Universidad Politécnica de Madrid

Máster Universitario en Inteligencia Artificial

Modelos de Razonamiento

Sistema aconsejador inteligente para recomendación de componentes de ordenador

[Grupo F]

Rafael Carlos Gimenez Aguilar

Mikel Moreno Moreno

21 de diciembre de 2021

Índice

1. Introducción y estado del arte **1**

2. Implementación **2**

2.1. Estructura de archivos 2

2.2. Obtención de los datos 3

2.3. Desarrollo del sistema 3

3. Conclusiones **5**

SECCIÓN 1

Introducción y estado del arte

El incremento en la oferta en el mercado de componentes de ordenador ha provocado la desorientación de muchos individuos a la hora de escoger las piezas más idóneas para sus sistemas (también conocido como *builds*). Con el fin de poder ayudar en la toma de decisiones, se presenta el siguiente sistema inteligente.

Dicho sistema es capaz de generar un esquema de componentes para crear el computador con la mínima potencia necesaria para satisfacer las necesidades del usuario. Para ello tiene en cuenta ciertos parámetros como tipo de uso, presupuesto y compatibilidad entre componentes.

En la figura 1 puede observarse como el proyecto se cataloga dentro de la clase *sistema recomendador* puesto que la interacción entre el sistema y el usuario está restringida a unas preguntas y respuestas previamente formuladas.

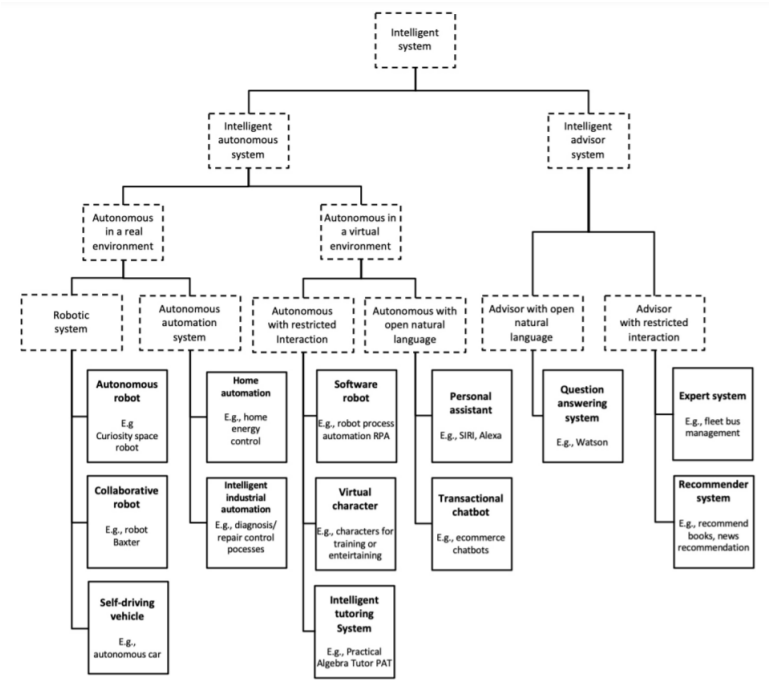

Figura 1: *Taxonomía de los sistemas inteligentes [Molina, 2020]*

Los sistemas de recomendación han sido extensamente utilizados a lo largo de la historia, pero presentan limitaciones para ser utilizados en determinados casos. Una de las limitaciones más importantes es lo que se denomina el *cuello de botella de la adquisición de conocimiento*. Los sistemas expertos utilizan conocimientos heurísticos formalizados con representaciones declarativas (por ejemplo, reglas y marcos) que se almacenan en una base de conocimientos. La creación y el mantenimiento de una base de conocimientos suele requerir un esfuerzo considerable. [Molina, 2020]

Otro problema está relacionado con la forma en que estos sistemas obtienen información sobre el entorno. Por ejemplo, Mycin [Swartout, 1985] obtiene información sobre el paciente utilizando un conjunto prefijado de preguntas y respuestas. Este mecanismo de comunicación puede ser demasiado estrecho y rígido para ser utilizado en situaciones en las que el formato de los datos disponibles es más diverso (por ejemplo, textos no estructurados).

SECCIÓN 2

Implementación

El siguiente apartado pretende mostrar los pasos seguidos para el desarrollo de la solución propuesta.

