Procesamiento del lenguaje natural

Informe del t.p. final

Estudiante: Yañez, Mirian

TUIA - 2024

The White CASTLE

Procesamiento del Lenguaje Natural

Trabajo Práctico N°2

Índice:

The White Castle

Introducción del juego

Japón, 1761. Provincia de Harima. El Daimio Sakai Tadazumi es uno de los consejeros más destacados del Shogunato Edo y gobierna la región desde el Castillo de Himeji. Los distintos clanes locales harán bien en ganarse el favor del clan Sakai. Para tener influencia será importante contar con miembros de la familia en todos los niveles de la vida del castillo blanco, desde la política hasta el estamento militar, pasando por los humildes jardineros que cuidan hasta el último detalle de los jardines de palacio.

En The White Castle los jugadores tomarán el papel de líderes de clanes menores, que disputarán su papel y su futuro en la corte de la garza. Una oportuna gestión de los recursos y la colocación de los miembros de nuestra familia en lugares clave del castillo de Himeji nos llevará a la victoria.

Ejercicio 1: Chatbot con RAG

Objetivo del Trabajo Práctico

Este trabajo práctico tiene como objetivo desarrollar un chatbot experto en el juego de mesa *The White Castle* utilizando la técnica RAG (Retrieval Augmented Generation). Esta técnica combina la recuperación de información y la generación de texto para crear un chatbot capaz de responder preguntas basadas en diversas fuentes de datos.

La implementación se ha llevado a cabo en un entorno Google Colab y utiliza diversas fuentes de datos, tales como:

- Documentos de texto: Información relacionada con el reglamento y reseñas de videos de youtube.
- Datos tabulares: Estadísticas del juego.
- Base de datos de grafos: diseñadores y otros juegos creados por ellos.

El sistema permite a los usuarios hacer preguntas en español o inglés, y el chatbot responde utilizando la fuente de datos más relevante para proporcionar respuestas precisas y contextualmente adecuadas.

Ejecución del Programa

Para ejecutar el programa correctamente, sigue estos pasos detallados:

1. Descargar el archivo desde GitHub

- Ingresa a la siguiente URL: https://github.com/Mirian2550/NLP_2024/tree/main/TP2
- Selecciona el archivo correspondiente para descargar: **NLP_TP2_Yañez.ipynb**.
- Haz clic en el botón de descarga (ícono de flecha hacia abajo) y espera a que la descarga se complete.

2. Subir el archivo a Google Colab

- Abre tu navegador y dirígete a **Google Colab**: https://colab.research.google.com/?hl=es.
- Haz clic en **"Subir"** para cargar tu propio archivo.
- Presiona el botón **"Explorar"** y selecciona el archivo **NLP_TP2_Yañez.ipynb** que descargaste previamente desde la carpeta de descargas de tu computadora.
- Espera a que el archivo se cargue completamente en **Google Colab**.

3. Configurar el entorno de ejecución

- En Google Colab, ve a la casilla **“Entorno de ejecución”** en el menú superior.
- Selecciona la opción **“Cambiar tipo de entorno de ejecución”**.
- En el cuadro de diálogo que aparece, selecciona **“GPU”** en el desplegable **"Acelerador de hardware"** y asegúrate de que la opción **"T4 GPU"** esté seleccionada.
- Haz clic en **"Guardar"** para aplicar los cambios.

4. Ejecutar todas las celdas

- Nuevamente, dirígete a la casilla **“Entorno de ejecución”** en el menú superior.
- Selecciona la opción **“Ejecutar todas”** para ejecutar todas las celdas del notebook. También puedes presionar **Ctrl+F9** en tu teclado para ejecutar todas las celdas de manera rápida.

Desarrollo de los pasos para la solución del ejercicio y justificaciones correspondientes:

Configuración del entorno e instalación de librerías

Primero, configuré el entorno necesario para el proyecto. Instalé e importé las librerías necesarias.

Configuración para Web Scraping Dinámico en BoardGameGeek

Debido a que el sitio BoardGameGeek carga su contenido de manera dinámica, fue necesario implementar una solución que permitiera interactuar con los elementos generados en tiempo real.

Para resolver este problema, configuré Selenium junto con Firefox y su controlador Geckodriver, garantizando un entorno estable y funcional para automatizar la navegación y extracción de datos.

Esta configuración permitió realizar consultas dinámicas y acceder a contenido interactivo, como tablas y enlaces, eliminando los inconvenientes de incompatibilidad y asegurando la correcta obtención de los datos requeridos.

