Informe Técnico: Implementación de Pipeline RAG para Análisis de Noticias Delictuales en Chile (2022-2025)

Integrantes: Renato Atencio, Tomás Contreras

1. Objetivos y Contexto

1.1 Contexto del Proyecto

El problema abordado consiste en la necesidad de procesar, consultar y extraer información relevante de un corpus de noticias relacionadas con robos violentos ocurridos en Chile entre los años 2022 y 2025.

Dado el volumen de información y la especificidad de las consultas requeridas (búsquedas semánticas y filtrado por metadatos), se optó por una arquitectura basada en RAG (Retrieval-Augmented Generation). Esta técnica permite mejorar la precisión de los Modelos de Lenguaje Grande (LLM) al proporcionarles contexto externo recuperado dinámicamente.

1.2 Objetivos

1. **Objetivo General:** Implementar un pipeline RAG funcional que permita realizar consultas en lenguaje natural sobre una base de datos de noticias periodísticas.
2. **Objetivos Específicos:**
    - Vectorizar un conjunto de 342 documentos periodísticos para permitir búsquedas por similitud semántica.
    - Implementar una arquitectura de ejecución 100% local para garantizar la privacidad de los datos y eliminar costos operativos asociados a APIs de terceros.
    - Integrar mecanismos de re-ranking para optimizar la relevancia de la información suministrada al LLM, dado el límite de contexto de los modelos locales.

2. Arquitectura del Sistema

El sistema fue diseñado priorizando la soberanía de los datos y la ejecución local. A diferencia de soluciones comerciales que dependen de la nube, esta arquitectura utiliza

hardware local para ejecutar tanto el almacenamiento vectorial como la inferencia del modelo de lenguaje.

2.1 Componentes Principales

- **Gestión de Datos (Preprocessing):** Se utiliza Pandas para la manipulación del dataset CSV. Se generó un campo sintético denominado rag_text que concatena el título, cuerpo de la noticia, país, medio, fecha y polaridad para enriquecer la representación semántica de cada vector.
- **Base de Datos Vectorial:** Se implementó ChromaDB en modo persistente. Esta base de datos almacena los embeddings de los documentos y sus metadatos asociados, permitiendo búsquedas híbridas (semánticas + filtros de metadatos) utilizando la similitud del coseno (hnsw:space: cosine).
- **Interfaz de Inferencia LLM:** Se utilizó LM Studio actuando como servidor local en el puerto 1234. Esta herramienta expone una API compatible con el formato de OpenAI, permitiendo utilizar librerías estándar de conexión (client = OpenAI(base_url="http://localhost:1234/v1")) pero redirigiendo el cómputo a la GPU/CPU local.

3. Técnicas TAL Utilizadas

Para lograr una recuperación precisa y una generación de texto coherente, se orquestaron tres modelos de Inteligencia Artificial distintos dentro del pipeline:

3.1 Generación de Embeddings (Sentence Transformers)

Se seleccionó el modelo paraphrase-multilingual-mpnet-base-v2.

- **Justificación:** Es un modelo basado en la arquitectura Transformer entrenado específicamente para generar representaciones vectoriales densas (768 dimensiones). Su capacidad multilingüe lo hace idóneo para textos en español, capturando relaciones semánticas, contexto y orden de palabras mejor que modelos léxicos tradicionales.
- **Proceso:** Los 342 documentos fueron procesados en lotes (batch_size=32) y normalizados para optimizar el cálculo de similitud.

3.2 Re-ranking Semántico (Cross-Encoders)

Se integró el modelo cross-encoder/ms-marco-MiniLM-L-6-v2 como paso intermedio de refinamiento.

- **Función Técnica:** Los modelos LLM locales pequeños suelen tener ventanas de contexto limitadas (aceptando solo 3 a 5 textos de entrada). La búsqueda vectorial inicial recupera un número amplio de candidatos (x resultados), pero estos pueden no estar perfectamente ordenados por relevancia directa para la pregunta específica.
- **Implementación:** El Cross-Encoder evalúa pares (Query, Documento) y asigna un puntaje de relevancia preciso. El sistema reordena los candidatos y selecciona solo los y documentos con mayor puntaje para enviarlos al LLM, maximizando la eficiencia del contexto.