2.1 Estructura de archivos

```
PCconfig
├── database.pl
├── handle_database.ipynb
├── pcconfig.pl
├── pcpart_scrapper.ipynb
└── Data
```

Figura 2: *Estructura de los ficheros*

A continuación se pasa a describir el contenido de los archivos:

- **database.pl** contiene los datos limpios utilizados por el sistema para realizar las consultas.
- **handle_database.ipynb** contiene un script en python para generar el archivo Database.pl a partir de los datos en bruto encontrados en el fichero Data.
- **pcconfig.pl** contiene el programa principal usado por el usuario para la realizacion de las consultas, obtiene los datos del archivo Database.pl.
- **pcpart_scrapper.ipynb** contiene un script en python para descargar informacion de las paginas web de venta de componentes y almacenarlo en diferentes archivos dentro del fichero Data.
- **Data** fichero donde se almacenan los datos en bruto generados por pcpart_scapper.ipynb.

2.2 Obtención de los datos

Para la creación de la base de datos, se ha hecho uso de un script de python que recopilaba información de las paginas web de venta de componentes, obteniendo así un informe detallado de cada una de las piezas y sus socket. Los enlaces a dichas paginas pueden ser encontrados en el anexo. Los datos en bruto no podían ser utilizados por el sistema, con lo que se planteo una limpieza de los mismos en los cuales se tuvieron en cuenta los siguientes puntos:

1. Eliminar productos descatalogados o dirigidos a servidores.
2. Eliminar campos innecesarios manteniendo únicamente los mostrados en la figura 1.
3. Transformar los valores de los campos al tipo de datos correspondiente:
   a) String a Integer.
   b) String a Float.
4. Convertir el precio de los componentes a Euros.
5. Unir campos relacionados (como nombre del componente y nombre del fabricante).
6. Ordenar las filas por precio descendente.
7. Introducir los datos en el .pl destino.

Esto se ha realizado con todos los componentes mostrados en la figura 1.

2.3 Desarrollo del sistema

Para la realización del sistema se ha hecho uso de un lenguaje de programación lógico e interpretado llamado Prolog. En este tipo de lenguajes de programación la ejecución de las instrucciones sigue un orden secuencial. Los programas en Prolog se componen de cláusulas de Horn que constituyen reglas del tipo "modus ponendo ponens"separados por comas. En Prolog no existen instrucciones de control. Su ejecución se basa en dos conceptos: la unificación y el backtracking. Gracias a la unificación, cada objetivo determina un subconjunto de cláusulas susceptibles de ser ejecutadas llegando a un punto de elección. A continuación se elige la primera opción y sigue ejecutando el programa hasta determinar si el objetivo es verdadero o falso. En caso de ser falso, pasa a deshacer todo lo ejecutado situando el programa en el mismo estado en el que estaba justo antes de llegar al punto de elección (backtracking). Entonces se toma la siguiente poción pendiente y se repite de nuevo el proceso. Todos los objetivos terminan su ejecución bien en éxito ("verdadero"), bien en fracaso ("falso"). El sistema desarrollado sigue el siguiente proceso:

Nombre	Atributo	Descripción
CPU	name	Nombre del chip
	socket	Tipo de conector que utiliza, este atributo sirve para comprobar si es compatible con la Motherboard
	igpu	Targeta grafica intergrada, none en caso de que no contenga
	tdp	Cantidad de energía necesaria para su funcionamiento
	performance	Rendimiento del procesador, este valor se ha discretizado a light, moderate y high
	price	Precio del procesador
GPU	name	Nombre de la tarjeta dado por el fabricante
	model	Modelo de la tarjeta
	tdp	Cantidad de energía necesaria para su funcionamiento
	vram	Cantidad de ram virtual, medida en GB
	price	Precio de la tarjeta en euros
Motherboard	name	Nombre de la placa dado por el fabricante
	socket	Conector, utilizado para comprobar la compatibilidad con el procesador
	form_factor	Tamaño de la placa madre, Micro ATX para ordenadores pequeños y ATX para un tamaño regular
	max_ram	Capacidad de ram maxima, limita el valor de RAM posible
	memory_slots	Número de espacios disponibles para la memoria RAM, indica si es posible utilizar mas de una memoria
	price	Precio de la memoria RAM en euros
RAM	name	Nombre de la RAM dado por la marca
	modules	Cantidad de módulos de RAM incluidos en el producto
	memory_per_module	Gigabytes de RAM por módulo
	price	Precio del paquete de RAM
Cooler	name	Nombre del ventilador dado por la marca
	socket	Tipo de conector de la CPU a la que se acopla
	performance	Cantidad de calor que es capaz de disipar
	price	Precio del componente
PSU	name	Nombre del la fuente de alimentación
	wattage	Energía de la fuente de almientacion en vatios
	price	Precio del componente
Case	name	Nombre de la caja o torre
	form_factor	Tipo de placa madre que puede introducirse
	price	Precio de la caja o torre