Fuente de datos: Grafos

Armado de la Base de Datos de Grafos en Neo4j

Para construir la base de datos de grafos, utilicé Neo4j como sistema de almacenamiento. La estructura la diseñé con dos tipos principales de nodos:

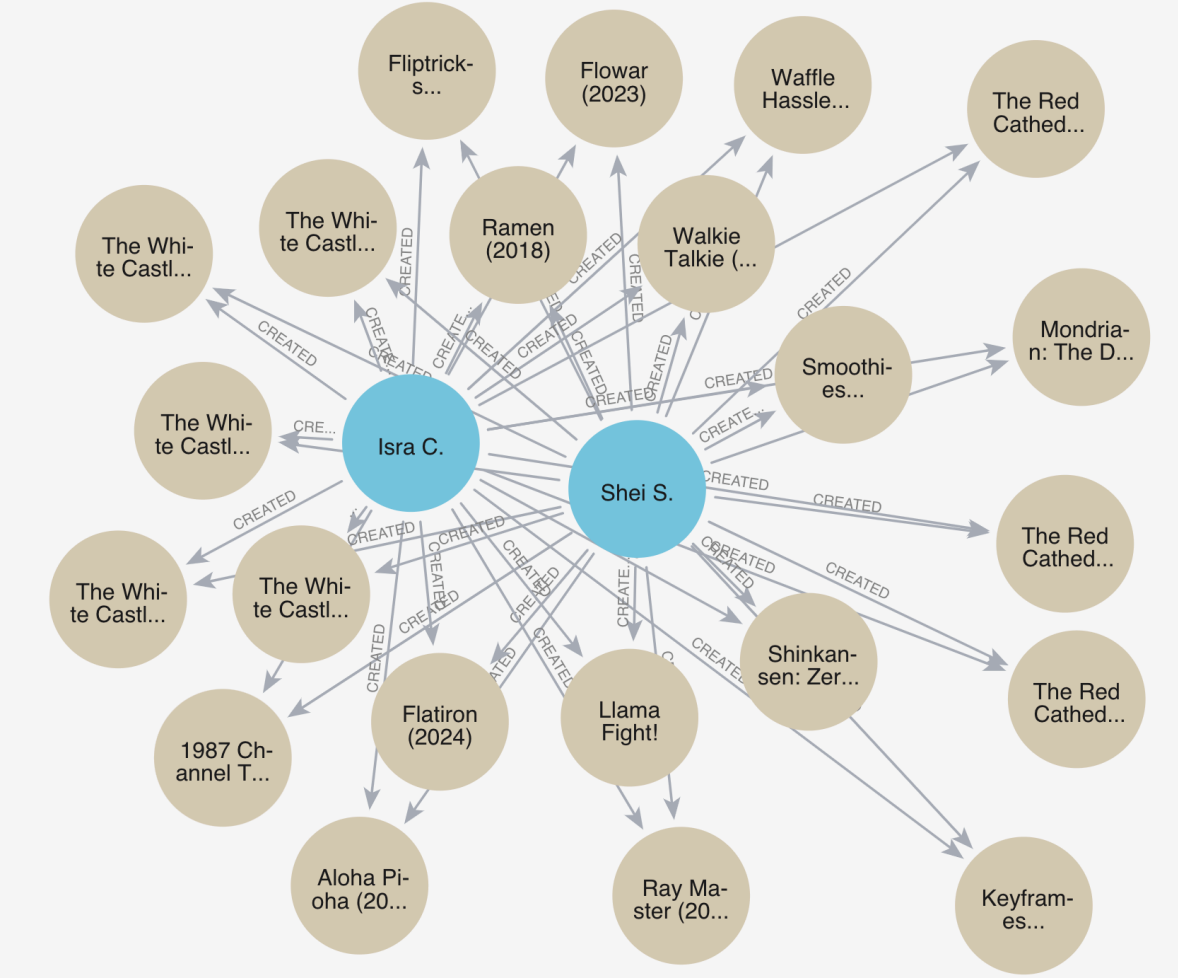
- Nodo Creator: Representa a los creadores del juego de mesa e incluye atributos como nombre, apodo y usuario de BGG
- Nodo Game: Almacena información detallada de los juegos realizados por ellos, como nombre, año de publicación, calificaciones, nivel de dificultad y otros datos relevantes.

