HE2: IA Aplicada a la Economía

Parcial 3:

Cocktail RAG – Documento Explicativo

Samuel Suárez, Julián Fandiño, Cesar Augusto Espinosa, Liz Katherine Lara

Este proyecto busca aplicar la metodología de *Retrieval Augmented Generation* (RAG) para enviar los documentos de la base *cocktail_recipes*[1] a un modelo de LLM capaz de proponer recetas de cocteles basándose en el input del usuario, en inglés o español.

0. Sobre la elección de la base de datos

Después de revisar varias propuestas, que incluían desde análisis de minutas de política monetaria, opiniones de profesores en X, sistemas de recomendación para rutinas de gimnasio, consultas sobre instrumentos ambientales, material de estudio, boletines laborales del DANE y reportes epidemiológicos de la OMS, concluimos que la opción más sólida para implementar un sistema RAG era la basada en recetas de cócteles. Esta propuesta resultó más interesante y práctica porque proviene de un dataset público, completo y bien estructurado, que ya incluye metadatos e instrucciones claras. Además, sus textos son cortos y uniformes, lo que facilita la preparación de embeddings, evita problemas de chunking y permite construir un prototipo funcional de manera eficiente. Por estas razones, optamos por desarrollar el RAG de "Bartender IA" como nuestro caso principal.

1. Sobre la base de datos

La base *cocktail_recipes* contiene 875 filas, cada una con una receta distinta para un coctel. Existen cocteles repetidos en la base, que provienen de distintas fuentes y tienen recetas que varían entre sí, por lo cual nuestra arquitectura toma en cuenta varios documentos al generar un output. Cada dato en la base contiene varios textos principales que incluyen los ingredientes (*ingredients)* los pasos a seguir para mezclar el coctel (*directions*), y una columna adicional con datos como el lugar de origen o la historia del coctel nombrada *misc*. También contiene metadatos que incluyen en nombre del coctel (*title*), la fuente de donde se tomó (*source*), y una etiqueta para el reconocimiento de entidades nombradas (*ner)* que contiene una lista de cadenas de caracteres con los nombres de los licores usados en el coctel. Lo anterior suma a 6 columnas. Existen columnas que no contienen ingredientes, direcciones, misc o ner, sin embargo, no existen filas vacías del todo y aquellas donde falta un texto o un metadato complementan la información faltante en el resto de las columnas.

Para determinar si debíamos aplicar *chunking* a los documentos, con el fin de no superar el límite de *tokens* permitido por la matriz de entrada de nuestro *encoder*, analizamos la longitud potencial de cada uno. Para ello, concatenamos todas las columnas y calculamos la extensión en palabras de cada documento. Encontramos que solo 3 documentos, equivalentes al 0,34% del total, superan las 210 palabras (ver anexo 1). Dado que

[1] Datos creados por Brian Arbuckle. Tomado de Hugging Face en: https://huggingface.co/datasets/brianarbuckle/cocktail_recipes

utilizaremos un *encoder* con un límite de 256 *tokens*, optamos por eliminar estos 3 casos: no resulta eficiente implementar un proceso de *chunking* únicamente para ellos, considerando además los costos asociados, como la pérdida de contexto, incluso si se utiliza *overlapping*. El límite de 210 palabras se debe a que en promedio cada palabra tiene más de un token. Finalmente, usamos los textos con las columnas concatenadas como input para el encoder.

2. El encoder

Para el *encoder* se utilizó el modelo sentence–transformer open source *all-MiniLM-L6-v2.* Este está diseñado para convertir cada documento en un vector denso de dimensión fija –384 dimensiones– que captura su significado global (a través de *mean pooling*). Se trata de un modelo de seis capaz, por lo tanto, tiene un costo computacional bajo en relación con otros modelos. Ha de decirse que fue publicado bajo licencia por *Apache 2.0,* por lo cual su uso está permitido en proyectos académicos.

De esta manera, con ayuda de la librería open source *FAISS (Facebook AI Similarity Search)* desarrollada por Meta AI se construyó un *vectorstore*. Esta estructura permite almacenar las representaciones vectoriales (*embeddings)* de los documentos y realizar consultas de similitud de manera eficiente. Así, se implementó una función que recibe como parámetro la búsqueda y entrega una cadena que es resultado de la concatenación de los tres primeros resultados de la consulta de similitud.

3. El decoder

Para el decoder se utilizó el modelo de lenguaje creado por Mistral AI: *Mistral-7B-Instruct-v0.2.* Este está diseñado para generar texto coherente. En esta etapa, el modelo recibe un *prompt* que combina el contexto recuperado por el encoder con la pregunta del usuario, y luego procesa este texto mediante su tokenizador, configurado para usar el *eos token* como *padding*. El modelo se ejecuta en precisión *float16* y distribuido automáticamente entre los dispositivos disponibles con `device_map="auto"`, lo que optimiza memoria y velocidad. Posteriormente, el modelo genera una respuesta de hasta 256 tokens nuevos utilizando muestreo con temperatura de 0.7 y top_p de 0.95, parámetros que balancean creatividad y coherencia en las respuestas. Finalmente, la función decoder limpia la salida del modelo y devuelve únicamente la respuesta relevante, completando así la fase generativa del sistema RAG.

4. Conclusión

Este proyecto logra construir un asistente conversacional capaz de recomendar cócteles de manera personalizada. Al combinar la búsqueda semántica con un modelo generativo, el sistema puede entender lo que el usuario busca, encontrar las recetas más relevantes en la base de datos y construir una respuesta coherente. El resultado es un bartender virtual que no solo recupera información, sino que la procesa y presenta de forma natural, demostrando cómo la metodología RAG puede aplicarse efectivamente a dominios específicos con datos bien estructurados.

Anexos:

Anexo 1: Histograma de la longitud de textos combinados