

Entrega 1 proyecto - Prolog.



Pontificia Universidad
JAVERIANA
Bogotá

Ian Tomas Felipe fonseca Coronado,

Juan manuel Diaz

Santiago Jose Hernandez Rendon

Alejandro Molina Aranza

2 de Septiembre de 2025

Pontificia Universidad Javeriana

Jose Leon Andrade

Introducción a la IA

1. Introducción

El presente proyecto consiste en el desarrollo de un sistema de recomendación de productos para una tienda, implementado en el lenguaje Prolog. Para ello, se construyó una base de conocimiento con usuarios, productos, categorías, compras y calificaciones, a partir de la cual se diseñaron reglas que permiten calcular similitudes entre usuarios y generar recomendaciones personalizadas.

El proyecto busca poner a prueba algunos conceptos vistos en clase, con el fin de crear el sistema de recomendaciones como un ejemplo de algo que puede ser potenciado por IA, por medio del lenguaje Prolog.

2. Base de conocimiento

La base de conocimiento implementada en Prolog contiene la información para implementar el sistema de recomendación. Está compuesta por los siguientes elementos:

- **Usuarios:** representan las personas que interactúan con el sistema. Se definieron ocho usuarios: Juan, María, Carlos, Ana, Pedro, Sofía, Luis y Carmen. Ejemplo: usuario(juan).
- **Productos:** conjunto de ítems que pueden ser adquiridos. Se dividen en varias categorías. Ejemplo: producto(laptop_gaming).
- **Categorías:** cada producto pertenece a una de cuatro categoría, las cuales son: tecnología, educación, entretenimiento o deportes. Ejemplo: categoria(laptop_gaming, tecnología).
- **Compras:** relación entre usuarios y productos que han adquirido. Ejemplo: compro(juan, laptop_gaming).
- **Calificaciones:** cada usuario puede dar una valoración (entre 1 y 5 estrellas) a los productos comprados. Ejemplo: calificacion(juan, laptop_gaming, 5).

De esta forma, la base de conocimiento representa el punto de partida para llevar a cabo cada una de las actividades necesarias para el sistema.

3. Predicados implementados

En el sistema se desarrollaron diferentes predicados que permiten consultar información de la base de conocimiento, calcular similitudes entre usuarios y generar recomendaciones. A continuación, se describen los principales:

- **ha_comprado(Usuario, Producto)**

Verifica si un usuario ha comprado un producto específico.

obtener_calificacion(Usuario, Producto, Calificacion)

Devuelve la calificación que un usuario dio a un producto.

- **calificacion_alta(Usuario, Producto)**

Determina si un usuario otorgó una calificación superior a 3 estrellas.

- **productos_comunes(Usuario1, Usuario2, Count)**

Calcula la cantidad de productos que dos usuarios tienen en común.

- **similitud_usuarios(Usuario1, Usuario2, Similitud)**

Mide el grado de similitud entre dos usuarios con base en sus productos comprados.

- **recomendar_item(Usuario, ProductoRecomendado)**

Sugiere un producto a un usuario basándose en las compras y calificaciones de los usuarios más similares.

- **recomendar_lista(Usuario, ListaRecomendaciones)**

Genera una lista de recomendaciones para un usuario, ordenadas según la similitud con otros usuarios.

- **recomendacion_recursiva(Usuario, ProductoRecomendado)**

Propone productos explorando categorías relacionadas con las preferencias del usuario y las calificaciones de otros usuarios similares.

- **top_10_items_usuarios(ListaUsuarios, Top10)**

Devuelve los diez productos más recomendados según las calificaciones altas otorgadas por un conjunto de usuarios.

- **info_usuario(Usuario)**

Muestra toda la información de un usuario, incluyendo sus compras y calificaciones.

- **productos_categoria(Categoría)**

Presenta los productos pertenecientes a una categoría específica.