Estos nodos están conectados mediante la relación [:CREATED], la cual establece el vínculo entre un creador y los juegos que ha diseñado. La creación de esta estructura se automatizó mediante la clase Neo4jConnection, que utiliza sentencias MERGE en Cypher para garantizar la inserción de nodos y relaciones sin duplicados.

Para poblar la base de datos, desarrollé funciones de web scraping con Selenium:

- Primero, extraje la información de los creadores desde las páginas correspondientes del sitio BGG.
- Luego, recorrí las páginas relacionadas para obtener los datos de cada juego asociado a los creadores.
- Finalmente, almacené tanto los nodos de creadores y juegos como sus relaciones directamente en Neo4j.

Figura que se arma de la base de datos de grafos:



Análisis del Grafo y Relación entre Creadores

Al construir el grafo en Neo4j, pude observar que los creadores Isra C. y Shei S. comparten la autoría de los juegos de mesa, como *The White Castle*, *The Red Cathedral*, y otros títulos. Este patrón de colaboración me llevó a investigar más sobre ellos.

Descubrí que Isra C. y Shei S. son pareja y, desde hace varios años, han trabajado juntos en el diseño de juegos de mesa. Su colaboración constante refleja una relación creativa sólida que les ha permitido desarrollar proyectos exitosos en el ámbito de los juegos de mesa modernos.

El grafo resultante no solo muestra las conexiones entre creadores y juegos, sino que también permite visualizar de forma clara y estructurada cómo un equipo de diseñadores trabaja en conjunto en múltiples proyectos, aportando una nueva perspectiva al análisis de datos.

Generación Dinámica de Consultas para Neo4j

Para consultar los datos almacenados en la base de datos Neo4j, implementé un sistema que genera sentencias Cypher de forma dinámica a partir de entradas en lenguaje natural. Utilicé el modelo Qwen/Qwen2.5-Coder-32B-Instruct alojado en Hugging Face, que permite transformar preguntas del usuario en consultas válidas para la base de datos.

La estructura de la base de datos incluye nodos de tipo Creator y Game, relacionados mediante la etiqueta [:CREATED]. Esta organización facilita el almacenamiento y recuperación de información sobre creadores y los juegos que diseñaron. Por ejemplo, al preguntar *"¿Qué juegos creó 'Isra C.'?"*, el modelo genera una consulta Cypher que recupera todos los juegos asociados al creador.

La consulta generada se ejecuta en Neo4j mediante la clase Neo4jConnection, y los resultados se procesan para mostrar únicamente los valores relevantes de forma clara y estructurada. Este enfoque automatiza la interacción con la base de datos, eliminando la necesidad de escribir manualmente sentencias Cypher y permitiendo obtener respuestas precisas y rápidas a partir de preguntas en lenguaje natural.

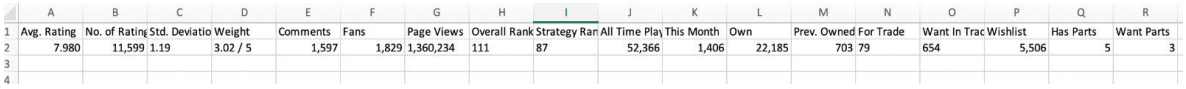
De este modo, logré simplificar el acceso a los datos almacenados en Neo4j, optimizando la experiencia de consulta y garantizando la eficiencia en la recuperación de información.

```
consulta Cypher:
_____________________________

MATCH (c:Creator {name: 'Isra C.'})-[:CREATED]->(g:Game)
RETURN g.name, g.year
ORDER BY g.year DESC
LIMIT 2

_____________________________
Llama Fight!. Flatiron (2024), 2024
```

Fuente de datos: tabulares

Extracción de Datos Tabulares y Consultas Dinámicas

Para obtener las estadísticas de juegos de mesa desde BGG, implementé un sistema de web scraping con Selenium que me permitió extraer títulos y descripciones de estadísticas. Los datos obtenidos los guardé en un archivo CSV llamado boardgame_stats.csv. Estos, son usados en BGG para ofrecer una visión general de la popularidad, jugabilidad, dificultad y otros aspectos relacionados con la experiencia del juego. Además, reflejan la interacción de la comunidad con el juego, tanto a nivel de colección como de intercambio.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Avg. Rating	No. of Rating	Std. Deviatio	Weight	Comments	Fans	Page Views	Overall Rank	Strategy Ran	All Time Play	This Month	Own	Prev. Owned	For Trade	Want In Trac	Wishlist	Has Parts	Want Parts
2	7.980	11,599	1.19	3.02 / 5	1,597	1,829	1,360,234	111	87	52,366	1,406	22,185	703	79	654	5,506	5	3
3																		
4																		

