IA-Practica 2

Sistemas basados en el conocimiento

Grupo: Sergi Avila Sanguesa, Francisco Sánchez Costas, Raul Montoya Pérez

ÍNDICE

1. Identificación del problema

1.1 El problema

En esta práctica se nos propone crear un sistema capaz de recomendar a una persona un subconjunto de viviendas que podrían adecuarse a sus gustos y preferencias (ya sea por los servicios cercanos, la iluminación natural del piso, la existencia de piscina, …) de entre todas las disponibles en una ciudad.

Para ello, necesitamos obtener datos del usuario y a partir de estos, junto con nuestros propios datos, otorgar al usuario una respuesta acorde a las características descritas para el piso ideal que se busca.

Para poder llevar a cabo esta práctica podemos observar un conjunto de características que identifican a las viviendas y que son relevantes para elegir una. Estas características son:

- **Precio mensual en euros**
- **Localización:** Una vivienda está localizada en unas coordenadas (x,y) determinadas
- **Superfície habitable**
- **Cantidad de dormitorios y tipo:** Si los dormitorios son dobles o simples
- **La inclusión de los muebles y/o electrodomésticos**
- **Tipo de vivienda:** Si es una vivienda unifamiliar, un piso o un dúplex. En caso de ser un piso o un dúplex además se indica su altura.
- **Servicios:** Los servicios cercanos o a una distancia razonable. Son obviamente un factor de peso considerable a la hora de establecer qué pisos son más adecuados para una persona.
- **Otras características:** Así como la iluminación natural, si el piso posee calefacción o aire acondicionado, entre otras muchas características extra.

1.2 Motivación para usar un Sistema Basado en el Conocimiento

Lo que se nos pide en concreto es desarrollar una aplicación capaz de recoger los gustos relacionados con una vivienda de un usuario y recomendar un conjunto de viviendas que podrían ser del agrado del usuario.

Usar un sistema basado en el conocimiento no es una elección nuestra, sino que se indica en la práctica. Aun así, estamos de acuerdo con que es la mejor forma de resolver el problema.

Para justificar la necesidad de un SBC, empecemos por plantear un problema como uno de búsqueda heurística como en la práctica anterior. Estos se basaban en una búsqueda por un espacio de estados o soluciones, y se iban moviendo a través del espacio aplicando operadores y evaluando el estado con un heuristico.

Buscar en un espacio donde cada vivienda es un estado y moverse cambiando la vivienda elegida podría llegar a ser viable, pero el gran problema de esos metodos de resolucion es la función heurística.

Para poder resolver nuestro problema deberíamos poder condensar toda la información (del usuario y preferencias, características de la vivienda, servicios cercanos, etc) en una sola función matemática. En nuestra práctica podría llegar a ser posible, pues es una versión extremadamente simplificada de la realidad. Sin embargo, a la que lo complicásemos un poco ya no sería viable.

Un sistema basado en el conocimiento es capaz de usar el conocimiento del dominio para solucionar el problema haciendo inferencia sobre la información de la cual dispone. Además, se pueden ampliar fácilmente añadiendo más conocimiento.
La desventaja de los SBC respecto a la búsqueda, es que son más complejos y costosos de desarrollar, por lo que el problema ha de ser suficientemente complejo para justificar su desarrollo.

Volviendo al problema de las viviendas, hemos de obtener información del usuario, y junto con los conocimientos de nuestro sistema hacer una recomendación. Necesitamos razonar

a partir de la información en un entorno con mucha información, por lo tanto, un SBC parece ideal para este problema. Además, disponemos de las fuentes de conocimiento suficientes

1.3 Fuentes de conocimiento

Una fuente de conocimiento es aquella información en que se basa un SBC y a partir de la cual el SBC aprende la forma en la que tiene que actuar.

Para el problema planteado en esta práctica, una fuente de conocimiento importante ha sido el enunciado, que ya nos indica posibles características de las viviendas, la ciudad, usuarios… así como reglas para razonar.

También hemos tenido que consultar en Internet algunas páginas web de inmobiliarias para informarnos sobre las características y preferencias de un usuario que usan sistemas actuales para recomendar viviendas, así como el razonamiento que hacen para recomendarnos un piso.

Para el conocimiento sobre la ciudad y las viviendas disponibles, nos ha sido necesario construir una base de datos con los campos que describen cada vivienda, como por ejemplo, si tiene piscina, el número de habitaciones simples y dobles, el precio, la superficie… y también para los servicios con los campos que lo representan, los cuales son el nombre del servicio y sus coordenadas.

