TUIA NLP 2024
TRABAJO PRÁCTICO FINAL

07/08/2024

Integrante:
Mateo Gravi Fiorino

Profesores:
Alan Geary
Juan Pablo Manson

Ejercicio 1.

Introducción

En este trabajo, desarrollaremos un chatbot experto en mitología griega utilizando la técnica de Retrieval Augmented Generation (RAG).

Este chatbot será capaz de responder preguntas de los usuarios basándose en diferentes fuentes de información: documentos de texto, datos numéricos en formato tabular y una base de datos de grafos.

El sistema estará diseñado para mantener conversaciones tanto en español como en inglés, y clasificará las preguntas para determinar qué fuentes de datos usar como contexto para generar respuestas precisas.

Desarrollo Detallado de los Pasos

Carga de librerías.

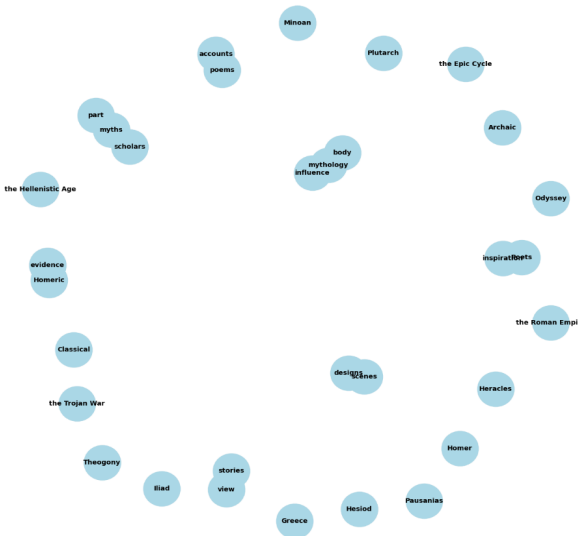
Importar las bibliotecas necesarias: Langchain, OpenAI, ChromaDB, Pandas, NetworkX.

Obtención y Preparación de las Fuentes de Conocimiento

Documentos de Texto: Se utiliza el pdf "dioses_heroes"

Datos Numéricos en Formato Tabular: Un archivo de generación propia con información brindada en internet.

Base de Datos de Grafos: Generada con WIKIDATA



Preprocesamiento de Texto y Split en Chunks

Uso de la libreria Langchain para aplicar la técnica Recursive Character Text Splitter. Podemos visualizar los chunks.

```
Chunk 0:
Y
 otra. Y otra. Siempre los mitos
griegos (y romanos, que son más o menos
los mismos con otros nombres). Porque
son extraños y maravillosos, pero también
familiares y cercanos. Porque están vivos.
Porque seguimos hablando de ellos,
porque los tenemos incorporados al
idioma (¿acaso a un hombre forzudo no se
lo llama
un
```

Luego se realiza una limpieza del texto para posteriormente poder aplicar embeddings.

Embeddings y Almacenamiento en ChromaDB

Se aplican los embeddings a las fuentes de datos correspondientes, utilizando el modelo "text-embedding-3-small" de openAI. Este proceso nos permitirá almacenar los embeddings generados en una base vectorial de ChromaDB

MODEL	~ PAGES PER DOLLAR	PERFORMANCE ON MTEB EVAL	MAX INPUT
text-embedding-3-small	62,500	62.3%	8191