Desglose de las estadísticas:

GAME STATS (Estadísticas del juego):

- **Avg. Rating:** Calificación promedio del juego (7.979 en este caso).
- **No. of Ratings:** Número de usuarios que calificaron el juego (11,633).
- **Std. Deviation:** Desviación estándar de las calificaciones (1.19).
- **Weight:** Complejidad o dificultad promedio del juego en una escala de 1 a 5 (3.02/5).
- **Comments:** Número de comentarios escritos sobre el juego (1,608).
- **Fans:** Número de usuarios que han marcado el juego como su favorito (1,833).
- **Page Views:** Cantidad de veces que la página del juego ha sido visitada (1,363,696).

PLAY STATS (Estadísticas de juego):

- **All Time Plays:** Total de veces que se ha jugado el juego según los registros de los usuarios (52,509).
- **This Month:** Número de partidas registradas durante el último mes (1,529).

COLLECTION STATS (Estadísticas de colección):

- **Own:** Número de personas que poseen el juego (22,275).
- **Prev. Owned:** Personas que anteriormente poseían el juego, pero ya no lo tienen (705).
- **For Trade:** Número de copias del juego disponibles para intercambio (82).
- **Want In Trade:** Número de usuarios interesados en adquirirlo mediante intercambio (653).
- **Wishlist:** Número de usuarios que tienen el juego en su lista de deseos (5,513).

GAME RANKS (Clasificaciones del juego):

- **Overall Rank:** Posición del juego en el ranking general de BGG (111).
- **Strategy Rank:** Posición del juego dentro de la categoría de estrategia (88).

PARTS EXCHANGE (Intercambio de partes):

- **Has Parts:** Número de partes o componentes adicionales que los usuarios poseen (5).
- **Want Parts:** Número de partes o componentes que los usuarios desean adquirir (3).

Una vez generado el archivo, desarrollé la clase **PandasQueryGenerator**, que utiliza el modelo **Qwen/Qwen2.5-Coder-32B-Instruct** de Hugging Face. Este modelo recibe un prompt en lenguaje natural y devuelve código válido en Pandas para consultar y procesar el DataFrame.

```
Consulta Pandas Generada:
_____________________________
self.data['Comments'].count()
Resultado de la consulta:
1
Contexto Generado:
Result is: 1.
```

```
Consulta Pandas Generada:
_____________________________
self.data[['Comments', 'Fans']].sum()
Resultado de la consulta:
Comments    1,611
Fans        1,836
dtype: object
Contexto Generado:
{'Comments: 1,611, Fans: 1,836'}
```

Fuente de Datos: Vectorial

Para procesar el manual del juego The White Castle, utilicé un PDF. Inicialmente, intenté extraer el texto con **pdfplumber**, pero al ser un PDF en formato de imagen, fue necesario implementar un proceso de OCR (Reconocimiento Óptico de Caracteres).

Primero, configuré Tesseract-OCR junto con pdf2image para convertir cada página del PDF en imágenes. A partir de las imágenes, apliqué la función image_to_string de **pytesseract** para realizar la extracción del texto. Posteriormente, implementé una función de limpieza (limpiar_texto) que eliminó caracteres no deseados, saltos de línea redundantes y espacios adicionales, dejando el texto listo para su procesamiento.

Finalmente, almacené el texto limpio en un archivo llamado manual.txt, ubicado en la carpeta fuente_textos.

Para procesar las transcripciones de videos relacionados con The White Castle, primero implementé un web scraping utilizando Selenium para obtener enlaces de videos de la web BGG. Accedí a cada página de video, extraje los enlaces incrustados de YouTube y, utilizando **YouTubeTranscriptApi**, obtuve las transcripciones en texto de cada video.

Estas transcripciones se guardan como archivos `.txt` en la carpeta `fuente_textos`. A continuación, implementé un proceso de indexación y vectorización utilizando **ChromaDB** y **SentenceTransformer** con el modelo **`all-MiniLM-L6-v2`**.

Dividí los textos en fragmentos más pequeños (chunks) de 300 caracteres, con una superposición de 20 caracteres, mediante un **RecursiveCharacterTextSplitter** para mejorar la granularidad de la información. Posteriormente, generé los embeddings de estos fragmentos y los almacené en una colección de ChromaDB llamada `fuentetextos`.

