
UNIVERSIDAD AUSTRAL DE CHILE

Facultad de Ciencias de la Ingeniería, Ingeniería Civil Informática



Proyecto de TAL: Sofia Search

Tratamiento Automatico del Lenguaje

Segundo Semestre, 2025

Docente responsable

Matthieu Vernier

Alumnos

Alann Kahler, alann.kahler@alumnos.uach.cl



Índice

1. Objetivos y Contexto	3
2. Arquitectura del sistema	4
3. Técnicas TAL utilizadas	5
4. Resultados y ejemplo	6
5. Discusión ética y social	7
6. Conclusiones	8



1. Objetivos y Contexto

El siguiente documento tiene como objetivo abarcar sobre la creacion y experimentacion de un prototipo RAG.

Un prototipo RAG (Retrieval-Augmented Generation) es una version basica(a menor escala) de lo que podria llegar a ser un Sistema RAG.

El significado de las siglas RAG significan:

1. Retrieval: Busca y recibe los datasets(textos que contienen datos) mas relevantes para la pregunta realizada.
2. Augmented: Se recopila el prompt y la información recopilada para ser enviada al modelo generativo(GPT, GEMINI, ETC...).
3. Generation: El modelo genera su respuesta el cual es desplegada al usuario.

Este prototipo RAG se diseñará en base a las combinación de técnicas TAL vistas en la asignatura INFO-279:

- Procesamiento del lenguaje natural.
- Recuperación semántica de información.
- Agentes inteligentes.
- Generación automática de texto.



2. Arquitectura del sistema

El prototipo RAG fue diseñado en un entorno virtual (.venv) en el sistema operativo Linux. Se usaron diversas librerías (Pandas, spaCy, etc.) las cuales vienen incluidas en el entorno virtual.

Este prototipo RAG fue implementado mediante módulos, cada uno con su propia función, de manera que se mantenga el orden y sea fácil su entendimiento:

- **main.py:** Es el script principal. Verifica si el indice esta al dia (Indice que nace del CSV), hace una reconstrucción de éste si cambio y elige el proveedor (Gemini o Groq en su defecto). Por ultimo abre el modo chat para poder hacer las consultas
- **ingestion.py:** Carga el CSV, une título con texto, parsea la fecha y extrae región (o ciudad) con Spacy para crear detected_region
- **indexing.py:** Genera un embeddings con sentence-transformers, crea el índice y lo guarda como new_index.faiss y los metadatos los guarda como new_metadata.pkl
- **search.py:** Es un agente de consulta, interpreta los filtro básicos (región y fecha) y recupera los top-k más relevantes.
- **generation.py:** Genera el prompt en base a las noticias extraídas y cita fuentes. Primeramente usa Gemini, si este falla, cambia a Groq automáticamente.
- **check_models.py:** Verifica que existe la GOOGLE_API_KEY. En caso que sea así, configura al cliente.

También se incluye -externamente a la arquitectura, pero relacionada con ella- un .env. Este .env contiene las keys de Gemini y Groq.



3. Técnicas TAL utilizadas

Para poder crear modelar este prototipo RAG, se utilizaron diferentes tecnicas TAL. Estas son:

1. **Procesamiento del lenguaje:** Se normalizo el texto y se extrajo la ubicacion con NER(Usando Spicy)
2. **Recuperacion semantica:** Se uso embeddings con sentence-transformers + FAISS para poder tener una representacion de texto. Tambien se uso top-k para la busqueda de metadatos.
3. **Agentes inteligentes:** Se usaron heurísticas de enrutado que detecta intenciones(region, fecha relativas) para generar filtros antes de buscar
4. **Generacion de texto:** Se usa una generacion controlada(RAG) en el cual se arma un prompt con contexto recuperado, citando fuentes y uso de LLM instruccional (Gemini primeramente, si falla, Groq)



4. Resultados y ejemplo

Tenemos un sistema RAG funcional para un dataset de noticias (en este caso Chile, septiembre 2025) que:

- Limpia y enriquece las noticias (fecha normalizada, región/ciudad detectada con NER).
- Indexa todo con embeddings y FAISS para búsquedas semánticas rápidas.
- Interpreta la consulta (región/fecha) con un agente ligero y trae los top-k más relevantes.
- Genera la respuesta citando las fuentes con un LLM (Gemini, con respaldo automático a Groq) y reintentos ante límites de cuota.
- Se actualiza solo si el dataset cambia y permite usarlo por CLI en modo chat u one-shot.

