Sistema Inteligente de Información y Recomendaciones Gastronómicas

Marian Aguilar Tavier[1] and Katherine Rodriguez Rodríguez.[2]

[1] Universidad de La Habana
[2] Universidad de La Habana

Introducción

En la actualidad existe una enorme tradición culinaria que proviene de nuestros antepasados y a su vez, el avance de la tecnología provoca que cada día surjan nuevos métodos y técnicas de cocina para explotar al máximo la experiencia gastronómica.

Por otro lado, los sistemas de recuperación de información, de conjunto con los modelos de lenguaje abren un campo de infinitas posibilidades para facilitar el acceso a la información, permitiendo a los usuarios acceder a una gran cantidad de datos y obtener resultados más precisos.

En este informe se abordará más a detalle cómo los modelos de lenguaje complementan a los modelos de recuperación de información mediante la implementación de un RAG que le permita a los usuarios obtener recomendaciones e información gastronómica relevante.

Descripción del Proyecto

El sistema consiste en un asistente de cocina que proporciona información personalizada a sus usuarios sobre recetas, técnicas culinarias, consejos, información nutricional, además de brindar recomendaciones de artículos y recetas, teniendo en cuenta lo que el usuario ha consultado anteriormente.

Para lograr esto se implementó un modelo de Generación Aumentada por Recuperación(RAG) que obtiene información de fuentes conocidas en el mundo de la gastronomía y combina los datos recopilados y almacenados en una base vectorial, con un modelo de lenguaje(LLM) para generar respuestas coherentes y enriquecidas.

A continuación se exponen las principales ideas o principales componentes implementadas en el sistema.

Retrieval-Augmented Generation (RAG)

La implementación del RAG permitió obtener respuestas más significativas en el ámbito gastronómico, ya que la respuesta del usuario, no depende solamnete del modelo del lenguaje, sino que cuenta con una base de conocimiento que

permite enriquecer las respuesta y a la vez proporciona información actualizada y relevante.

Cuando el usuario realiza una consulta, el sistema utiliza similarity search para determinar los documentos más relevantes y los ordena además según su grado de coincidencia con la consulta. Luego, con el contexto creado con estos documentos que se recuperan, el modelo de lenguaje genera una respuesta donde tiene en cuenta los documentos recuperados así como la consulta hecha por el usuario.

Para la implementación del RAG se utilizó LangChain, un framework que permite la integración entre los LLMs y fuentes externas de información. Luego, la base de datos utilizada, los embeddings, las funciones de similitud y el modelo de lenguaje empleado fueron integraciones con este framework.

Como modelo de lenguaje se utilizó el modelo Mistral de Ollama (desde su integración en LangChain).

Crawling

En aras de lograr una mayor variedad en las respuestas y recopilar la mayor cantidad de información, se implementó un crawler sobre la base de más de una fuente de información. En esta caso: The Gourmet Journal, Project Gutenberg y Cookpad.

The Gourmet Journal es un periódico de gastronomía que proporciona artículos variados que incluyen desde recetas, curiosidades sobre alimentos y técnicas culinarias, consejos gastronómicos para mejores hábitos alimenticios, información sobre vinos y estrategias de marketing para restaurantes.

The Project Gutenberg es una biblioteca digital bastante conocida por tener libros de libre acceso sobre temas muy variados, por lo que se utilizó esta fuente para obtener libros de recetas, de historia de la cocina en todo el mundo, así como para la obtención de información relacionada con las propiedades nutricionales de los alimentos.

Por otro lado, se utilizó Cookpad, una plataforma de cocina mundial, que contiene miles de recetas, consejos, opiniones de usuarios y permite adentrarse un poco más en el mundo de la comida casera. A continuación se describe cómo se llevó a cabo el proceso de implementación del crawler en sus dos etapas:

Fase Inicial

Para la fase inicial, se trabajó con los artículos de The Gourmet Journal y los libros de Project Gutenberg, cada uno con sus peculiaridades a la hora de recopilar y almacenar la información. En ambos casos, el sistema recopila consultas que se consideran relevantes en el mundo de la gastronomía y busca en sus respectivas fuentes artículos o libros relacionados, luego se recorren los enlaces de forma tal que, si el nuevo enlace descubierto no ha sido visitado, entonces se almacena la información así como los metadatos del artículo de forma local(como raw data inicialmente). Los archivos se almacenaron en formato plano(txt) para

su posterior procesamiento e introducción en la base de datos vectorial, de la que se hablará en las próximas secciones. En la fase inicial se recuperaron más de 2000 ficheros para garantizar, en parte, un mejor funcionamiento del sistema.

Fase Dinámica o Automatizada

Para el crawler dinámico se tomó Cookpad como fuente de datos, y se consideró información desactualizada a todas aquellas fuentes cuya fecha de publicación estuvieran por debajo de una fecha X; luego el crawler dinámico se activa cuando se cumple lo mencionado anteriormente y/o cuando los documentos recuperados de la base de datos vectorial tienen un valor de relevancia para el usuario por debajo de un umbral establecido.