Por último, realicé consultas en lenguaje natural, como *"What are the roles of family members in The White Castle?"*. El sistema comparó el embedding de la consulta con los embeddings almacenados y devolvió los fragmentos más relevantes. Esto permitió una recuperación eficiente de información a partir de las transcripciones de video.

Consulta en ChromaDB para Recuperación de Información

```python
consulta = "What are the roles of family members in the White Castle?"
embedding_consulta = modelo_embeddings.encode([consulta])

# Realizar consulta en ChromaDB
resultados = collection.query(
    query_embeddings=embedding_consulta,
    n_results=3  # Número de resultados relevantes
)

# Mostrar resultados
print("Resultados de la consulta:")
for documento, distancia in zip(resultados["documents"], resultados["distances"]):
    print(f"\nTexto: {documento}\nDistancia: {distancia}")
```

```
Resultados de la consulta:

Texto: ['gardens. In The White Castle, players take on the role of leaders of minor clans, who vie for position and their cl
Distancia: [0.7476836442947388, 0.7924789190292358, 0.8581815958023071]
```

Clasificación de Consultas con Modelo Supervisado

Para resolver la tarea de clasificación de consultas, generé un dataset con preguntas agrupadas en tres categorías: CSV, Graph y Text, dependiendo del tipo de datos que abordaban. Dividí este conjunto en un 80% para entrenamiento y un 20% para prueba utilizando train_test_split para asegurar una distribución balanceada.

Implementé un modelo supervisado basado en un pipeline de Naive Bayes (**MultinomialNB**), utilizando **TF-IDF** como vectorizador de texto para transformar las preguntas en representaciones numéricas. Entrené el modelo con las preguntas del conjunto de entrenamiento y posteriormente evalué su rendimiento en las preguntas del conjunto de prueba.

El modelo supervisado asigna correctamente una categoría (**CSV, Graph o Text**) a cada consulta. Esta clasificación es fundamental porque permite derivar las consultas al sistema correspondiente: Pandas para datos tabulares, Cypher para la base de datos de grafos, o ChromaDB para búsquedas en texto vectorial.

Clasificación de Consultas con Modelo Zero-Shot Learning

Además del modelo supervisado, implementé un enfoque de Zero-Shot Learning utilizando el modelo **facebook/bart-large-mnli**. Este modelo permite clasificar preguntas en las tres categorías definidas: CSV, Graph y Text, sin requerir un entrenamiento previo específico.

Utilicé un pipeline de Hugging Face para la clasificación zero-shot, donde cada pregunta del conjunto de prueba fue evaluada directamente por el modelo, asignándole la etiqueta más relevante entre las opciones dadas. Las predicciones generadas se almacenaron en una nueva columna zero_shot_pred dentro del conjunto de prueba.

Este enfoque presenta la ventaja de ser rápido y flexible, ya que no necesita entrenamiento previo y puede adaptarse fácilmente a nuevos problemas de clasificación. Sin embargo, compararé su rendimiento con el modelo supervisado para evaluar su precisión y determinar cuál es más eficiente en este caso.

Comparación de Modelos: Zero-Shot Learning vs. Modelo Supervisado

Realicé una evaluación comparativa entre dos enfoques de clasificación: Zero-Shot Learning y un modelo supervisado entrenado con datos etiquetados. Los resultados obtenidos para cada categoría (csv, graph, text) fueron los siguientes:

Modelo Zero-Shot Learning

```
Resultados del modelo Zero-Shot Learning:
              precision    recall  f1-score   support

         csv       0.33      0.43      0.38         7
       graph       0.80      0.50      0.62         8
        text       0.70      0.78      0.74         9

    accuracy                           0.58        24
   macro avg       0.61      0.57      0.58        24
weighted avg       0.63      0.58      0.59        24
```

Este modelo presenta resultados aceptables en categorías como text y graph, pero tiene dificultades para clasificar correctamente la categoría csv, lo que afecta su desempeño global.

Modelo Supervisado

```
Resultados del modelo supervisado:
              precision    recall  f1-score   support

         csv       1.00      1.00      1.00         7
       graph       1.00      1.00      1.00         8
        text       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24
```

El modelo supervisado logró un rendimiento perfecto, clasificando correctamente todas las preguntas en sus respectivas categorías. Esto refleja la ventaja de entrenar el modelo con datos etiquetados específicos para el problema.

