Universidad de la República

Facultad de Ingeniería

Recuperación de Información y Recomendaciones en la Web

Amazon Copilot

Autores:
Juan Pablo Conde
Xavier Iribarnegaray
Juan Pablo Sotelo

Profesores:
Libertad Tansini

27 de abril de 2025

Índice

1. Introducción

Amazon Copilot constituye un sistema diseñado para optimizar la búsqueda y selección de productos en plataformas de comercio electrónico. Basado en un conjunto de datos de productos de Amazon, el sistema implementará técnicas avanzadas de recuperación de información y procesamiento de lenguaje natural para proporcionar una experiencia de compra intuitiva y personalizada.

El proyecto integra tres funcionalidades principales: un sistema de búsqueda híbrido que combina técnicas semánticas y tradicionales para ofrecer resultados más relevantes; un asistente conversacional (Agente de Inteligencia Artificial) que proporciona una interfaz de búsqueda interactiva, refinando requerimientos mediante diálogo natural y utilizando como mecanismo subyacente el sistema de búsqueda híbrido implementado; y un sistema de recomendación que, basándose en los productos seleccionados en el carrito, emplea el mismo agente para identificar artículos similares o complementarios, mejorando así la experiencia de descubrimiento de productos.

2. Funcionalidades del Sistema

En esta sección se presentan las tres principales funcionalidades que se implementarán en Amazon Copilot.

2.1. Búsqueda Híbrida de Productos

Este componente combina técnicas de búsqueda semántica con métodos tradicionales de correspondencia textual para optimizar resultados. Incorpora representaciones vectoriales semánticas (embeddings) para capturar el significado contextual de las consultas, mantiene búsqueda por términos exactos para consultas específicas e implementa filtrado por categorías. El sistema ordena los resultados mediante un modelo que pondera tanto la relevancia semántica como la coincidencia sintáctica tradicional, logrando un equilibrio óptimo que aprovecha las fortalezas de ambos enfoques para mostrar los productos más pertinentes al usuario.

2.2. Búsqueda Conversacional con Agente de IA

El asistente conversacional proporciona una interfaz de búsqueda interactiva en lenguaje natural. Este agente mantiene el contexto durante la conversación, refina progresivamente los requerimientos del usuario mediante preguntas específicas, y utiliza la API de búsqueda híbrida como herramienta subyacente para ofrecer resultados personalizados. Resulta especialmente útil para usuarios sin conocimientos específicos sobre los productos que buscan.

2.3. Sistema de Recomendación con Agente de IA

Este sistema analiza los productos seleccionados en el carrito para sugerir artículos complementarios o alternativos. Opera automáticamente al visualizar el carrito, generando recomendaciones basadas en productos frecuentemente adquiridos conjuntamente y alternativas similares. Presenta explicaciones concisas sobre cada recomendación y adapta las sugerencias dinámicamente según el contenido del carrito y el historial de la conversación con el usuario.

3. Arquitectura del Sistema

Esta sección describe los componentes principales del sistema, sus responsabilidades e interacciones, así como las tecnologías propuestas para su implementación.

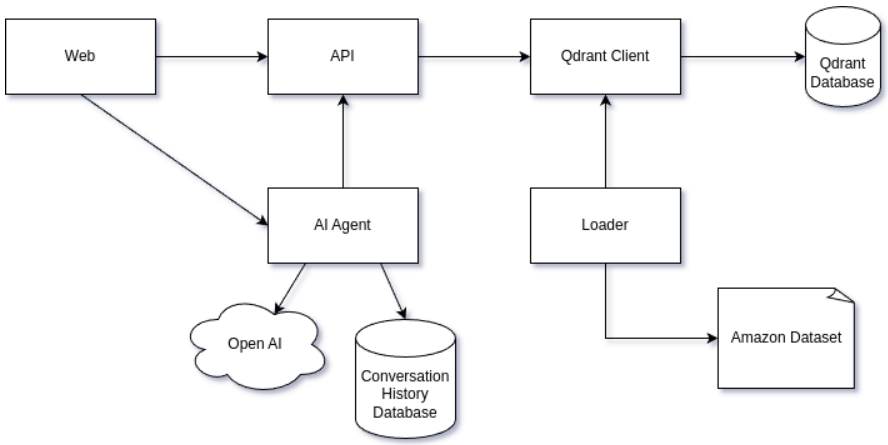

Figura 1: Diagrama de alto nivel de la arquitectura del sistema.