Tabla 1: *Componentes y atributos del sistema*

1. Obtención de los parámetros de los siguientes parámetros de entrada:
   - **Use**. Describe el uso final del equipo, tomando como valores relevantes para la inferencia *gaming* o *editing*.
   - **Performance**. Expresa el rendimiento esperado del equipo a configurar, puede tener valor *light*, *moderate* o *high*.
   - **Budget**. Presupuesto a tener en cuenta para la construcción del equipo.
   - **Size**. Tamaño del equipo final configurado, puede tener valor *regular* o *small*.
2. Uso de los parámetros *performance* y *use* para determinar los porcentajes del presupuesto que alocar a cada componente un porcentaje del presupuesto.
3. Obtención de todos los CPUs cuya *performance* se corresponde con la estipulada por el usuario y que se ajustan al presupuesto. Esta lista será considerada como el punto de comienzo para todas las builds posibles.
4. Regla para buscar qué GPU usar con cada CPU:
   - Si el usuario no especifica un rendimiento alto y la CPU cuenta con una gráfica integrada (iGPU), se toma como la gráfica por defecto del sistema.
   - Si la regla anterior falla, se busca una GPU en la base de datos que entre dentro del presupuesto.
5. Regla para buscar la placa base a usar en función de la CPU y el tamaño del ordenador (*size*). Se busca una placa base cuyo factor de forma se ajuste al

tamaño especificado por el usuario, tenga el *socket* necesario para la CPU y se ajuste al presupuesto.

6. Regla para buscar los módulos de RAM que utilizar dado el *use* introducido por el usuario y la placa base escogida, así como la cantidad de ellos que se ha de comprar:

    - Se escoge un kit de memorias RAM si uno o varios del mismo pueden poblar todas las ranuras de la placa base, proporcionan más memoria que la mínima requerida por el uso y menos que la soportada por la placa base, y se ajustan al presupuesto.
    - Si la anterior regla falla, se busca un kit cuyos módulos proporcionen más de la memoria mínima dada por el uso, no sobrepasen el número de ranuras de la placa base y no sobrepasen el presupuesto.

7. Regla para buscar la PSU en función de la energía consumida por la CPU y la GPU, y si se ajusta al presupuesto.

8. Regla para determinar el ventilador de la CPU en función de la *performance* del ordenador, el *socket* de CPU en el que se puede montar y si se ajusta al presupuesto.

9. Regla para obtener una caja o torre compatible con el factor de forma de la placa base y que se ajuste al presupuesto.

10. Finalmente, el programa imprime en orden decreciente las configuraciones posibles en orden decreciente de precio, descartando aquellas en las que no se encuentra algún componente compatible en la base de datos.

SECCIÓN 3

Conclusiones

El presente proyecto ha permitido conocer el proceso de diseño e implementación de sistemas recomendadores, comenzando desde la obtención de los datos hasta la explotación del conocimiento pasando por averiguar las fortalezas y debilidades de nuestra propuesta frente a otras alternativas.

En conclusión, el sistema presentado es capaz de proponer al usuario las características mínimas requeridas por su computador para satisfacer sus necesidades. El repositorio con el código desarrollado puede encontrarse en el anexo XX o en el siguiente enlace https://github.com/MikelMoreno/PCconfig.git

Referencias

[Molina, 2020] Molina, M. (2020). What is an intelligent system? *CoRR*, abs/2009.09083.

[Swartout, 1985] Swartout, W. R. (1985). Rule-based expert systems: The mycin experiments of the stanford heuristic programming project: B.g. buchanan and e.h. shortliffe, (addison-wesley, reading, ma, 1984); 702 pages, 40,50. $Artificial Intelligence, 26(3) : 364 - -366$.