No obstante, la fuente de conocimiento esencial para el programa es el mismo usuario que busca una vivienda, ya que al principio de la sesión se le hacen una serie de preguntas para saber sus caracteristicas, sus restricciones y sus preferencias, las cuales, ayudaran a filtrar las viviendas que más se adecuan a dicho usuario.

1.4 Objetivos

Los objetivos que debe cumplir nuestro SBC son los siguientes:

- Obtener toda la información necesaria del usuario sobre sus características personales y preferencias y gustos que nos puedan ayudar a hacer una recomendación personalizada.

- Utilizar los conocimientos para filtrar las viviendas y quedarnos con las más adecuadas para el usuario.

- Valorar las viviendas según sus características y finalmente poder separarlas en 3 categorías: Parcialmente adecuada, adecuada y muy recomendable.

- Mostrar la información al usuario de forma que se entienda fácilmente, mostrando las viviendas por categorias, asi como justificaciones del porqué el sistema la ha elegido.

2. Conceptualización

Partimos de una ciudad diseñada por nosotros con una base de datos compuesta por viviendas con sus respectivos atributos (identificador de vivienda, altura de la vivienda (piso), amueblada, coordenadas, número de dormitorios simples y dobles, electrodomesticos, garaje, piscina, precio mensual, superficie, terraza, vistas, sol por la mañana o por la tarde, tipo de vivienda, aire acondicionado, balcón, calefacción, mascotas, seguridad, reformada, ascensor, numero de baños, TV, WIFI, accesible, fumadores, portero y los servicios que tiene cerca y a media distancia), servicios de la ciudad con los atributos nombre del servicio y sus coordenadas y finalmente tenemos todas las coordenadas de la ciudad donde hay algun servicio o vivienda.

Para cada alquiler, se quiere saber el nombre, su edad, su tipología (es decir, si es un solo individuo, una familia, una pareja o un grupo) y el número de personas que solicitan el alquiler. También, queremos saber si posee vehículo y si trabaja o estudia en la ciudad (en ese caso pediremos las coordenadas de dicho trabajo o zona de estudio). Con estos datos podemos deducir algunas características de las viviendas que pueden ser favorables para el/los solicitantes y podemos ofrecer los alquileres que más se adecuan a estos.

Además obtendremos las restricciones y/o preferencias del usuario respecto a las características de la vivienda, que nos permitirá clasificar los alquileres en tres grupos: alquiler parcialmente adecuado, si no se cumplen 1 o 2 restricciones marcadas por el usuario; adecuado, si cumplen todos los requisitos; y muy recomendable si además de cumplirse todos los requisitos tiene varias caracteristicas extra que pueden favorecer al usuario.

2.1 Conceptualización del problema

Tipología del cliente

Para hacer una division del tipo de cliente y la cantidad de gente para la que buscamos vivienda, separaremos los solicitantes en los siguientes grupos:

- Pareja (2 personas)
- Individual (1 persona)

- Familia (más de 2 personas, con hijos, padres u otros familiares)
- Grupo (mas de 1 persona, por ejemplo, estudiantes o compañeros de piso)

Esta separación nos permite determinar cuántas personas van a vivir en piso, cuantos dormitorios dobles necesitan, así como dar mas importancia a la cercanía de unos servicios u otros.

Calidad de la vivienda según el perfil

Para poder decidir que una vivienda es mejor que otra según el perfil del usuario tenemos en cuenta características de dos tipos:

- Preferencias del usuario: Estas son las más importantes. Si un usuario pide explicitamente una característica o un servicio que quiere cerca, hay que intentar buscar un piso que se adapte.
- Perfil del usuario: A través de la información que tenemos del usuario, aunque no esté directamente relacionada con el piso, podemos dar un prioridad a los pisos que deducimos que les gustaran (por ejemplo, si son una familia quizás quieran una piscina comunitaria para los niños).

Calidad inherente de la vivienda

La elección de una vivienda no solo depende del usuario, hay viviendas que por si solas son mejores que otras, por ejemplo: es mejor una casa que un dúplex y ambos son mejores que un piso, mejor si da el sol siempre, o si tiene muchos servicios cerca, si hay cerca zonas verdes, etc.