3.1. Web

Este módulo define la interfaz gráfica y las interacciones del usuario con el sistema (UI/UX), constituyendo el frontend de la aplicación. A través de esta capa, los usuarios pueden acceder a todas las funcionalidades del sistema de manera intuitiva y eficiente.

3.1.1. Diseño UX

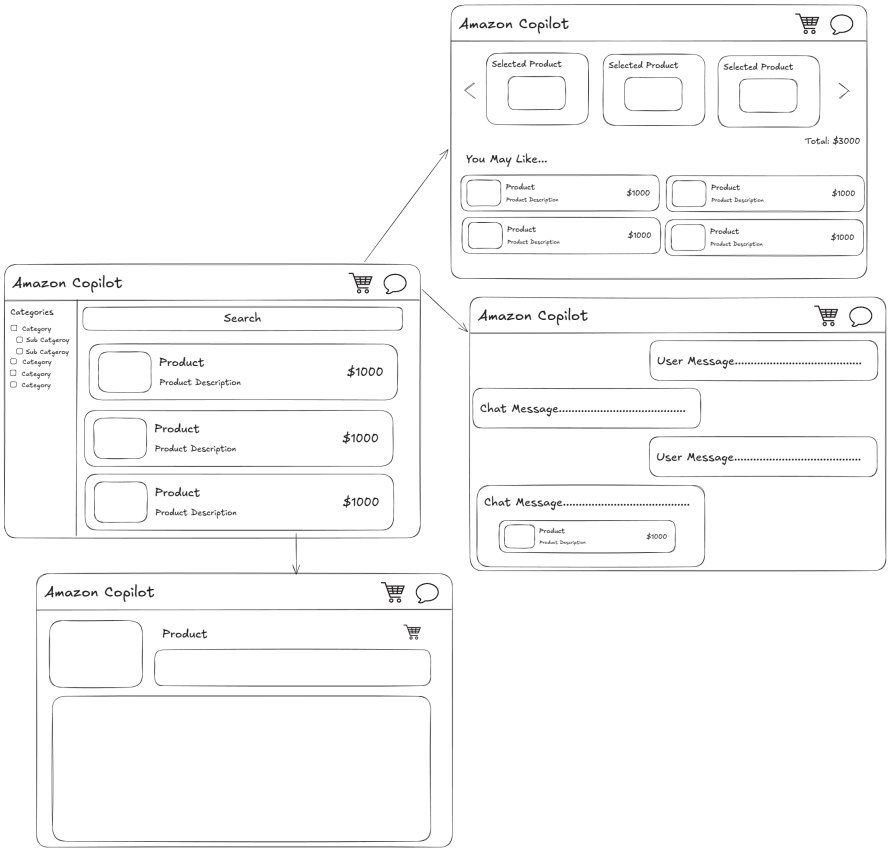

Figura 2: Diseño UX de la interfaz web.

Nota: Este diagrama modela las interacciones del usuario con el sistema. El diseño final de la interfaz de usuario está pendiente de desarrollo.

La interfaz permite al usuario: visualizar un listado inicial de productos con barra de búsqueda y filtros por categorías; realizar búsquedas que actualizan dinámicamente los resultados; acceder a detalles de productos específicos; gestionar un carrito de compras que muestra recomendaciones relacionadas; e interactuar con el asistente conversacional para búsquedas conversacionales, manteniendo el historial de la conversación.

3.1.2. Implementación

Dado que el objetivo principal del proyecto se centra en la funcionalidad de búsqueda y en el desarrollo del AI Agent, para el frontend adoptaremos la técnica de "Vibe Coding", utilizando herramientas como Cursor [1] y Replit [2].

> "There's a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. [...] I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works."

Andrej Karpathy, X [3]

Esta filosofía de desarrollo, asistida por modelos de lenguaje avanzados, nos permitirá iterar rápidamente sobre prototipos funcionales, concentrando nuestros esfuerzos técnicos en los componentes de búsqueda, recomendación e inteligencia artificial que constituyen el verdadero valor del proyecto.

