TUIA NLP 2024
TRABAJO PRÁCTICO FINAL

Pautas generales:

- El trabajo deberá ser realizado de forma individual.
- Se debe entregar un informe en PDF en el cual se incluya las justificaciones y explicación de lo realizado en código.
- Para la entrega del trabajo práctico 2 (TP2), se deberá utilizar el repositorio individual creado para la materia, de la misma manera que se hizo con el trabajo práctico 1 (TP1). En el repositorio, deberá crearse una carpeta específica para el TP2. El código debe entregarse en un cuaderno **Jupyter** ejecutado, acompañado de un informe en formato **PDF** que incluya:

  - Carátula

  - Introducción

  - *Ejercicio 1:* - Resumen

    - Desarrollo detallado de los pasos para la solución del problema, y justificaciones correspondientes.

    - Conclusiones

    - Enlaces a los modelos y librerías utilizados

  - *Ejercicio 2:*

    - Resumen

    - Desarrollo detallado de los pasos para la solución del problema, y justificaciones correspondientes.

    - Conclusiones

    - Enlaces a las herramientas utilizadas

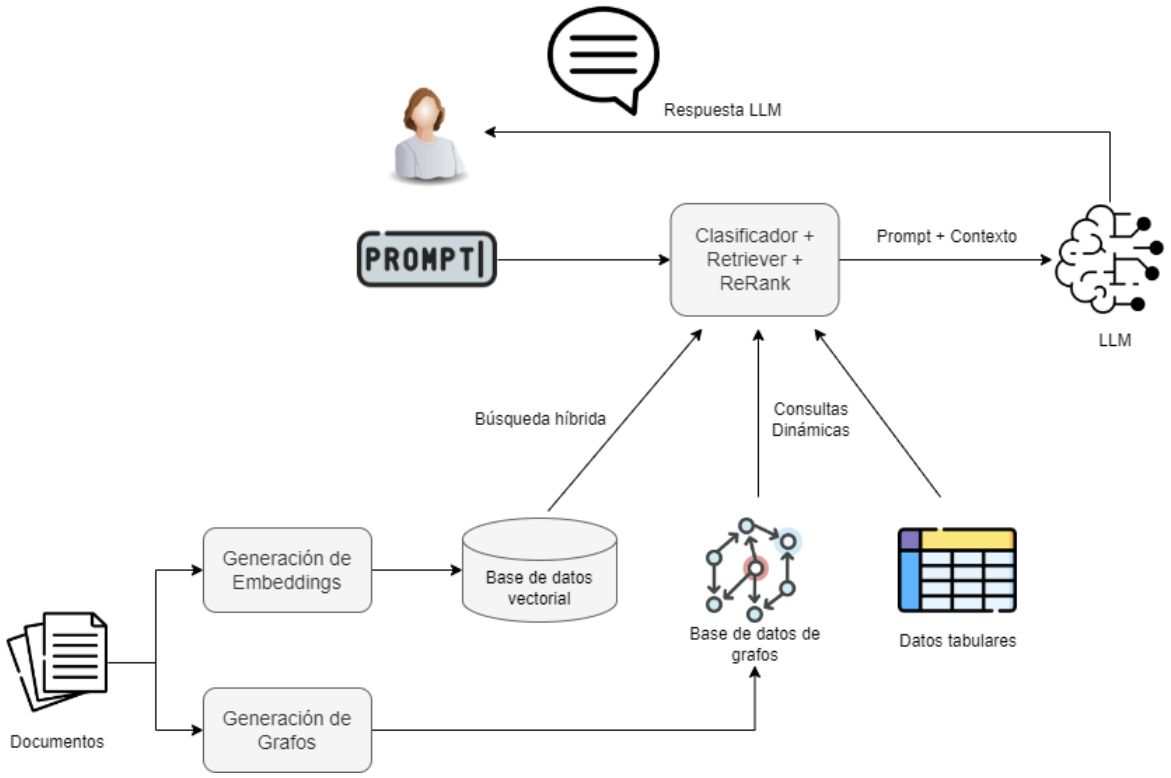
    .

  Las modificaciones en el repositorio deberán realizarse antes de las **00:00 horas del día 16/12/2024**.