2.2 Elementos del dominio

Características de una vivienda:

- Identificador.
- Altura de la vivienda (planta baja,entresuelo,primero...ático).
- Si tiene aire acondicionado.
- Si está amueblada.
- Si tiene balcon.
- Si tiene calefacción.
- Coordenadas.
- Número de dormitorios simples.
- Número de dormitorios dobles.

- Número de baños.
- Si tiene electrodomésticos.
- Si tiene garaje.
- Si es apta para mascotas.
- Si tiene piscina.
- Precio mensual.
- Servicios cerca.
- Servicios a media distancia.
- Si da el sol por la mañana.
- Si da el sol por la tarde.
- Superficie.
- Si tiene terraza.
- Tipo vivienda (piso, vivienda unifamiliar o dúplex).
- Si tiene seguridad.
- Si ha sido reformada recientemente.
- Si tiene ascensor.
- Si tiene TV.
- Si tiene WIFI.
- Si es accesibilidad para todo tipo de personas..
- Si es apta para fumadores.
- Si tiene portero.

Características de un servicio:

- Nombre del servicio.
- Coordenadas.

Características de una coordenada:

- Valor X.
- Valor Y.

2.3 División en subproblemas

Obtener información del usuario

Lo primero que ha de hacer nuestro sistema es obtener el conocimiento sobre el usuario, primero la personal (nombre para un trato más personal, edad, tipología...) y después sobre sus preferencias (servicios que quiere cerca, preferencias de características de la vivienda…).

Determinar las distancias
Antes de empezar a razonar sobre la información que hemos obtenido del usuario, hace falta completar la de nuestro sistema. Como trabajamos con un sistema de coordenadas, para saber que servicios estan cerca de cada vivienda es necesario hacer el cálculo de las distancias y asignar a cada vivienda los servicios que tiene cerca y a media distancia.

Hacer un filtro inicial de viviendas
Algunas de las características del usuario ya eliminan totalmente algunas viviendas, como puede ser el precio y el número de dormitorios. Un filtro inicial ayuda a dejar de tener en cuenta algunas viviendas y por lo tanto, tener un sistema más eficiente.

Valoración de las viviendas
Una vez hecha la criba, hay que valorar las viviendas restantes: se puntúan en función de si cumplen o no las preferencias del usuario, y se añaden comentarios con la justificación de ese aumento o reducción de puntos.

Organización de la solución
Teniendo ya las viviendas puntuadas, tenemos que organizarlas en los tres grupos mencionados anteriormente, así como hacer un segundo filtro y eliminar aquellas viviendas que no cumplan más de dos de las preferencias introducidas por el usuario.

Mostrar la solución
El paso final es mostrar la solución al usuario de forma que sea fácilmente comprensible, escribiendo todas las características de la vivienda y además, una explicación de las características extras que pueden ser beneficiosas y de las que al contrario, pueden ser desfavorables.

3. Formalización

3.1 Desarrollo de la ontología

Para el desarrollo de la ontología, empezamos separando el problema en sus 3 conceptos principales:

- **Usuario:** El concepto de usuario se separó rápidamente en dos mas: usuario (información personal) y preferencias usuario, para tener la información más estructurada. Finalmente, como veremos en el siguiente apartado, acabaron siendo templates por facilidad de implementación.

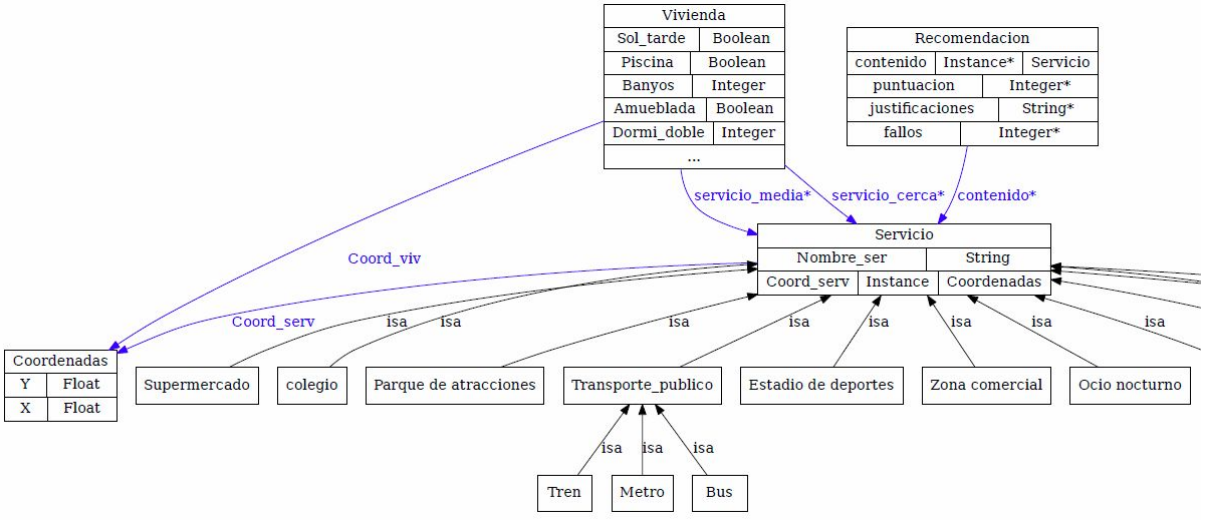
- **Vivienda:** La clase vivienda era la más obvia y ha estado desde el principio del desarrollo, pero se ha ido expandiendo.