Algunos ejemplos del sistema interactuando con distintos prompts serían:

1. **Prompt:** ¿Cómo fue la inflación del mes anterior, agosto?

Respuesta: La inflación del mes de agosto de 2025 fue de 0,0 %, registrando la primera variación nula del año. Esto llevó a que el indicador acumulara un 2,9 % en el año y 4,0 % a 12 meses respecto de la serie empalmada [1].

2. **Prompt:** ¿Cuánta gente salió de la Región Metropolitana por las Fiestas Patrias?

Respuesta: Un total de 1.050.000 vehículos salieron de la Región Metropolitana durante el fin de semana de Fiestas Patrias [3].

Inicialmente, el 17 de septiembre, se reportó que más de 750.000 vehículos habían dejado la RM [6]. Para el 19 de septiembre, la cifra aumentó a más de 860.000 vehículos [2].

3. **Prompt:** ¿Cuántas infracciones a la ley hubo en la semana de las Fiestas Patrias?

Respuesta: Durante las celebraciones de Fiestas Patrias se cursaron cerca de 3.000 infracciones, asociadas principalmente a causalidades que contribuyen a siniestros viales con consecuencias fatales [3]. De estas, 2.086 estuvieron asociadas a la velocidad, 636 al no uso del cinturón de seguridad y 278 al sistema de retención infantil [3].



5. Discusión ética y social

El modelo no debe ser tomado como una fuente 100 % verídica de información, ya que si el contexto otorgado es débil, puede inventar (alucinar) datos. Por lo mismo, el sistema RAG debe incluir obligatoriamente un sistema de citas para que el usuario pueda investigar por sí mismo si la información está bien respaldada.

También se debe tener en cuenta que palabras de carácter sensible (como, por ejemplo, términos despectivos o discriminatorios) pueden pasar por alto, ya que el sistema no tiene ningún filtro. Aun así, es necesario aclarar que los LLM usados tienen filtros, por lo que no es probable que suceda.

Por último, aclarar que el sistema no es 100 % objetivo. Si los diferentes noticieros de los cuales se compuso el dataset tienen sesgos y/o agendas políticas, ideológicas o religiosas, la información proporcionada por el prototipo RAG será subjetiva. Esto, al igual que el primer punto, se combate con las citas y el pensamiento crítico del usuario.



6. Conclusiones

El desarrollo de este prototipo RAG permitió comprender de manera práctica cómo integrar diversas técnicas del Tratamiento Automático del Lenguaje para construir un sistema capaz de recuperar, procesar y generar información de forma coherente y relevante.

A lo largo del proyecto se logró implementar un flujo completo: desde la ingestión y enriquecimiento de noticias, pasando por la indexación semántica y los mecanismos de búsqueda, hasta la generación final de respuestas con respaldo de fuentes. Este enfoque evidenció la importancia de separar la arquitectura en módulos claros, facilitando el mantenimiento, la escalabilidad y la capacidad de auditoría del sistema.

Asimismo, se identificaron limitaciones inherentes a los modelos generativos, como la posibilidad de errores, invenciones o sesgos provenientes tanto del modelo como del propio dataset. Esto refuerza la necesidad de incorporar citas, filtros y mecanismos de validación que permitan al usuario evaluar de forma crítica la información producida.

En conjunto, el prototipo demuestra que un sistema RAG bien diseñado puede mejorar significativamente la precisión y confiabilidad de un modelo generativo, especialmente en escenarios donde es fundamental basarse en datos actualizados. Aunque aún es un sistema de pequeña escala, establece una base sólida para futuras extensiones, optimizaciones y aplicaciones en entornos reales.