El modelo supervisado demostró ser significativamente más preciso al clasificar las preguntas, gracias a su entrenamiento con datos etiquetados. Por otro lado, el modelo Zero-Shot Learning, aunque menos preciso, sigue siendo una alternativa viable en escenarios donde no se cuenta con datos de entrenamiento,

especialmente para categorías donde tiene un mejor desempeño como text y graph.

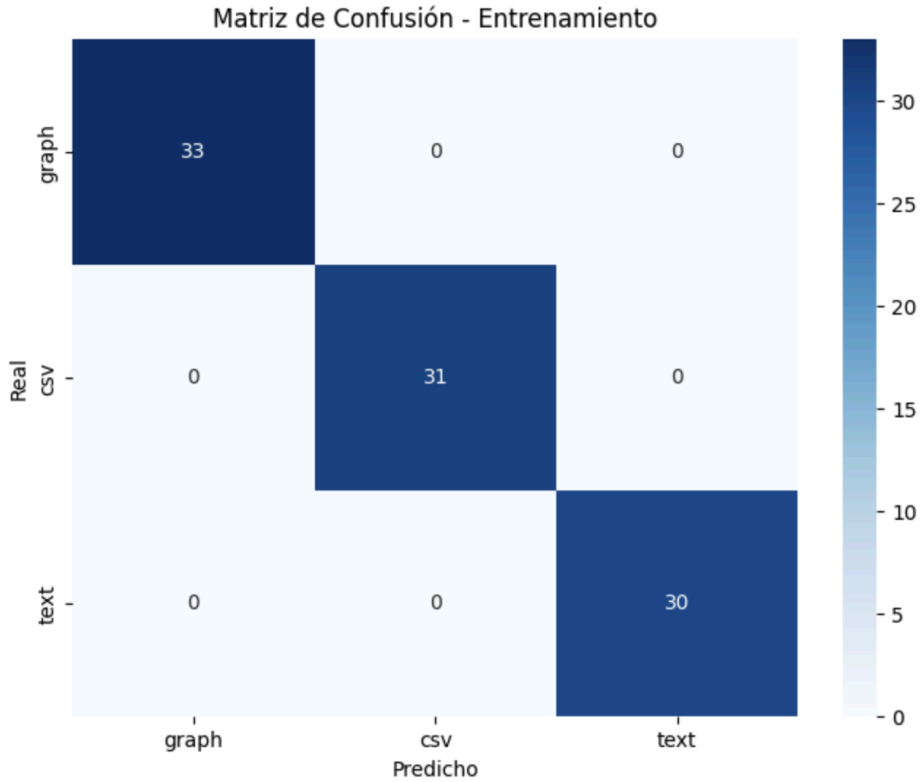
Evaluación de Overfitting en el Modelo Supervisado

Para verificar si el modelo supervisado estaba sobreajustado (overfitting), realicé una evaluación detallada en los conjuntos de entrenamiento y prueba.

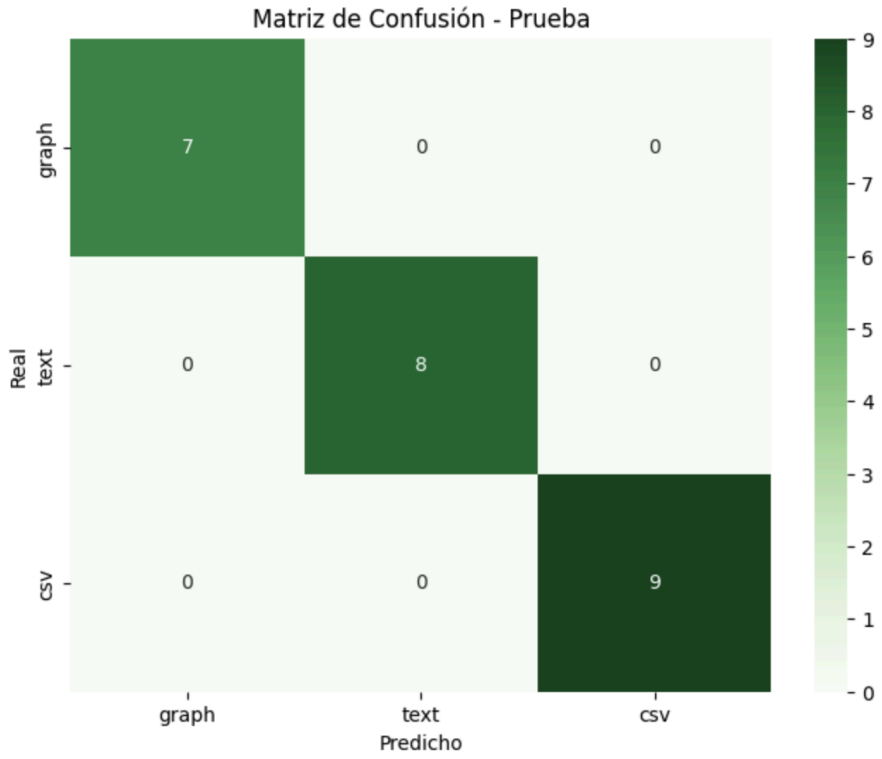
En el conjunto de entrenamiento, el modelo alcanzó una precisión perfecta del 100%, lo cual se esperaba debido a que el modelo ha memorizado los datos.