Ejercicio 1 - RAG

Crear un chatbot experto en un juego de mesa estilo Eurogame (ver Anexo) que le fue asignado, usando la técnica RAG (Retrieval Augmented Generation). Como fuentes de conocimiento se utilizarán al menos las siguientes fuentes:

- Documentos de texto
- Datos numéricos en formato tabular (por ej., Dataframes, CSV, sqlite, etc.)
- Base de datos de grafos (Online o local)



El sistema debe poder llevar a cabo una conversación en lenguaje español o inglés. El usuario podrá hacer preguntas, que el chatbot intentará responder a partir de datos de algunas de sus fuentes. El asistente debe poder clasificar las preguntas, para saber qué fuentes de datos utilizar como contexto para generar una respuesta. Se espera que las respuestas del chatbot sean en el mismo lenguaje de la consulta del usuario.

Requerimientos y consideraciones generales

- Realizar todo el proyecto en un entorno Google Colab. El cuaderno completo debe correr sin errores para ser aceptado. Además, vínculos a documentos, datasets o cualquier recurso externo, debe ser obtenido desde el código, sin tareas manuales de subir documentos a Colab.

- El conjunto de datos debe tener al menos 100 páginas de texto y un mínimo de 3 documentos.

- Realizar split de textos usando Langchain (RecursiveTextSearch, u otros métodos disponibles).

- Verificar la calidad de los chunks y limpiar el texto extraído si lo considera conveniente.

- Realizar los embeddings que permitan vectorizar el texto y almacenarlo en una base de datos ChromaDB.

- Los modelos de embeddings y LLM para generación de texto son a elección.

- Desarrollo de dos versiones del “Clasificador”:
    - 1) basado en LLM (Unidad 6)
    - 2) basado en un modelo entrenado con ejemplos y embeddings (Unidad 3)

  Comparar los resultados y seleccionar el clasificador que mejor funcione. Justificar la decisión.

- La query a la base de datos de grafos, debe ser dinámica, recuperando solamente elementos relacionados con el prompt. No está permitido traer **siempre todo** el contenido de la base de datos e incorporarlo como contexto en la aumentación, ya que eso resulta ineficiente y podría consumir muchos tokens. La query puede hacerse con SPARQL, Cypher o similar.

- En el mismo sentido, la recuperación de los datos tabulares, debe ser a través de una query dinámica (Por ejemplo con SQL, o filtros de Pandas), y no se debe incorporar tablas completas a la ventana de contexto. Se deben entonces buscar solamente los elementos relevantes de la tabla referidos al prompt.

- Para crear el contenido de la base de datos de grafos, se debe extraer el conocimiento desde una fuente de texto o tabular, y generar luego las tríadas para insertar en la base de datos.

- La utilización de reglas fijas en el código para la búsqueda de palabras claves en los prompts, suelen quitar flexibilidad al sistema. Se valora la utilización de técnicas de análisis de texto flexibles, de tipo POS, NER, Lematización, o ingeniería de prompts, para evitar palabras “hardcoded” en la aplicación.

- Además de las tres fuentes requeridas, pueden añadirse fuentes de información adicionales a través de APIs públicas (por ejemplo buscar en internet con Serper, SerpAPI, etc.) si el caso lo justifica.

