Informe de Proyecto: Sistema RAG para Recetas de Cocina

Estudiantes:

- Laura Sofia Vera - 201123145
- Juan Camilo Arrieta - 202121839
- Juan Pablo Camacho - 202110977
- Daniela Herrera - 202113704

Asignatura: Inteligencia Artificial aplicada a la economía (HE2)
Evaluación: Parcial 3

1. Introducción y Objetivo

Para nuestro proyecto final, desarrollamos un sistema de **Generación Aumentada por Recuperación (RAG)** enfocado en el dominio culinario. El objetivo principal es asistir a los usuarios en la preparación de alimentos, proporcionando recetas detalladas, traducidas y adaptadas a sus consultas en lenguaje natural.

Aunque la visión ideal del proyecto contempla un corpus especializado en gastronomía colombiana (con variaciones regionales de platos como el ajiaco o el sancocho), para la implementación técnica utilizamos el dataset corbt/all-recipes de Hugging Face. Este enfoque nos permitió validar la arquitectura del sistema con una base de datos robusta, simulando el funcionamiento que tendría con un corpus local. El sistema busca mitigar las alucinaciones comunes de los Modelos de Lenguaje (LLMs) al restringir sus respuestas a información verídica recuperada de una base de conocimiento confiable.

2. Arquitectura del Sistema y Procesamiento

La arquitectura desarrollada sigue un esquema modular dividido en dos grandes componentes: el **Encoder** (responsable de la ingestión y recuperación) y el **Decoder** (responsable de la generación de respuestas).

2.1. Preprocesamiento de Datos (Ingestión)

El primer desafío fue manejar la volumetría de datos. El dataset original contiene cientos de miles de recetas, lo cual excedía la capacidad de memoria y tiempo de procesamiento en el entorno de desarrollo (Google Colab).

- **Reducción y Limpieza:** Seleccionamos un subconjunto de **2,000 recetas** para permitir una iteración ágil. Se aplicó una limpieza de texto mediante expresiones regulares (regex) para eliminar saltos de línea innecesarios y normalizar espacios en blanco, facilitando la lectura por parte del modelo.

- **Chunking (Fragmentación):** Utilizamos el RecursiveCharacterTextSplitter. Definimos un tamaño de fragmento (*chunk size*) de **1500 caracteres** con un solapamiento (*overlap*) de **200 caracteres**. Esta decisión fue crítica: un tamaño mayor asegura que la lista de ingredientes y las instrucciones de una receta no se corten abruptamente, preservando el contexto semántico necesario para la recuperación.

2.2. Componente Encoder (Recuperación)

Este componente se encarga de traducir el texto a representaciones matemáticas (vectores) y buscar la información relevante.

- **Modelo de Embeddings:** Implementamos el modelo **sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2**. Elegimos este modelo por tres razones:

1. **Multilingüismo:** Capacidad para entender consultas en español y relacionarlas con contenido en inglés.
2. **Eficiencia:** Es un modelo ligero diseñado para tareas de similitud semántica (SBERT).
3. **Dimensionalidad:** Genera vectores densos que capturan eficazmente el significado de oraciones cortas y párrafos.

- **Vector Store (FAISS):** Para el almacenamiento y búsqueda, utilizamos **FAISS (Facebook AI Similarity Search)**. Esta tecnología permite indexar los vectores y realizar búsquedas de similitud (distancia euclidiana o producto punto) de manera extremadamente rápida.
- **Proceso de Búsqueda:** Cuando el usuario realiza una pregunta, el sistema la vectoriza y recupera los $k=3$ **fragmentos** más similares. Esto asegura que el modelo generativo reciba solo la información más pertinente, filtrando el ruido del resto de la base de datos.

2.3. Componente Decoder (Generación Aumentada)

Para la fase de generación, diseñamos una arquitectura híbrida que combina la recuperación local con el procesamiento en la nube, superando las limitaciones de hardware locales.

- **Selección del Modelo (Mistral-7B):** Utilizamos **Mistral-7B-Instruct-v0.2** a través de la API de Hugging Face. Tras experimentar con modelos más pequeños (como Qwen-1.5B), identificamos que carecían de la capacidad de razonamiento necesaria para traducir términos culinarios sin errores (alucinando, por ejemplo, "cacahuetes" en lugar de "cacao"). Mistral-7B demostró una comprensión superior del contexto y las instrucciones.
- **Pipeline Híbrido:** El flujo de datos es el siguiente:

1. El entorno local recupera los *chunks* relevantes con FAISS.
2. Se construye un *prompt* enriquecido que viaja a la API de Hugging Face.
3. El modelo en la nube procesa la solicitud y devuelve la respuesta generada.

- **Ingeniería de Prompts (Prompt Engineering):** Implementamos un "System Prompt" robusto para controlar la salida:

- **Rol:** "Chef Experto Bilingüe".
- **Guardrails (Reglas de Seguridad):** Instrucciones explícitas para evitar errores de traducción recurrentes (ej. *"Cocoa = CACAO, no maní"* y *"Stick of butter = Barra de mantequilla"*).
- **Restricción de Contexto:** Se obliga al modelo a responder **únicamente** basándose en las recetas recuperadas, reduciendo drásticamente las invenciones.

3. Limitaciones y Desafíos

A pesar de la robustez del sistema, identificamos limitaciones inherentes al enfoque RAG y al corpus utilizado:

- **Subjetividad del Usuario:** Consultas como *"dame la mejor receta"* son difíciles de satisfacer, ya que el modelo carece de información sobre los gustos personales del usuario.
- **Dependencia del Corpus:** La calidad de la respuesta está acotada por la calidad de los documentos recuperados. Si la base de datos (reducida a 2000 recetas) no contiene un plato específico (ej. "ajiaco"), el sistema no podrá generar la respuesta, aunque el modelo de lenguaje conozca el plato por su entrenamiento general.
- **Alucinaciones Numéricas:** Aunque se corrigieron errores de traducción, los LLMs tienden a fallar en conversiones matemáticas precisas (ej. gramos a tazas) sin el uso de herramientas externas.

4. Conclusiones

El proyecto demuestra la viabilidad de implementar un sistema RAG funcional y eficiente utilizando recursos accesibles y gratuitos. La combinación de **FAISS** para la recuperación local de alta velocidad y **Mistral-7B (vía API)** para el razonamiento complejo resultó ser una arquitectura óptima.

Logramos transformar una base de datos de texto crudo en un asistente culinario capaz de entender preguntas en español, buscar en documentos en inglés y generar respuestas coherentes y culturalmente adaptadas. Este sistema modular permite futuras expansiones, como la integración de un corpus 100% colombiano o la adición de filtros dietéticos, sin necesidad de reentrenar los modelos fundacionales.