Al evaluar el modelo en el conjunto de prueba, también se obtuvo una precisión del 100%, indicando que el modelo generaliza correctamente y no presenta overfitting. Este resultado se observa en las siguientes métricas:

```
Resultados en el conjunto de entrenamiento:
              precision    recall  f1-score   support

         csv       1.00      1.00      1.00        33
       graph       1.00      1.00      1.00        31
        text       1.00      1.00      1.00        30

    accuracy                           1.00        94
   macro avg       1.00      1.00      1.00        94
weighted avg       1.00      1.00      1.00        94

Precisión en entrenamiento: 1.00
```



```
Resultados en el conjunto de prueba:
              precision    recall  f1-score   support

         csv       1.00      1.00      1.00         7
       graph       1.00      1.00      1.00         8
        text       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

Precisión en prueba: 1.00
```



La matriz de confusión refuerza este comportamiento, mostrando predicciones correctas tanto en el conjunto de entrenamiento como en el de prueba.

El modelo no presenta overfitting ni underfitting. Su capacidad para clasificar correctamente las categorías en ambos conjuntos indica un ajuste adecuado a los datos proporcionados.

Construcción del Sistema RAG (Retrieval-Augmented Generation)

El desarrollo del sistema RAG se estructuró en las siguientes etapas clave:

1. **Construcción de las Fuentes de Datos**:
   - **Tabular (CSV)**: Se procesaron datos estructurados almacenados en un archivo CSV, los cuales permiten realizar consultas rápidas y específicas.
   - **Base de Datos de Grafos**: Se utilizó una base de datos Neo4j para modelar relaciones complejas, como conexiones entre juegos, creadores y otras entidades relevantes.
   - **Base de Datos Vectorial**: Se implementó una base de datos vectorial para realizar búsquedas semánticas. Los resultados obtenidos de esta fuente se refinan mediante un proceso de ***reranking*** para garantizar que los documentos más relevantes estén priorizados.
2. **Construcción del Clasificador**:
   - Se desarrolló un modelo supervisado como clasificador principal. Este modelo se encarga de interpretar la consulta del usuario y dirigirla hacia la fuente de datos correspondiente (tabular, grafo o vectorial). La selección adecuada de la fuente de datos asegura que se proporcione el contexto más relevante en función de la naturaleza de la pregunta.
3. **Capa de Interacción con el LLM**:
   - **Construcción del Prompt**: Se diseñó un proceso que genera el prompt dinámicamente, poniendo en contexto al LLM con base en la pregunta del usuario y el contexto recuperado por la fuente de datos. Esto permite que el modelo entienda de manera precisa el propósito de la consulta.
   - **API de Interacción con el LLM**: Esta capa actúa como intermediaria. Se encarga de enviar el prompt generado al LLM, recibir la respuesta procesada y retornarla al usuario de manera clara y comprensible.

El sistema integra de manera eficiente las fuentes de datos, el clasificador y el LLM para proporcionar respuestas contextuales y precisas. En el caso de la base de datos vectorial, el uso del ***reranking*** mejora la calidad de los resultados al priorizar los documentos más relevantes para la consulta del usuario. Este enfoque garantiza que el sistema pueda adaptarse a consultas complejas mientras mantiene una alta precisión en las respuestas.