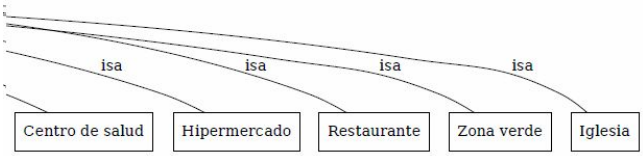
- **Ciudad:** Desde el principio sabíamos que hacía falta representar la ciudad en la ontología. Tras pensar en qué aspectos de la ciudad nos interesaban, reducimos ciudad a servicio, pues era lo único que nos servia. Además, con nuestro sistema de coordenadas arbitrario, no nos hacia falta guardar el tamaño de la ciudad.

Ya durante la etapa de implementación, hizo falta añadir el concepto de recomendación, que nos permite guardar la información sobre la vivienda que se genera durante la ejecución, como la puntuación obtenida y la justificación de esta. Además, separamos el concepto de coordenadas, pues era informacion común en servicios y viviendas.

3.2 Documentación

Nuestra ontología es la siguiente:





La clase recomendación se añadió fuera de Protégé

Vivienda

- **Aire:** Si tiene aire acondicionado.
- **Altura_piso:** Altura del piso (planta baja,entresuelo,primero...ático).
- **Balcon:** Si tiene balcón.
- **Calefaccion:** Si tiene calefacción.
- **Coord_viv:** Instancia de coordenadas.
- **Dormi_doble:** Número de dormitorios dobles.
- **Dorm_simple:** Número de dormitorios simples.
- **Banyos:** Número de baños.
- **Electrodomesticos:** Si ya tiene electrodomésticos.
- **Garaje:** Si dispone de garaje.
- **Id:** Identificador de la vivienda.
- **Mascotas:** Si permite mascotas.
- **Piscina:** Si dispone de piscina.
- **Precio_mensual**: Coste mensual del alquiler.
- **servicio_cerca:** Instancias de servicios a menos de 500m.
- **servicio_media:** Instancias de servicios a menos de 1km.
- **Sol_man:** Si hay sol por la mañana.
- **Sol_tarde:** Si hay sol por la tarde.
- **Superficie:** Superfície del piso en m^2.
- **Terraza:** Si tiene terraza.
- **Tipo:** Si es piso, vivienda unifamiliar o dúplex.
- **vistas:** Si tiene vistas**.**
- **Seguridad:** Si tiene seguridad.
- **Reformada:** Si ha sido reformada recientemente.
- **Ascensor:** Si tiene ascensor.
- **TV:** Si tiene TV.
- **WIFI:** Si tiene WIFI.
- **Accesible:** Si es accesible para todo tipo de personas.

- **Fumadores:** Si es apta para fumadores.
- **Portero:** Si tiene portero.

Coordenadas

- **X:** coordenada X.
- **Y:** coordenada Y.

Servicio

- **Coord_viv:** Instancia de coordenadas
- **Nombre_ser:** nombre del servicio

Las siguiente clases no forman parte de Protégé:

Recomendación

- **Contenido:** Instancia de vivienda.
- **Puntuacion:** Puntuacion que damos a la vivienda, 100 por defecto.
- **Justificaciones:** Mensajes que indican porque hemos restado o sumado puntos.
- **Fallos:** Número de diferencias entre lo que la vivienda tiene y lo que el usuario quiere.

