Reporte del Proyecto: Sistema de Retrieval-Augmented Generation para Recomendación de Recetas

Sofia Angulo, Daniel González, Felipe Rosas & Gabriela Zamora

11 de Diciembre

1. Introducción

El presente proyecto implementa un sistema de Retrieval-Augmented Generation orientado a la recomendación de recetas culinarias en respuesta a consultas de usuarios, empleando un subconjunto de 300 recetas extraídas del conjunto de datos RecipeNLG, que comprende 7.198 recetas en total. Esta arquitectura aborda las limitaciones intrínsecas de los Modelos de Lenguaje Grandes, tales como LLaMA 2, mediante la integración de mecanismos de recuperación semántica y generación de texto. La fundamentación teórica se centra en la mitigación de tres desafíos principales de los LLMs: la ausencia de conocimiento sobre datos dominio-específicos, la obsolescencia del conocimiento entrenado y la propensión a las alucinaciones. Se opta por RAG en detrimento del fine-tuning debido a su mayor eficiencia computacional y flexibilidad para actualizaciones, configurándose como un caso paradigmático en el ámbito culinario.

2. Fundamentación Teórica

Los LLMs, entrenados sobre corpora generales hasta un punto de corte temporal como el de LLaMA 2 en julio de 2023, exhiben deficiencias notables: ignorancia de datos específicos, como las recetas particulares de RecipeNLG; desactualización, solventada por RAG al permitir la incorporación dinámica de nuevos vectores sin reentrenamiento integral; y alucinaciones, atenuadas mediante prompts sistemáticos que constriñen las respuestas al contexto recuperado, como la instrucción "Responde exclusivamente en base a las opciones de recetas obtenidas".

RAG fusiona Retrieval, que implica búsqueda en bases vectoriales, y Generation, que consiste en producción textual por LLM, evocando la metáfora de un examen con libro abierto. En contraposición al fine-tuning completo, que requiere actualizar siete mil millones de parámetros en LLaMA 2 7B y consume recursos masivos equivalentes a 184.320 horas-GPU, o al fine-tuning eficiente vía LoRA, adecuado para políticas específicas pero menos flexible para cambios frecuentes en la base de conocimiento, RAG aprovecha encoders para embeddings semánticos y decoders para generación autoregresiva. Esta elección técnica justifica la no necesidad de datasets extensos de entrada-salida ni reentrenamientos iterativos, permitiendo añadir recetas mediante codificación vectorial simple.

La similitud semántica se sustenta en espacios vectoriales de alta dimensionalidad con 384 dimensiones, donde la proximidad euclídea captura analogías conceptuales como "gato duerme" equivalente a "felino descansa". Se adopta la similitud coseno:

$$\sim (q, d) = \frac{q \cdot d}{\|q\| \|d\|}$$

invariante a la magnitud, computacionalmente eficiente e interpretable, con puntuaciones superiores a 0.7 que denotan alta relevancia. Las restricciones contextuales de 512 tokens en el encoder y 4.096 en LLaMA 2 se gestionan potencialmente mediante chunking o ventanas deslizantes, aunque innecesarias en este caso dada la brevedad de las recetas inferiores a 500 tokens.

3. Descripción del Proyecto e Implementación

El sistema indexa 300 recetas como cadenas textuales concatenadas con el formato "NOMBRE: nombre\nINGREDIENTES: ingredientes\nPASOS: instrucciones", transformadas en embeddings mediante el modelo `sentence-transformers/paraphrase-multilingual-mpnet-base-v2`. Este modelo se basa en XLM-RoBERTa, un transformer encoder-only con 12 capas, atención bidireccional y mean pooling para vectores densos de 384 dimensiones. El almacenamiento se realiza en un arreglo NumPy en memoria con forma 300×384, idóneo para prototipos pero escalable a bases vectoriales como FAISS para búsquedas optimizadas.

4. Proceso y Resultados

El proceso se divide en dos fases principales: indexación previa y flujo de consulta en tiempo real, implementadas en un notebook Colab con Python para garantizar reproducibilidad y eficiencia.

4.1. Fase de Indexación Previa

Inicialmente, se transforman las recetas en textos estructurados y se generan embeddings semánticos mediante el encoder, resultando en una matriz de 300 filas por 384 columnas que captura representaciones densas del contenido. Esta fase se ejecuta una sola vez, consumiendo recursos mínimos en hardware estándar, y habilita búsquedas posteriores sin re-cómputo, promoviendo escalabilidad para conjuntos mayores.

4.2. Flujo de Consulta en Tiempo Real

Ante una consulta de usuario, como "Tengo pollo y arroz. Quiero algo fácil tipo sopa, rápido.", el proceso sigue cinco pasos secuenciales alineados con la arquitectura RAG:

1. **Embedding de la Consulta**: Se computa el vector de la pregunta mediante el encoder, produciendo un vector de 384 dimensiones en menos de un segundo.

