pegi* de la app) o por que el usuario no se la puede permitir (el valor del campos *wallet* es menor que el valor del slot *price* de la app), y por último miramos que dicha app no se haya recomendado anteriormente haciendo *(test (not (member?x?recom)))* para evitar repeticiones.

Cambios respecto a la práctica 5

Respecto de la práctica 5 hemos añadido algunos datos que han sido requeridos para responder a las cuestiones presentadas en el enunciado de la práctica 7, pues en aquella práctica no fueron necesarios para el desarrollo.

Además hemos realizado algunos cambios respecto a la jerarquía de clases, para ello hemos suprimido algunas clases de la clasificación y hemos añadido otras con temática diferente, aunque la funcionalidad es la misma. También hemos añadido un tercer nivel de profundidad a las clases, para tener una jerarquía más extensa.

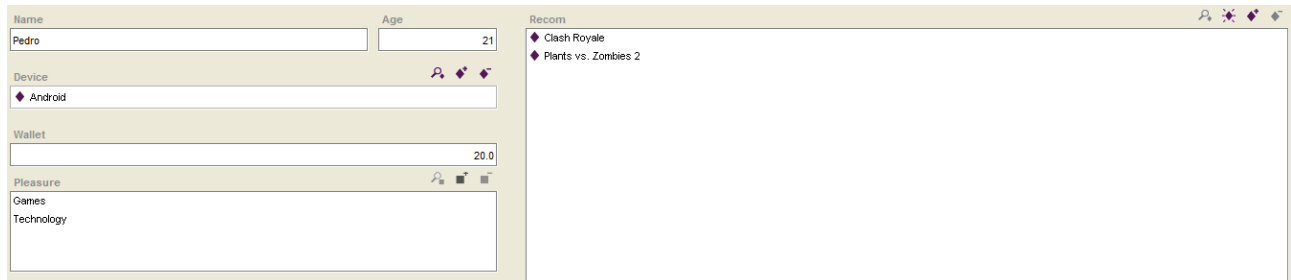
Ejemplos de ejecución

Para ver el funcionamiento del sistema hemos recomendado aplicaciones para dos usuarios con distintos gustos, nivel económico y sistemas operativos. El primer ejemplo corresponde a la siguiente imagen, donde hemos recomendado a un usuario que puede gastar como máximo 60,5€ en aplicaciones, que usa un dispositivo con SO iOS versión 10 y que además está interesado en Viajes, Películas y Música. El resultado obtenido han sido en su mayoría aplicaciones recomendadas en base a la preferencia del usuario por los Viajes, principalmente porque están desarrolladas para su sistema operativo, pues todas ellas son libros de distintas categorías, aunque el sistema hubiese recomendado también aplicaciones relacionada con películas, vídeo o música, siempre y cuando estas hubiesen sido desarrolladas para su sistema operativo.

The screenshot shows a user profile and a list of recommended applications. The profile section on the left includes fields for Name, Age, Device, Wallet, and Pleasure. The recommended applications are listed on the right.

Name	Age	Device	Wallet	Pleasure	Recom
Javi	21	iOS	60.5	Film Music Travel	<ul style="list-style-type: none">Benjamin FranklinBookiCloud BetaSteve JobsAutomator para OS XThe Face of BritainSo the Echo

En el segundo ejemplo, tenemos el siguiente resultado donde hemos recomendado a un usuario que nos ha proporcionado menos información pues de él solo sabemos que puede permitirse aplicaciones por debajo de 20 €, usa un dispositivo con SO Android y está interesado tanto en Juegos como en Tecnología. En este caso solo se han recomendado dos aplicaciones de juegos, las cuales han sido desarrolladas para su SO y son aptas para la versión del mismo.



The screenshot shows a web interface with a light beige background. On the left, there is a user profile section with the following fields: 'Name' (Pedro), 'Age' (21), 'Device' (Android), 'Wallet' (20.0), and 'Pleasure' (Games, Technology). On the right, there is a 'Recom' (Recommendations) section displaying two game titles: 'Clash Royale' and 'Plants vs. Zombies 2'. The interface includes standard web form elements like text boxes and dropdown menus, as well as small icons for user avatars and settings.

Recursos y fuentes

Hemos usado las siguientes fuentes para el desarrollo de la práctica:

- https://protegewiki.stanford.edu/wiki/Main_Page
- <http://www.jessrules.com/jess/>

además de varios post en <https://es.stackoverflow.com/> sobre el manejo de Jess con Protégé y la documentación proporcionada en el guión de la práctica.