Usuario

```
(deftemplate MAIN::Usuario
  (slot nombre (type STRING))
  (slot edad (type INTEGER)(default -1))
  (slot tipo (type SYMBOL)(default desconocido))
  (slot tam_familia_grupo (type INTEGER)(default -1))
  (slot posee_vehiculo (type SYMBOL)(default desconocido))
  (slot coorX (type INTEGER) (default -1))
  (slot coorY (type INTEGER) (default -1))
)
```

- **nombre:** nombre del usuario.
- **edad:** edad del usuario.
- **tipo:** tipología del usuario (pareja,familia individual o grupo).
- **tam_familia_grupo:** número de personas para las que buscamos piso.
- **posee_vehiculo:** si posee vehículo propio.
- **coorX y coorY:** en caso de trabajar o estudiar en la ciudad, las coordenadas del sitio.

Preferencias usuario

```
(deftemplate MAIN::preferencias_usuario
  (slot precio_maximo (type INTEGER)(default -1))
  (slot precio_estricto (type SYMBOL)(default desconocido))
  (slot num_dormitorios_dobles (type INTEGER)(default -1))
  (slot num_banyos (type INTEGER)(default -1))
  (slot precio_minimo (type INTEGER)(default -1))
  (multislot distancia_servicio (type SYMBOL))
  (multislot preferencias_vivienda (type SYMBOL))
)
```

- **precio_maximo:** precio máximo que está dispuesto a pagar el usuario.
- **precio_estricto:** si el usuario podría llegar a pagar un poco más.
- **num_dormitorios_dobles:** Número de dormitorios doble que el usuario necesita.
- **num_banyos:** Número de baños que el usuario quiere
- **precio_minimo:** lo mínimo que tiene pensado gastarse en el piso.
- **distancia_servicio:** servicios que el usuario quiere tener a menos de 1km.
- **preferencias_vivienda:** otras características de la vivienda que quiere el usuario.

Listas de recomendaciones

```
(deftemplate MAIN::lista-rec-desordenada
  (multislot recomendaciones (type INSTANCE))
)


(deftemplate MAIN::lista-rec-ordenada
  (multislot recomendaciones (type INSTANCE))
)

(deftemplate MAIN::Poco_Recomendables
  (multislot recomendaciones (type INSTANCE))
)

(deftemplate MAIN::Recomendables
  (multislot recomendaciones (type INSTANCE))
)

(deftemplate MAIN::Altamente_Recomendables
  (multislot recomendaciones (type INSTANCE))
)
```

- **lista-rec-desordenada:** primera lista donde se juntan las recomendaciones.
- **lista-rec-ordenada:** lista ordenada para gestionarla mejor.
- **Poco_Recomendables, Recomendables, Altamente_Recomendables:** listas para los 3 grados de recomendación.

3.3 Relación con los subproblemas

La mayoría de clases surgieron al plantear el problema en general, sin embargo, al querer resolver algunos subproblemas tuvimos que añadir clases.

Para hacer el filtro y la valoración de las viviendas, nos hacia falta otra clase para poder añadir información sobre viviendas sin tener que modificar la clase, pues surgia durante la ejecución del problema. Además, no queríamos tener que eliminar las instancias de vivienda. Así surgió la clase recomendacion.

Para organizar la información nos hacia falta juntar y ordenar, y despues dividir las recomendaciones, así que tuvimos que añadir listas de recomendaciones, aunque están implementadas como templates y no como clases para simplificar el código.

3.4 Método de resolución

Este se trata de un problema de análisis, por lo que es adecuado aplicar la clasificación heurística, la cual, consiste en 3 etapas.

Abstracción de los datos
Obtener información del usuario y abstraer cierta información (como si son una pareja, grupo, familia, individuo, si es viejo, joven o de mediana edad, si quiere gastarse mucho dinero o no…)

Asociación heurística

Relacionar los datos abstractos para obtener una solución abstracta. Analizar un parámetro y sumar a la puntuación de la vivienda, con relaciones del tipo “si es solo una persona querrá ocio nocturno”.

Refinamiento de la solución/Adaptación
Convertir la solución abstracta en concreta. Comparar las recomendaciones por la puntuación y mostrar todos los datos de la vivienda. En nuestro caso, no usamos reglas extra en esta fase.