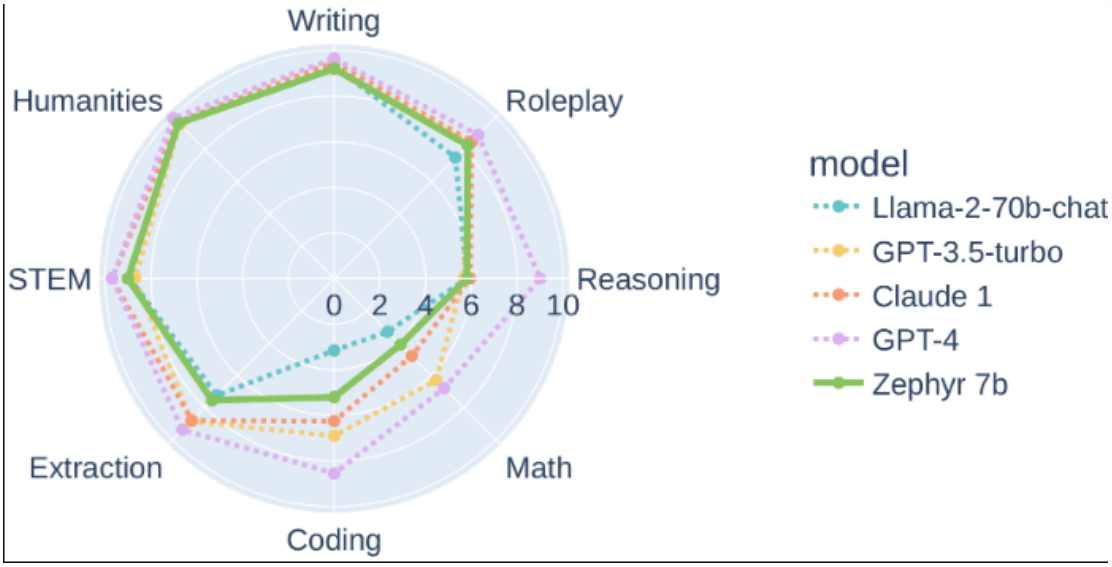
Base de datos Vectorial

Se almacenan los embeddings generados anteriormente en una base de datos vectorial que brinda ChromaDB.

Esto nos permitirá en los próximos pasos, poder hacer búsquedas de cercanías para que el modelo LLM pueda generar una respuesta apropiada.

Modelo basado en Ejemplos

Para este paso del trabajo, implementamos las plantillas proporcionadas por `jinja2`.
Gracias a esta librería podemos utilizar ejemplos específicos de diálogo para entrenar y guiar al modelo de lenguaje en la generación de respuestas.
Se utiliza el modelo de Hugging Face Zephyr.



Podemos ver que es un modelo muy potente y se puede comparar con otros modelos.

A continuación podemos ver un ejemplo de esta plantilla de ejemplos proporcionada:

```
{
    "role": "user",
    "content": "Who is Zeus?"
},
{
    "role": "assistant",
    "content": "Zeus is the king of the gods in Greek mythology."
```

Esté modelo tiene 3 grandes ventajas

- ☐ ***Relevancia:*** Ayuda a mantener las respuestas del modelo relevantes y específicas al dominio.
- ☐ ***Consistencia:*** Proporciona una guía clara al modelo, resultando en respuestas más coherentes.
- ☐ ***Control***

Modelo basado en LLM

Para el modelo elegí gpt-3.5-turbo una opción rápida, eficaz y económica que brinda OpenAI.

MODEL	DESCRIPTION	CONTEXT WINDOW	MAX OUTPUT TOKENS	TRAINING DATA
gpt-3.5-turbo-0125	The latest GPT-3.5 Turbo model with higher accuracy at responding in requested formats and a fix for a bug which caused a text encoding issue for non-English language function calls. Learn more.	16,385 tokens	4,096 tokens	Up to Sep 2021

Para esta parte necesitamos la base de datos vectorial declarada almacenada en ChromaDB.

Este asistente virtual, utiliza una combinación de embeddings de texto, recuperación de datos de una base de datos vectorial (ChromaDB) y una base de datos de grafos local.

El código importa las bibliotecas necesarias y carga un modelo de Spacy para procesamiento de lenguaje natural. Se definen funciones para obtener embeddings de texto usando el modelo de OpenAI y funciones para recuperar textos relevantes de ChromaDB basados en una consulta.

Para la base de datos de grafos se define una función para extraer nodos y relaciones relevantes de un grafo basado en una consulta.