Procesamiento y Almacenamiento de Datos

Como se mencionó anteriormente, el formato de cada fuente utilizada varía, por lo que las estrategias de limpieza y almacenamiento, difieren entre ellas.

Limpieza

Luego de almacenar los datos en la fase inicial, se inició un proceso de exploración para determinar cuáles eran los principales modificaciones necesarias en las fuentes extraídas para lograr un mejor funcionamiento del sistema, por lo que se eliminaron espacios en blanco, caracteres especiales, se normalizó el texto, se eliminaron los links referentes a redes sociales, los pie de páginas, headers, todo en dependencia de la fuente de datos.

Chunking

Debido a la heterogeniedad de los ficheros recuperados, se comprobó mediante la experimentación que el uso de la misma estrategia de chunking para todas las fuentes de datos no garantiza una buena recuperación de información, por lo que se adoptaron varias estrategias. Para el caso de los artículos de The Gourmet Journal, se implementó el chunking recursivo, teniendo en cuenta una número n de tokens y un overlap de 200. Debido a la variedad en el contenido de es etsa fuente, establecer un tamanno fijo para cada artículo resultó algo muy difícil e ineficiente, por lo que con el overlap de 200 se evitó que los chunks fueran interrumpidos abruptamente.

La estrategia adoptada para el caso de Cookpad, fue un poco diferente, ya que todos los ficheros seguían un formato común(recetas), por lo que cada receta fue dividida tanto por ingredientes como por sus pasos de elaboración.

Almacenamiento

Luego de limpiar y dividir por chunks los archivos, se almacenaron los mismos en un repositorio vectorial. En este caso se utilizó Chroma, ya que permite el almacenamiento persistente y además permite almacenar correctamente los metadatos de los archivos. Es importante mencionar que no todas las fuentes almacenan la misma cantidad de metadatos, sin embargo en todas se recopila el id del documento, el título, autor, la url de la fuente, así como la fecha en que se publicó; logrando una mejor recuperación de información.

Para representar los textos como vectores se utilizó un modelo de embedding de HuggingFace, específicamente un modelo de *sentence-transformers*, compatible con LangChain.

Componente de Recomendación

Se implementó una lógica de recomendación basada en el filtrado por contenido. Para ello se creó una nueva colección en Chroma, esta colección sería la encargada de almacenar como vectores las queries de los usuarios, por lo que, para cada consulta, se guarda el id del usuario, el texto de la consulta y la fecha en la que se realizó.

Perfil de Usuario

Luego, para determinar el perfil de usuario, se promedió por cada usuario todas sus consultas, generando un vector que refleja las preferencias del usuario. Este vector se utilizó posteriormente en la base vectorial de documentos, de esta forma se le recomienda al usuario documentos similares a los intereses de su perfil.

Además para evitar que el usuario obtenga las mismas recomendaciones, se le brinda como sugerencia artículos que él aún no haya consultado.

Para consultar la implementación revise https://github.com/mariandc18/ForkPilot

Conclusiones

En este proyecto se implementó un sistema de Generación Aumentada por Recuperación(RAG) que proporciona al usuario un asistente para consultar temas relacionados con la gastronomía. A pesar de la implementación abordada se debe tener en cuenta que, se pueden implementar mejoras para lograr una recuperación de información más eficiente. Por ejemplo, para que el usuario obtenga recomendaciones más personalizadas se puede incluir un filtrado híbrido que no solo tome en cuenta el perfil del usuario, sino que también sea capaz de revisar en la base vectorial, aquellos usuarios con un historial similar a él.

Para una mayor variedad de resultados se pueden incluir más fuentes de datos en el crawler dinámico, y a su vez se pueden seguir experimentando estrategias para dividir los documentos por chunks para garantizar una búsqueda semántica más eficiente.

En aras de una mayor relevancia de resultados de los documentos para el usuario en las recomendaciones, además de promediar sus consultas se pueden ponderar aquellas consultas que realizó de forma más reciente; se puede incluir además el contexto de los documentos que consultó (siempre que tenga una relevancia significativa).

References

1. https://python.langchain.com/docs/tutorials/rag/ LangChain Docs.
2. Shailja Gupta, Rajesh Ranjan, Surya Narayan Singh: A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. Arxiv, 2024.
3. https://ollama.readthedocs.io/en/quickstart/. Ollama Docs
4. https://python.langchain.com/docs/how$_t o/semantic$ – $chunker/percentileSemanticChunking$.
5. https://towardsdatascience.com/rag-101-chunking-strategies-fdc6f6c2aaec/ Estrategias de Chunking.
6. https://docs.trychroma.com/docs/overview Chroma Database Docs.
7. https://huggingface.co/docs/inference-providers/providers/hf-inference HuggingFace Hub Docs.