4. Implementación

4.1 Construcción de la ontología

La ontología ha sido generada con Protégé con las características descritas anteriormente. Además, para gestionar algunos aspectos del programa hemos creado clases, como por ejemplo, la clase Recomendación, donde por cada vivienda se guarda la puntuación, las justificaciones y el número de restricciones impuestas por el usuario que no cumple.

También se han creado tres clases más, una para cada categoría de alquiler donde se guardan las viviendas que forman parte de cada uno de los diferentes grupos.

Por otro lado tenemos dos deftemplates para guardar los datos y preferencias del usuario, y cinco más que sirven de estructuras de datos que dan más comodidad a la hora de programar.

Las clases Vivienda, Servicio y Coordenadas ya tienen instancias desde el principio, mientras que el resto de clases y de los deftemplates se instancian dinámicamente.

4.2 Módulos

MAIN
Contiene una serie de funciones para hacer preguntas(útiles para los otros módulos), una función que calcula la distancia entre dos puntos, la regla inicial que inicia la ejecución del sistema, y la regla para fijar los servicios cercanos y a media distancia a las viviendas.

Recopilacion-usuario
Módulo de recopilación de los datos del usuario: nombre, sexo, edad, tipología, tamaño del grupo/família, si tiene vehículo, si estudia/trabaja y la localización de estudio/trabajo. Esta información se recopila a partir de una serie de preguntas, implementadas en el módulo MAIN. También contiene una regla para pasar al módulo recopilacion-preferencias.

Recopilacion-preferencias
Módulo de recopilación de las preferencias y restricciones del usuario: precio máximo y si es estricto o no, número de dormitorios dobles, precio mínimo… Esta información se recopila a partir de una serie de preguntas, implementadas en el módulo MAIN. También contiene una regla para pasar al módulo procesado.

Procesado
Modulo que realiza todo el procesado de la información obtenida del usuario para puntuar las viviendas según sus características y eliminar aquellas que no forman parte de la solución. También contiene una regla para pasar al módulo generacion-sol.

Generacion_sol
Modulo que contiene las reglas para generar las soluciones. Se miran las viviendas con máxima puntuación y se dividen entre los tres grupos de alquiler. También contiene una regla para pasar al módulo mostrar-resultados.

Mostrar_resultados
Este módulo contiene una única regla, la cual, presenta los resultados al solicitante y finaliza la ejecución.

4.3 Prototipos

Hemos trabajado sobre un solo prototipo, el cual hemos ido actualizando a medida que se iba añadiendo nuevo contenido. La estructura de CLIPS utilizada es la siguiente:

Clases e instancias

- Codigo hecho con Protégé para crear clases e instancias.

Definición de defclass

- Recomendación: para las viviendas, su puntuación y justificaciones.

Definición de defmodule

- Recopilacion-usuario: para obtener los datos del usuario.
- Recopilacion-preferencias: para obtener las preferencias y restricciones del usuario.
- Procesado: para filtrar la información obtenida y valorar viviendas.
- Generacion-sol: para generar las soluciones con las viviendas valoradas.
- Mostrar_resultados: para mostrar las recomendaciones al usuario.

Definición de defmessage-handlers

- Imprimir(Vivienda): para imprimir la información de la vivienda.
- Imprimir(Recomendacion): para imprimir una recomendación.

Definición de deftemplates

- Usuario: almacena los datos del usuario.
- Preferencias_usuario: almacena las preferencias y restricciones del usuario.
- lista-rec-desordenada: lista de viviendas desordenada.
- lista-rec-ordenada: lista de viviendas ordenada por puntuación.
- Poco_Recomendables: almacena las viviendas poco recomendables.
- Recomendables: almacena las viviendas recomendables.
- Altamente_Recomendables: almacena las viviendas altamente recomendables.

Definición de deffunction

- Funciones para hacer preguntas al usuario.
- Euclidean: función para calcular la distancia entre dos puntos.
- Max_punt: función para calcular la recomendación que tiene más puntuación.

Definición de defrule

- Todas las reglas del implementadas.

5. Juegos de prueba

Para probar el funcionamiento de nuestro sistema hemos diseñado los siguientes juegos de prueba. Son perfiles inventados pero que pretenden ser realistas y cubrir la mayoría de características que hemos considerado. Los juegos de prueba empezaran con casos que deberían ser relativamente fáciles e irán aumentando de dificultad

Debido a lo largo que es todo el proceso y el texto final en clips, separaremos el copy-paste en otro documento, y aquí nos centraremos en la explicación.