Descubrí la **importancia de las instrucciones** dadas al modelo para el mejor y óptimo funcionamiento del modelo. Las instrucciones dadas para el asistente sobre mitología griega son:

introduction = """
OBJETIVO PRINCIPAL

Eres un experto en mitología griega y tu objetivo es resolver preguntas relacionadas con el tema. Usa los documentos proporcionados para contestar la pregunta.

CARACTERÍSTICAS DEL LLM

- Rol Profesional: Experto en mitología griega.

- Tienes una alta capacidad de autoevaluación.

- Te esfuerzas por identificar y contextualizar las preguntas principales del usuario.

ESTRUCTURA DE LA RESPUESTA

1. **Presentación:**

- Comienza con: "Hola, soy tu asistente especializado en mitología griega."

2. **Respuesta:**

- Responde la pregunta de manera detallada y específica, siguiendo un enfoque paso a paso.

- Si no tienes el contexto, responde: "No tengo suficiente información para responder a esta pregunta. Consulta otras fuentes o pregunta de nuevo con más detalles."

3. **Referencia al Documento:**

- Incluye el nombre del documento de donde obtuviste la información.

"""

Formando una instrucción en formato **MARKDOWN** entendí que para modelos de openAI es una muy buena estrategia.
Se establece el objetivo principal del asistente, debe ser claro y conciso.

Además se brindan unas características principales para guíar al modelo en la generación de respuestas.

Por último, se define la estructura de la respuesta esperada por el asistente.

Mediante ensayos de prueba y error pude concluír en que estas instrucciones funcionan adecuadamente para este modelo de openAI.

En resumen, el funcionamiento se basa en la obtención de embeddings de la consulta realizada por el usuario.
Luego se utiliza RAG para la base de chromadb para encontrar texto relevante basados en esa consulta y para la base de grafos se utiliza spicy para encontrar relaciones relevantes.

Por último. Se combina la información y es formateada para el modelo GPT para poder ser utilizada y generar un mensaje de respuesta.

Conclusión

Para finalizar este ejercicio, podemos ver buenas respuestas a diferentes preguntas en ambos modelos.

 # Ejemplo
prompt = '¿Cual es la historia de Afrodita?'
answer = choose_data_source(prompt, api_key="apikey")
print('\Respuesta del modelo: \n', answer)

\Respuesta del modelo:

Respuesta:
"Mito: La historia de Afrodita es una leyenda de la mitología griega que narra la creación y las aventuras de la diosa del amor y la belleza, Afrodita."

 # Ejemplo
prompt = 'Quien fue hercules'
answer = choose_data_source(prompt,
api_key="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")
print('\Respuesta del modelo: \n', answer)

\Respuesta del modelo: ---------- Your answer could be: "Deity: Heracles (Hercules in Latin)"

Para el modelo gpt-3.5-turbo de openAI, podemos ver buenos resultados:

 # Uso del código

query = **"¿Quién es Zeus en la mitología griega?"**
response = ask(query, collection, graph)
print(response)

Hola, soy tu asistente especializado en mitología griega.

Zeus es el rey de los dioses en la mitología griega. Es conocido como el dios del cielo y el trueno, y su equivalente en la mitología romana es Júpiter. Zeus es hijo de Cronos y Rea, y logró derrocar a su padre para convertirse en el gobernante supremo del Olimpo.

Zeus es representado como un dios poderoso, con una personalidad dominante y a menudo impulsiva. Es conocido por lanzar rayos y truenos desde el cielo, y su símbolo principal es el rayo. Además de ser el dios del cielo y el trueno, Zeus también es considerado el protector de la ley, el orden y la justicia en el mundo de los dioses y de los mortales.

Referencia al Documento: Mitología Griega - Zeus, el rey de los dioses.