- **usuarios_similares(Usuario, UsuariosSimilares)**

Devuelve una lista con los usuarios más similares a un usuario dado.

4. Consultas de ejemplo

Para corroborar el correcto funcionamiento de la base de conocimiento, se implementaron las siguientes consultas:

- 

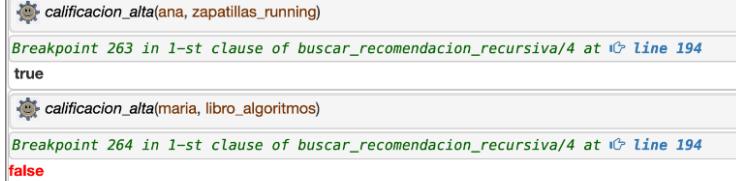
```
ha_comprado(sofia, smartphone_premium)
true
```

devuelve true si una persona compró dicho producto.

- 

```
obtener_calificacion(luis, monitor_4k, Calificacion)
Breakpoint 268 in 1-st clause of buscar_recomendacion_recursiva/4 at line 194
Calificacion = 5
```

Devuelve la calificación que una persona dió a un producto.

- 

```
calificacion_alta(ana, zapatillas_running)
Breakpoint 263 in 1-st clause of buscar_recomendacion_recursiva/4 at line 194
true
calificacion_alta(maria, libro_algoritmos)
Breakpoint 264 in 1-st clause of buscar_recomendacion_recursiva/4 at line 194
false
```

Devuelve true si la calificación es mayor a 3 o false si es menor o igual a 3.

- 

```
productos_comunes(carlos, luis, Count)
Breakpoint 266 in 1-st clause of buscar_recomendacion_recursiva/4 at line 194
Count = 2
```

Devuelve la cantidad de productos que dos personas tienen en común.

- 

```
similitud_usuarios(carlos, luis, Similitud)
Breakpoint 269 in 1-st clause of buscar_recomendacion_recursiva/4 at line 194
Similitud = 0.25
```

Devuelve el grado de similitud entre dos personas, basándose en sus productos comprados.

- 

```
recomendar_item(juan, ProductoRecomendado)
ProductoRecomendado = monitor_4k
Next 10 100 1,000 Stop
```

Sugiere un producto recomendado, basándose en compras y calificaciones de usuarios similares.

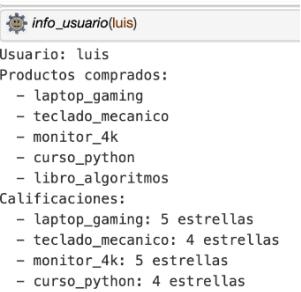
- 

Genera una lista de recomendaciones, ordenadas según la solicitud de otros usuarios

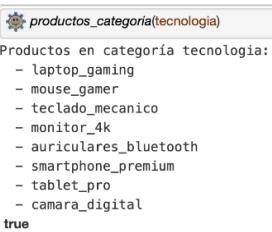
- 

Propone productos explorando categorías relacionadas con las preferencias del usuario y las calificaciones de otros usuarios similares.

- htgh

- 

Muestra toda la información de un usuario.