5.1 Compañeros jóvenes

Perfil:
Persona de 21 años, quiere buscar piso con un compañero para independizarse. No disponen de vehículo pero trabajan dentro de la ciudad. Como acaban de empezar, no tienen demasiado dinero, pero entre los 2 pueden juntar 800€ mensuales para el alquiler. Mientras el piso tenga muebles y electrodomésticos, lo básico para vivir, y esté en su presupuesto están contentos. Les gustaría estar cerca del trabajo, pero no les importa estar un poco lejos si el precio es mas barato

Resultado esperado:
Este primer caso es sencillo, apenas tienen exigencias aparte del precio. El sistema debería mostrar los pisos más económicos y que no les supongan un gasto extra (amueblar o electrodomésticos), pues seguramente no vivan ahí mucho tiempo.

Resultado real:
El resultado ha sido el esperado: la criba se ha producido en el precio, pero como a parte de eso las exigencias eran mínimas, el sistema ha encontrado varias ofertas buenas. Además, se indica si el trabajo esta cerca, o si hay bus cerca para llegar.

5.2 Pareja con recien nacido

Perfil:
Pareja de 30 años, acaban de tener un hijo y buscan un piso para criarlo. No descartan tener otro hijo en el futuro. Ambos tienen trabajo fuera de la ciudad y disponen de coche.
Tienen trabajo estable y son de clase media, con unos 1300€ mensuales disponibles, aunque podrian hacer un esfuerzo extra si hay una buena oferta. Les gustaria tener una buena vivienda y menos de 700€ les haría desconfiar. Una piscina comunitaria les haría bastante ilusión. No tienen muchas más preferencias ni les importaria tener que amueblar la casa. Por comodidad, un supermercado y colegio cerca serian muy utiles y, por supuesto, un parque para el niño. Quieren evitar estar cerca de bares.

Resultado esperado/comentario:
Este podría ser el típico caso de pareja y, en principio, lo hemos tenido en cuenta al crear el sistema. Por lo tanto, debería ser capaz de dar resultados bastante adecuados aunque no den demasiada información sobre sus preferencias. En este caso introducimos restricciones sobre el precio mínimo y número de dormitorios dobles, así como alguna limitación en los servicios cercanos.

Resultado real:
El presupuesto y las plazas han limitado mas de lo esperado. La mayoría de viviendas o eran demasiado caras o eran baratas pero para pocas personas. Casi todas las restantes sobrepasaban ligeramente el precio, pero no suficiente para ser eliminadas. Aun con unas preferencias más estrictas de lo esperado, el sistema ha podido dar varias opciones, e incluso encontrar una ideal si no fuera por el precio.

5.3 Soltero medio

Perfil:
Persona de 35 años, le ha surgido un buen trabajo en la ciudad y busca un piso cercano. Tiene un buen presupuesto de 2000€ e incluso podria pagar mas. Quiere un piso con dormitorio doble. Vivir cerca de algún centro de ocio nocturno facilitaría su uso. Quiere un piso soleado que ya tenga muebles y TV, además de una terraza para poder fumar a gusto. Como tiene dinero y es un poco perezoso, tener un restaurante cerca le iría bien. También, un estadio de deportes haría su estancia mucho más entretenida.

Resultado esperado/comentario:
Aqui el dinero no limita demasiado, pero este cliente tiene algunas preferencias extra que iran bien para probar el sistema. Con el dinero sin ser un problema, se debería poder elegir entre la mayoría de viviendas y encontrar varias que cumplan lo que busca. Además, puede ser interesante ver que pasa con información, a priori, contradictoria, como es un soltero buscando dormitorio doble. Aparte de eso, el sistema da buenas recomendaciones y muestra las viviendas con ocio nocturno cerca.

Resultado real:

Debido al presupuesto alto y a solamente necesitar una cama doble, la mayoría de ofertas no se han eliminado. La criba se ha hecho con las eliminando viviendas que no cumplian las preferencias. El sistema acepta cualquier combinación de preferencias, asi que eso no ha sido un problema. Como pasará en lo siguientes casos, agrupar en 3 categorías es complicado, pues ninguna o casi ninguna vivienda cumplirá el 100% de las preferencias cuando el usuario tenga muchas preferencias y restricciones y, además, a la mínima característica extra que tenga una vivienda ya pasara a ser altamente recomendada, por lo que en el grupo de adecuada no habrá prácticamente nada.