3.2. API

Este componente constituye la capa de servicios que comunica el frontend con los distintos motores de búsqueda y recomendación. Incluye endpoints para listar productos (recibiendo parámetros de filtros y búsqueda) y para obtener información detallada de productos específicos.

3.2.1. Implementación

La API será implementada utilizando FastAPI [4], un framework moderno de Python que permite un desarrollo eficiente y robusto. FastAPI ofrece validación automática de datos mediante Pydantic [5], tipado estático que reduce errores en tiempo de desarrollo, y generación automática de documentación con OpenAPI y Swagger UI. Esta documentación servirá como recurso valioso para la fase de desarrollo del frontend.

3.3. Agente de IA

Este componente implementa la funcionalidad conversacional y de recomendación del sistema utilizando LangGraph [6] y los modelos generativos de OpenAI [7]. Utiliza la API de búsqueda híbrida como herramienta para acceder a los productos, permitiendo una arquitectura donde el diálogo con el usuario y la obtención de información están claramente separados.

Para mantener el contexto de las interacciones, el sistema se integra con una base de datos que almacena el historial de conversaciones, permitiendo referencias a consultas anteriores y facilitando la personalización progresiva de las respuestas.

3.4. Qdrant

Este módulo constituye el núcleo del sistema de búsqueda vectorial. Se compone de:

- **Base de datos vectorial**: Almacena productos y sus representaciones vectoriales (embeddings densos y esparsos), permitiendo búsquedas por similitud y coincidencia exacta.
- **Cliente de acceso**: Componente que gestiona las operaciones de indexación y consulta a la base de datos. Implementa búsqueda híbrida siguiendo las prácticas recomendadas por Qdrant [8], combinando embeddings densos (para capturar significado semántico) y esparsos (para coincidencias textuales precisas), permitiendo así obtener resultados relevantes tanto para consultas conceptuales como específicas.

La separación de este módulo facilita adaptaciones futuras, permitiendo reemplazar modelos de embeddings o migrar a soluciones alternativas sin impactar al resto del sistema.

3.5. Dataset

Para la información de productos, se utilizará el conjunto de datos de Amazon disponible en Kaggle [9]. Este conjunto de datos ofrece una amplia variedad de atributos por producto, incluyendo:

- Título y descripción detallada
- Categorías y subcategorías
- Precio y disponibilidad
- Valoraciones y número de reseñas
- Imágenes de productos
- Especificaciones técnicas

La riqueza de estos atributos permitirá implementar las modalidades de búsqueda previamente mencionadas.

Para la carga inicial del conjunto de datos, se desarrollará un módulo de carga consistente en scripts de Python que procesarán el archivo CSV, transformarán los datos en el formato requerido y utilizarán la funcionalidad de almacenamiento del cliente de Qdrant para indexar cada producto.

4. Referencias

[1] Cursor. (2024). *Cursor: The AI-first code editor*. https://www.cursor.com

[2] Replit. (2024). *Replit: The collaborative browser based IDE*. https://replit.com

[3] Karpathy, A. (2023, 11 noviembre). *Tweet sobre "vibe coding"*. X. https://x.com/karpathy/status/1886192184808149383

[4] FastAPI. (2024). *FastAPI: Framework web de alto rendimiento para APIs con Python*. https://fastapi.tiangolo.com

[5] Pydantic. (2024). *Pydantic: Validación de datos para Python*. https://docs.pydantic.dev/latest

[6] LangGraph. (2024). *LangGraph: Orchestration framework for building stateful, multi-actor applications with LLMs*. https://www.langchain.com/langgraph

[7] OpenAI. (2024). *OpenAI: Creating safe AI that benefits humanity*. https://openai.com

[8] Qdrant. (2024). *Qdrant: Vector Database*. https://qdrant.tech/documentation/search-precision/reranking-hybrid-search

[9] Parab, L. (2023). *Amazon Products Dataset*. Kaggle. https://www.kaggle.com/datasets/lokeshparab/amazon-products-dataset/data?select=Amazon-Products.csv