3.3 Generación de Respuesta (LLM)

Se utilizó el modelo gpt-oss-20b (desplegado vía LM Studio).

- **Prompt Engineering:** Se diseñaron funciones de consulta (rag_query_simple y rag_query_simple_filter) que construyen un prompt estricto: "Responde la pregunta usando EXCLUSIVAMENTE el siguiente contexto". Esto reduce las alucinaciones y fuerza al modelo a basarse únicamente en la evidencia recuperada.

4. Resultados y Ejemplos

El sistema demostró capacidad para sintetizar información dispersa y aplicar filtros de metadatos.

4.1 Caso de Estudio 1: Reincidencia Juvenil

- **Consulta:** "Delincuente menor de edad es detenido nuevamente".
- **Recuperación:** El sistema identificó y priorizó documentos sobre la fuga de un menor desde un ala de psiquiatría y su posterior detención por robo.
- **Respuesta del Sistema:** El modelo generó una respuesta coherente citando la noticia "Menor fugado del hospital fue detenido tras cometer un nuevo robo", demostrando capacidad para asociar conceptos de "detención", "menor" y "reincidencia".

4.2 Caso de Estudio 2: Delitos Graves con Filtro Geográfico

- **Consulta:** "asesinato durante intento de robo" (Filtro aplicado: country: chile).
- **Análisis:** El sistema recuperó noticias sobre el homicidio de la carabinera Rita Olivares y crímenes asociados a préstamos "gota a gota".
- **Capacidad Analítica:** Ante la pregunta sobre qué zona se menciona más, el modelo analizó los textos recuperados y concluyó correctamente que la zona de Limache aparecía con mayor frecuencia en el contexto de los antecedentes penales y operativos citados.

5. Discusión Ética y Social

5.1 Privacidad y Soberanía de Datos

Una decisión crítica del proyecto fue evitar el uso de APIs comerciales como las de OpenAI o Google. Al utilizar LM Studio y modelos Open Source, se garantiza que ningún dato (incluyendo potenciales datos sensibles en las noticias o metadatos de los usuarios) salga del entorno local. Esto protege contra la recolección de datos por parte de grandes tecnológicas y asegura la confidencialidad de la investigación.

5.2 Reducción de Barreras de Entrada

Este proyecto demuestra que el despliegue de sistemas de análisis avanzado no requiere necesariamente de grandes presupuestos operativos. La utilización de modelos abiertos y ejecución local **viabiliza** y reduce las barreras de acceso a tecnologías de TAL complejas para estudiantes e instituciones con recursos limitados, democratizando la capacidad de auditar y analizar grandes volúmenes de información pública.

5.3 Limitaciones y Sesgos

Si bien el sistema es funcional, el uso de un modelo cuantizado y más pequeño (gpt-oss-20b) implica una menor capacidad de razonamiento comparado con modelos SOTA (State of the Art) comerciales. Esto puede derivar en una menor precisión en inferencias complejas. Además, al trabajar con noticias de crímenes, existe el riesgo ético inherente de que el modelo pueda amplificar sesgos presentes en la redacción periodística original si no se supervisa adecuadamente.

6. Conclusiones

1. **Viabilidad de RAG Local:** Se ha demostrado exitosamente la implementación de un pipeline RAG completo sin dependencias de internet ni costos por token. La combinación de ChromaDB y LM Studio ofrece una plataforma robusta para prototipado académico y aplicaciones privadas.
2. **Importancia del Re-ranking:** En entornos locales con modelos de lenguaje más pequeños, la etapa de re-ranking (usando Cross-Encoders) demostró ser crítica. Sin este paso, la capacidad limitada de contexto del LLM se llenaría con información ruidosa, degradando la calidad de la respuesta final.
3. **Costo Computacional:** Aunque se eliminan los costos financieros, el costo se traslada al hardware. La ejecución simultánea de modelos de embedding, re-ranking y generación conlleva una alta demanda computacional, lo cual es un factor limitante para la escalabilidad del sistema en equipos de consumo estándar.