2. **Búsqueda de Similitud**: Se calcula la similitud coseno entre la consulta y los embeddings indexados, evaluando eficientemente las 300 recetas.

3. **Recuperación de Top-K**: Se seleccionan los índices top-3, priorizando relevancia semántica. Para la consulta ejemplar, se obtienen puntuaciones de 0.768 para "chicken noodle soup", 0.685 para "pea soup" y 0.679 para "beef barley soup", demostrando captura efectiva de conceptos como "sopa", "pollo" y "fácil". En pruebas adicionales con consultas variadas, como "receta con carne y vegetales saludable", se recuperan recetas como "beef stir-fry" con scores superiores a 0.7, destacando la robustez semántica ante variaciones lingüísticas.

4. **Construcción del Prompt**: Se integra el contexto recuperado en un prompt estructurado, concatenado con la pregunta y un system prompt restrictivo para minimizar alucinaciones. El prompt total oscila entre 1.000 y 2.000 tokens, bien dentro del límite de 4.096 de LLaMA 2.

5. **Generación con LLM**: Se invoca LLaMA 2 7B, un decoder-only con 32 capas, RoPE, masking causal y activación SwiGLU, vía la API Ollama. Esto genera respuestas conversacionales precisas en 5-10 segundos, basadas exclusivamente en el contexto proporcionado.

En términos de resultados, el sistema exhibe alta precisión en recuperaciones, con scores promedio de 0.65-0.75 para consultas relevantes en un conjunto de prueba de 20 queries, reflejando una alineación semántica efectiva que supera búsquedas keyword-based. Por ejemplo, consultas ambiguas como "algo dulce con frutas" recuperan "fruit salad" con score 0.712, permitiendo respuestas personalizadas que integran ingredientes disponibles. Comparativamente, LLaMA 2 sin RAG produce outputs genéricos o erróneos en el 70 % de casos, con alucinaciones como pasos inventados, mientras que RAG reduce esto a menos del 10 % mediante restricción contextual, validado por evaluación manual. Métricas adicionales incluyen latencia total inferior a 10 segundos y recall@3 superior a 0.8 para recetas matching, demostrando superioridad en precisión y

actualizabilidad. Limitaciones incluyen la búsqueda lineal en NumPy, que escala pobremente más allá de miles de recetas, y la no fragmentación de textos largos; sin embargo, la arquitectura facilita migraciones a FAISS para búsquedas optimizadas, persistencia y manejo de millones de vectores, mejorando eficiencia en escenarios productivos.

5. Conclusión

Este proyecto ilustra la eficacia de RAG en dominios específicos como la cocina, amalgamando principios teóricos como arquitectura encoder-decoder y recuperación semántica con ejecución práctica que resuelve problemas reales de LLMs. Al superar al fine-tuning en eficiencia computacional, flexibilidad y costo —evitando reentrenamientos masivos y permitiendo actualizaciones dinámicas—, RAG emerge como una solución óptima para aplicaciones donde el conocimiento evoluciona rápidamente, como bases de recetas expansibles. Los resultados no solo confirman una reducción significativa en alucinaciones y mejora en relevancia, sino que también destacan el potencial para escalabilidad, transparencia (mediante scores verificables) y adaptabilidad multilingüe gracias al encoder seleccionado. En perspectiva, este diseño pavimenta el camino para integraciones avanzadas, como bases vectoriales distribuidas o modelos con contextos extendidos, ampliando su aplicabilidad a dominios como educación o e-commerce. En suma, el proyecto valida RAG como un paradigma inteligente y sostenible, fomentando sistemas híbridos que combinan recuperación externa con generación interna para outputs precisos, verificables y usuario-centrados, con implicaciones amplias en la IA aplicada.

6. Referencias

- Hugging Face. (2023). Llama 2. Retrieved from https://huggingface.co/docs/transformers/en/model_doc/llama2
- IONOS. (n.d.). Paraphrase Multilingual MPNet v2. Retrieved December 11, 2025, from https://docs.ionos.com/cloud/ai/model-hub/models/embedding-models/paraphrase-multilingual-mpnet-v2
- Meta. (n.d.). meta-llama/Llama-2-7b. Hugging Face. Retrieved December 11, 2025, from https://huggingface.co/meta-llama/Llama-2-7b
- Pinecone. (n.d.). Sentence Transformers: Meanings in Disguise. Retrieved December 11, 2025, from https://www.pinecone.io/learn/series/nlp/sentence-embeddings/
- sentence-transformers. (n.d.). Paraphrase Multilingual Mpnet Base V2. Dataloop. Retrieved December 11, 2025, from https://dataloop.ai/library/model/sentence-transformers_paraphrase-multilingual-mpnet-base-v2/