- 

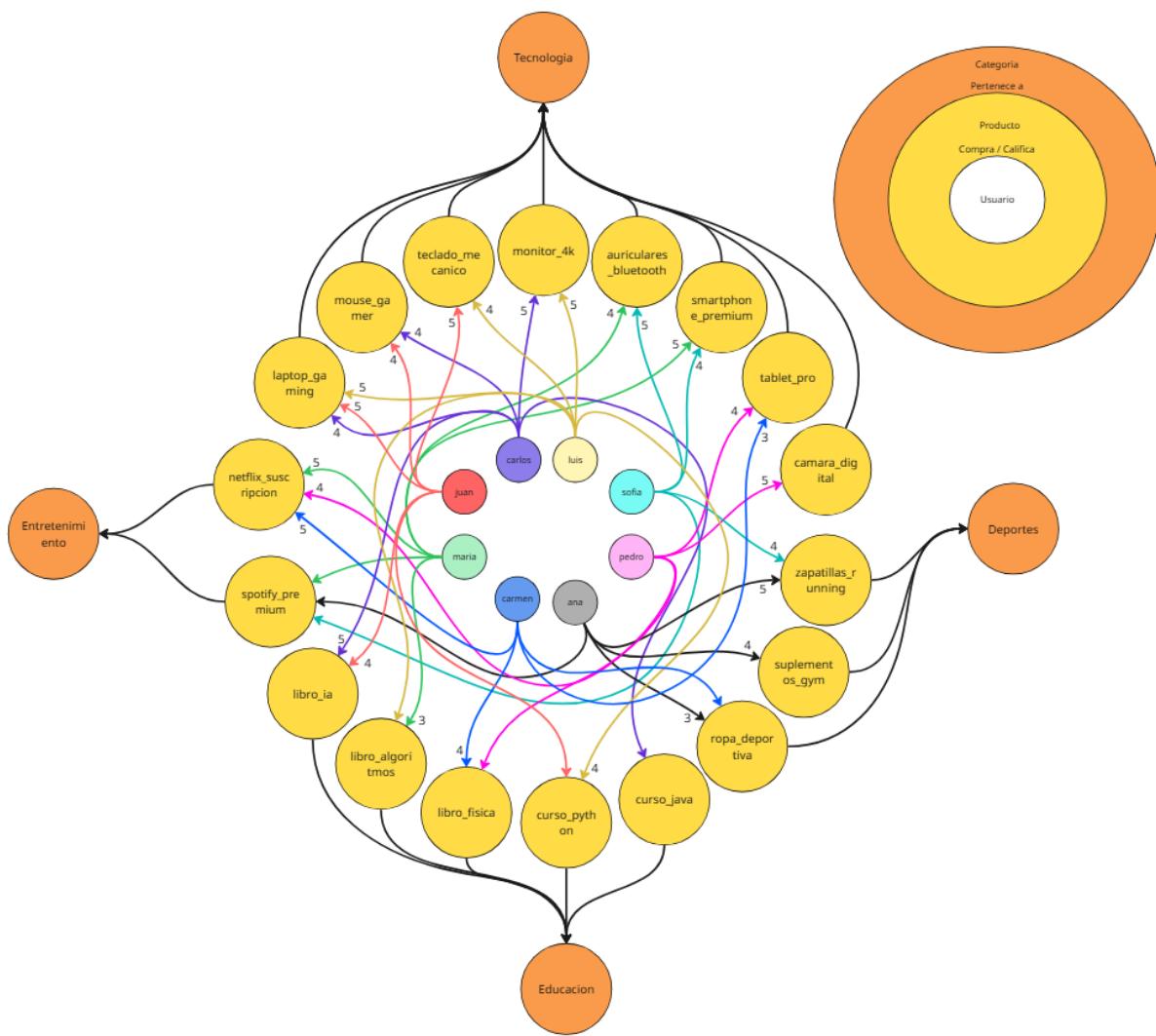
Muestra todos los productos en una categoría

- 

Devuelve una lista con los usuarios más similares.

5. Grafo

Al analizar el problema, se realizó el siguiente grafo:



el cual indica las relaciones entre los productos, los compradores y las categorías de los productos.

6. Conclusiones

Con este ejercicio se pudo evidenciar la programación lógica como herramienta para modelar soluciones con reglas y relaciones. Concentrándose en prolog, este se centra en describir hechos y crear reglas, lo cual resulta muy útil para manejar problemas de este tipo desde el paradigma utilizado.

Dentro del proyecto, este modelo se aplicó al diseño del sistema; la representación de usuarios, categorías y valoraciones como una base de hechos permitió establecer las relaciones entre preferencias y comportamientos. A través de reglas y queries específicos el sistema desarrolló recomendaciones personalizadas.