Implementación de un Chatbot para la Recuperación de Artículos Científicos mediante Generación Aumentada por Recuperación (RAG)

Jennifer de la Caridad Sánchez Santana[1] and Reinaldo Cánvas Gamón[2]

Facultad de Ciencia de Datos, Universidad de La Habana
La Habana, Cuba

Índice

Abstract. Este artículo presenta el desarrollo de un sistema de recuperación y recomendación de artículos científicos, basado en el modelo de Generación Aumentada por Recuperación (RAG). El objetivo es construir un chatbot capaz de responder preguntas en lenguaje natural utilizando artículos previamente recopilados y procesados, mejorando el acceso a la información académica relevante mediante técnicas de embeddings y búsqueda semántica.

Keywords: Recuperación de Información, RAG, chatbot, embeddings, artículos científicos

1 Introducción

En el ámbito de la investigación científica, mantenerse actualizado sobre los últimos descubrimientos es esencial. Sin embargo, el volumen de publicaciones científicas crece exponencialmente, lo que representa tanto una oportunidad como un desafío para investigadores, estudiantes y profesionales. La búsqueda eficiente de información relevante y el acceso a los documentos completos suelen ser tareas complejas.

Con el objetivo de facilitar el acceso a conocimiento científico, se propone el desarrollo de un chatbot que simule una conversación natural con el usuario, permitiéndole recuperar información especializada mediante una interfaz interactiva. Este sistema combina técnicas modernas de procesamiento de lenguaje natural, embeddings semánticos y recuperación de información basada en similitud.

2 Desarrollo

2.1 Extracción de Artículos

Para alimentar el sistema con información científica, se extrajeron aproximadamente 2000 artículos relacionados con unos 100 temas de interés. Las consultas se realizaron utilizando la API de Crossref (`https://api.crossref.org/works`), que devuelve metadatos como el autor, la fecha de publicación, el DOI, entre otros. Posteriormente, se empleó la API de Unpaywall (`https://api.unpaywall.org/v2/{doi}`) para recuperar la URL del artículo completo, en los casos en que esté disponible gratuitamente. Los artículos con acceso abierto se procesaron para extraer tanto sus metadatos como su contenido textual. Inicialmente, los datos se almacenaron en un archivo CSV por razones de conectividad.

Extracción de Referencias Una vez recopilados los artículos, se analizaron sus contenidos en busca de referencias a otros trabajos mediante expresiones regulares. A partir de los DOIs identificados, se consultó nuevamente la API de Crossref para obtener los metadatos correspondientes y enriquecer la colección.

2.2 Almacenamiento en Repositorio Vectorial

Preprocesamiento Los documentos se depuraron para eliminar aquellos con contenido o metadatos faltantes. Se unificaron los artículos originales y las referencias, y se procesaron campos como el contenido y el resumen. Se eliminaron caracteres especiales, espacios innecesarios, secciones irrelevantes como las referencias, y artículos con menos de 200 palabras.

Generación de Embeddings Cada artículo se dividió en dos componentes: texto (contenido + resumen) y metadatos (autor, idioma, fecha, URL, etc.). El texto se fragmentó en bloques de unas 1200 palabras con una superposición de 200, generando aproximadamente 91 717 fragmentos. Estos fueron procesados mediante el modelo `sentence-transformers/all-MiniLM-L6-v2`, seleccionado por su eficiencia, bajo consumo de recursos y buen rendimiento en tareas de búsqueda semántica.

Los vectores resultantes, junto con los metadatos, fueron almacenados en la base de datos vectorial Chroma, una solución open-source optimizada para búsquedas basadas en embeddings y filtrado por metadatos.

2.3 Implementación del Chatbot

Interfaz Visual La interfaz, desarrollada con Streamlit, presenta el título "Chatbot RAG de Investigación Científica". En la parte superior se sugieren preguntas al usuario, mientras que en la parte inferior se permite ingresar consultas personalizadas. Al seleccionar una pregunta, se muestra la respuesta generada, los artículos utilizados como referencia y recomendaciones adicionales basadas en el historial de interacciones.

Lógica Backend El backend, implementado con FastAPI, se conecta al repositorio vectorial para recuperar documentos relevantes según la similitud coseno con la consulta del usuario. Si el puntaje de similitud es inferior a 0.25 o los documentos están desactualizados (más de cinco años), se realiza una búsqueda en tiempo real similar a la fase de recolección inicial.

Las respuestas generadas provienen del modelo `HuggingFaceH4/zephyr-7b-beta`, accedido vía API en `https://api-inference.huggingface.co/models/HuggingFaceH4/zephyr-7b-beta`. Este modelo fue elegido por su balance entre tamaño, eficiencia y calidad de respuesta en inglés.

Para generar recomendaciones, se analiza el historial de consultas almacenado en un archivo JSON. Se filtran las palabras vacías (stop words) y se extraen los trigramas más representativos mediante la métrica TF-IDF.

3 Conclusiones

Este trabajo presenta una solución práctica al problema de acceso a información científica mediante un chatbot que combina técnicas de recuperación semántica y generación de lenguaje. El sistema permite obtener respuestas sintetizadas y referencias relevantes, facilitando la exploración de conocimiento en diversos dominios.

4 Recomendaciones

- Aumentar la cantidad de artículos en el repositorio vectorial.
- Ampliar la cuota de consultas disponibles para el modelo `HuggingFaceH4/zephyr-7b-beta`.
- Desplegar el repositorio y el chatbot en servidores en línea para mayor accesibilidad.
- Incluir documentos en otros idiomas, especialmente en español, para diversificar las fuentes.

References

1. Repositorio del chatbot descrito en este informe. Disponible en: `https://github.com/Jennyfer2004/SRI-Investigaci-n-cient-fica`, last accessed 2025/5/18.
2. How to Use RecursiveCharacterTextSplitter in LangChain , `https://medium.com/towards-agi/how-to-use-recursivecharactertextsplitter-in-langchain-23bcb0448fca`, last accessed 2024/9/26.
3. Check for URL in a String - Python, `https://www-geeksforgeeks-org.translate.goog/python/python-check-url-string/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=rq`, last accessed 2025/4/12.
4. Top embedding models for RAG, `https://modal-com.translate.goog/blog/embedding-models-article?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc`, last accessed 2024/10/30.
5. ¿QUÉ ES EL DOI?, `https://3ciencias.com/que-es-el-doi/`, last accessed 2021/5/3.

6. sentence-transformers/all-MiniLM-L6-v2 , https://huggingface-co.translate.goog/sentence-transformers/all-MiniLM-L6-v2?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc, last accessed 2024/6/18.
7. Tutorial de la base de datos vectorial Chroma, https://anderfernandez.com/blog/chroma-vector-database-tutorial/, last accessed 2023/6.
8. Chroma is the open-source AI application database. Batteries included, https://www.trychroma.com/, last accessed 2025.
9. Exploring Vector Databases: A Guide to Their Role in AI Tech, https://www-projectpro-io.translate.goog/article/vector-databases/903?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc, last accessed 2024/10/28.
10. Model Card for Zephyr 7B Beta. Disponible en: https://huggingface.co/HuggingFaceH4/zephyr-7b-beta.