 # Ejemplo 2
Quien fue hercules

Hola, soy tu asistente especializado en mitología griega.\n\nHércules, también conocido como Heracles en la mitología griega, fue un héroe y semidiós famoso por su fuerza sobrehumana y sus doce trabajos. Era hijo de Zeus, el rey de los dioses, y Alcmena, una mortal. Hércules enfrentó numerosos desafíos y aventuras a lo largo de su vida, demostrando su valentía y habilidades excepcionales.\n\nEn la mitología griega, Hércules realizó hazañas impresionantes, como matar al león de Nemea, derrotar a la hidra de Lerna, capturar el jabalí de Erimanto y limpiar los establos de Augías, entre otros trabajos. Estas tareas fueron impuestas como castigo por Euristeo, rey de Micenas y primo de Hércules, debido a una locura temporal que lo llevó a cometer un acto terrible.\n\nHércules es un personaje icónico en la mitología griega, representando la fuerza, la valentía y la lucha

contra la adversidad. Su historia ha sido contada y reinterpretada en diversas obras literarias, artísticas y culturales a lo largo de la historia.\n\nReferencia al Documento: Mitología Griega - Documento principal.

Enlaces a modelos y librerías.

Modelos:

Embeddings

GPT-3.5-TURBO

Zephyr

Librerías:

openai:

openai GitHub repository

pandas:

pandas GitHub repository

langchain:

langchain on GitHub

chromadb:

chromadb on GitHub

PyPDF2:

PyPDF2 GitHub repository

requests:

requests GitHub repository

spacy:

spacy GitHub repository

networkx:

networkx GitHub repository

matplotlib:

matplotlib GitHub repository

tqdm:

tqdm GitHub repository

typing (Standard Library):

typing module documentation

glob (Standard Library):

glob module documentation

os (Standard Library):

os module documentation

jinja2:

jinja2 GitHub repository

re (Standard Library):

re module documentation

tiktoken:

tiktoken GitHub repository

scipy:

scipy GitHub repository

numpy:

numpy GitHub repository

Ejercicio 2.

Rerank es por ejemplo cuando haces una búsqueda en Google y se obtiene una lista de páginas que podrían responder a la pregunta. Pero la primera lista que ves no siempre está en el orden que mejor responde tu pregunta. Rerank es tomar esa lista inicial y reorganizarla para que las páginas más útiles salgan primero.

Cómo Funciona el Rerank:

Primera Búsqueda: Imaginemos que estás buscando recetas de pasta en internet. Haces una búsqueda y tenes un montón de recetas. La primera lista que ves puede tener recetas que no son las mejores o más relevantes.

Primera Clasificación: Los resultados iniciales están ahí, pero no todos están ordenados como ideal. Puede que las recetas más útiles estén al final y las menos interesantes al principio.

Rerank: Acá es donde entra el rerank. Se toma la lista de recetas que encontraste y la volves a ordenar de manera más inteligente. Podes usar un sistema que sepa cuál receta es más popular, más recomendada, o que se ajuste mejor a lo que estás buscando.

Resultado Final: Al final, se obtiene una lista de recetas ordenada para que las más útiles y relevantes estén al principio. Así cuando miras los resultados, es más fácil encontrar la receta perfecta sin tener que revisar toda la lista.

Diagrama Simple

Copiar código [Inicio] → [Primera Búsqueda] → [Lista Inicial de Resultados] -- (Reordenar) —> [Rerank] → [Lista Final de Resultados] → [Resultados Útiles]

En el diagrama:

Primera Búsqueda: Encuentras una lista de resultados.

Lista Inicial de Resultados: Los resultados que encontraste, aunque no estén en el mejor orden.

Rerank: Reorganizas los resultados para que los más útiles estén arriba.

Lista Final de Resultados: Los resultados ajustados y ordenados de mejor a peor.
Resultados Útiles: La información que realmente te ayuda.

Aplicación del Rerank en el Código En el chatbot RAG, podría usar rerank para mejorar los resultados que se obtiene de la base de datos ChromaDB antes de que el modelo GPT genere la respuesta.