5.4 Pareja de ancianos

Perfil:
Pareja de 75 años, quieren buscar un piso para estar relajados. Tienen un estricto presupuesto de 1100€ y no podrían permitirse ningún coste extra en la vivienda. Bajar escaleras puede ser muy pesado para ellos, asi que un ascensor les seria muy util. Quieren un piso soleado y que permita traer a su perro. Además, un baño para cada uno les facilitaría bastante las cosas. Con la edad saben que las visitas al médico son frecuentes, y les iría bien tener un centro de salud cerca. Bus, zonas verdes y supermercados les ayudarían mucho en su dia a dia.

Resultado esperado/comentario:
Este perfil utiliza muchas características que no tuvimos en cuenta hasta el final del proyecto, como el de gente mayor, restricciones en baños y ascensor... Así que es interesante ver como responde nuestro sistema. Es un presupuesto medio con varias restricciones, pero debería encontrar alguna vivienda adecuada.

Resultado real:
Las últimas características añadidas funcionan bien. En este caso el dinero vuelve a ser algo importante, y elimina la mayoría de opciones, aunque deja una buena oferta.

5.5 Familia rica exigente

Perfil:
Familia con hijos, padres y abuelos que buscan un buen sitio donde vivir. El dinero no es un problema para ellos, asi que quieren un buen sitio para los 6 que son, con 2 dormitorios dobles por supuesto. Quieren una vivienda que tenga de todo. Estar cerca de ese nuevo parque de atracciones sería un plus, pero lo importante es tener restaurantes y un centro comercial cerca.

Resultado esperado/comentario:
El dinero es algo que hemos tenido bastante en cuenta al crear el sistema, asi que en esta prueba lo eliminamos como factor totalmente. Queremos ver como reacciona nuestro sistema con tantas preferencias.

Resultado real:
Como era de esperar, la mayoría de viviendas se han eliminado por precio bajo o por no cumplir las exigencias. Aun así ha sobrevivido una.

5.6 Joven estudiante

Perfil:
Persona de 20 años que estudia en una gran ciudad cercana. Con un presupuesto de 700€ sabe que en la ciudad donde estudia no encontraría piso, así que ha venido a buscar a la nuestra. Quiere un piso amueblado, con electrodomesticos, TV y WIFI. Es muy importante que tenga bus y metro cerca, para moverse por las 2 facultades donde estudia.

Resultado esperado/comentario:
Este caso es como el primero pero con más restricciones para ver si el sistema sigue dando una respuesta buena.

Resultado real:
Los resultados han sido buenos, aun con las restricciones extra, el sistema ha encontrado unas pocas ofertas.

5.7 Grupo diverso

Perfil:
Grupo de 4 amigos que quieren vivir juntos en la ciudad. Juntan 2500€ mensuales y entre todos han apuntado las características de la vivienda que quieren: vistas, amueblada y con electrodomésticos, que permita fumadores, con garaje, piscina, terraza, TV, WIFI, calefacción y aire acondicionado. Además, quieren estar cerca de una iglesia, supermercado, restaurante, zona verde, transporte público y algun lugar de ocio nocturno.

Resultado esperado/comentario:
Para el último juego de pruebas vamos a poner muchas preferencias con un presupuesto no demasiado alto para todo lo que pide, a ver si el sistema es capaz de hacer una buena recomendación.

Resultado real:
Como era de esperar, todas las viviendas se han eliminado por no cumplir las preferencias

6. Conclusiones

El problema ha sido dividido y resuelto en subproblemas partiendo de la metodología en cascada a partir de la creación de un sistema basado en el conocimiento.

Nuestro sistema cumple con los objetivos ya que a partir de los datos, preferencias y restricciones introducidos por el usuario, es capaz de generar recomendaciones y dividirlas en tres categorías según las características de cada vivienda respecto a los datos y preferencias que se establecen. Una vez hecho todo esto, se muestran los resultados al usuario con las justificaciones y puntuaciones de cada vivienda de cada grupo.

Finalmente, cabe destacar que en el actual sistema podrían considerarse varias mejoras, como por ejemplo, ajustar mejor las puntuaciones dadas por características extra, es decir, dar más valor a algunas y menos a otras, así como la división entre los tres grupos de recomendación, ya que pocas veces se dara el caso de tener una recomendación que cumpla todas las restricciones pero sin características extras debido a que cualquier mínimo valor que sea levemente mejor que lo que pide el usuario ya se asignan puntos extra a la vivienda.