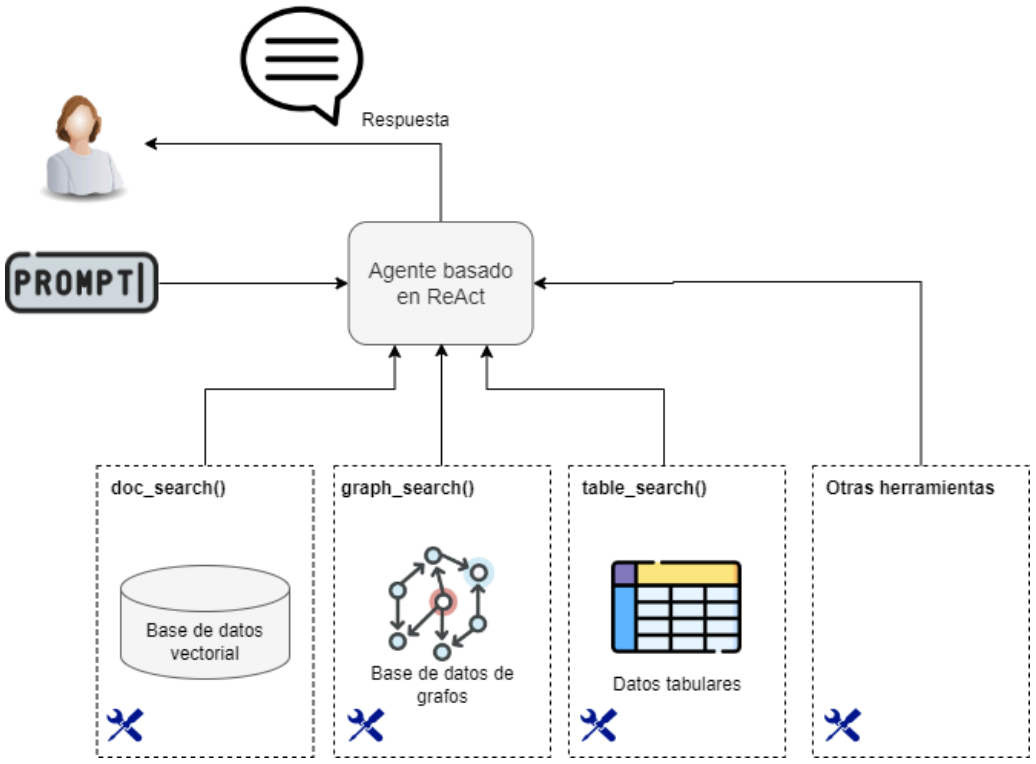
- Todos los pasos del RAG deben ser implementados en Colab de forma transparente. No se permite la utilización de librerías o aplicaciones adicionales que realizan todo o parte del proceso de RAG (por ejemplo ragapp, ragflow, llmware, cognita, etc.). Se pretende que el trabajo demuestre haber aprendido los conceptos que forman parte de la materia. El ejercicio puede resolverse con las herramientas vistas en la cátedra, sin recurrir a otras herramientas de alto nivel.

- Retriever sobre los documentos:
    - Se debe realizar una búsqueda híbrida sobre los documentos (Semántica y por palabras claves)
    - Se debe implementar algún mecanismo de ReRank visto en la teoría

Ejercicio 2 - Agente

Este ejercicio se basa en el ejercicio 1, e incorpora el concepto de Agente, basado en el concepto ReAct. Nuestro agente debe cumplir con los siguientes requisitos:

- Utilizar al menos 3 herramientas, aprovechando el trabajo anterior:
    - doc_search(): Busca información en los documentos
    - graph_search(): Busca información en la base de datos de grafos
    - table_search(): Busca información sobre los datos tabulares
- Se puede implementar alguna nueva herramienta que se considere necesaria y que pueda enriquecer las capacidades del agente.
- Utilizar la librería Llama-Index para desarrollar el agente:
    - `llama_index.core.agent.ReActAgent`
    - `llama_index.core.tools.FunctionTool`
- Se debe construir el prompt adecuado para incorporar las herramientas al agente ReAct



Presentar en el informe los resultados:

- Presentar 5 ejemplos de prompts donde se deba recurrir a más de una herramienta para responder al usuario. Evaluar los resultados obtenidos
- Explicar con 3 ejemplos, donde el agente falla o las respuestas no son precisas.
- Explicar cuáles son las mejoras que sería conveniente realizar para mejorar los resultados.

Anexo:

En el siguiente link podrá entender que es un Eurogame.

Materiales:
A continuación se detallan los links correspondiente al juego en la BBG (Board Game Geek)

Rajas de Ganges
Ruinas perdidas de Arnak
Viticultura
The white castle

Nota 1: Para la base de datos de grafos se puede pensar una estructura de nodos y aristas que permita, por ejemplo, modelar la producción e interrelación entre los diseñadores y artistas de los juegos. Si clickea sobre alguno de los nombres de los creadores (los verá dentro del link del juego correspondiente), puede encontrar datos que le serán útiles en esta tarea.

Nota 2: También explorando en el link del juego podrá encontrar recursos en varios idiomas. clickeando en las siguientes etiquetas :

- Files (donde encontrará material escrito y visual, resúmenes de reglas, etc.)
- Videos (Reviews en distintos idiomas. Puede filtrar por idioma)
- Forum (Un foro donde se discuten reglas y particularidades)
- Stats: valores que puede usar para generar alguno de los recursos para el tp (como fuentes tabulares)
-

EXPLORE Y EXTRAIGA LOS TEXTOS PARA DAR FORMA A LAS BASES Y FUENTES DE DATOS QUE CONSTITUYEN EL RAG

También se facilita un blog con análisis de juegos, donde puede encontrar la reseña del juego y opinión de autor del blog sobre el mismo (Esto es un recurso que puede utilizar junto a otros que le ayuden en la tarea).

Rajas de Ganges
Ruinas perdidas de Arnak
Viticultura
The white castle