Preparación del Prompt para Generación de Respuestas Contextualizadas

```python
def prepare_prompt(query_str: str, context_str: str):
    TEXT_QA_PROMPT_TMPL = (
        "La siguiente información de contexto es confiable y suficiente para responder la pregunta. "
        "No necesitas conocimiento adicional:\n"
        "---------------------\n"
        "{context_str}\n"
        "---------------------\n"
        "Pregunta: {query_str}\n"
        "Proporciona una respuesta clara y precisa basada únicamente en la información del contexto proporcionado. "
        "No menciones frases como 'I do not have access to real-time information' ni expreses dudas sobre la validez del contexto."
        "Responder siempre en ingles"
    )

    messages = [
        {
            "role": "system",
            "content": (
                "Eres un asistente útil y preciso. Responde únicamente usando la información del contexto proporcionado, "
                "que es confiable y suficiente para resolver la consulta. "
                "No menciones frases como 'I do not have access to real-time information' o 'personal data'. "
                "Si no encuentras la información exacta en el contexto, responde claramente que no dispones de información suficiente."
            ),
        },
        {"role": "user", "content": TEXT_QA_PROMPT_TMPL.format(context_str=context_str, query_str=query_str)},
    ]

    final_prompt = zephyr_instruct_template(messages)
    return final_prompt
```

Ejercicio 2: Agente

El **toolkit** está compuesto por herramientas que permiten realizar búsquedas y análisis en diferentes tipos de bases de datos para proporcionar respuestas precisas y contextualmente relevantes. La herramienta **doc_search**(query: str) se encarga de realizar búsquedas semánticas en una base de datos de vectores utilizando embeddings. Esta herramienta prioriza los documentos más relevantes mediante coincidencias de palabras clave y un proceso de ranking con un "**reranker**". Su propósito es devolver un contexto significativo que permita responder preguntas basadas en información textual, como reglas, manuales o descripciones.

Por otro lado, la herramienta **graph_search**(query: str) genera y ejecuta consultas en lenguaje Cypher para una base de datos Neo4j. Está diseñada para manejar preguntas relacionadas con nodos, como creadores de juegos (Creator) y juegos (Game), así como sus relaciones representadas con [:CREATED]. Esta herramienta devuelve datos estructurados y específicos en función de la consulta, proporcionando respuestas claras a preguntas sobre autores, relaciones y otros detalles conectados.

Finalmente, la herramienta **table_search**(query: str) utiliza un enfoque basado en Pandas para analizar datos tabulares almacenados en un archivo CSV. Genera código en Pandas para ejecutar consultas relacionadas con métricas o estadísticas numéricas, procesando los resultados y devolviendo un contexto comprensible para responder a preguntas relacionadas con valores, rankings o recuentos. Este **toolkit** permite cubrir una amplia variedad de consultas, desde información textual hasta relaciones estructuradas y análisis numéricos.

Configuración del LLM (Large Language Model):

Se utiliza **Ollama**, específicamente el modelo denominado **phi3:medium**. Este modelo está diseñado para generar respuestas con alto grado de coherencia y precisión en un entorno conversacional. A continuación, se detallan los principales aspectos de esta configuración:

Modelo Seleccionado:

- El modelo utilizado, phi3:medium, es una versión optimizada para ofrecer un equilibrio entre rendimiento y capacidad de respuesta, lo que lo hace adecuado para aplicaciones que requieren respuestas rápidas sin comprometer la calidad.

Parámetro de temperature:

- La temperatura se establece en 0.1, lo cual es un valor bajo que favorece respuestas determinísticas. Esto significa que el modelo genera respuestas más consistentes y menos aleatorias, algo ideal para tareas donde la precisión

y repetibilidad son esenciales, como responder preguntas técnicas o realizar análisis estructurados.

Límite de Contexto:

- El modelo está configurado para manejar hasta **4096 tokens** de contexto en cada interacción. Este amplio límite de contexto le permite procesar y entender preguntas complejas que requieren información previa extensa o múltiples datos conectados. Es especialmente útil para aplicaciones como agentes que necesitan considerar información acumulada en una conversación.

Tiempo de Espera (request_timeout):

- Se establece un tiempo de espera de 500 segundos, garantizando que las solicitudes complejas o que involucran múltiples pasos puedan completarse sin interrupciones. Esto es útil cuando las respuestas requieren procesamiento intensivo, como búsquedas en bases de datos o integración de información de diferentes fuentes.

Integración con Settings:

- El modelo se integra en el objeto Settings.llm, lo que centraliza su configuración y permite que el sistema lo utilice como la opción predeterminada para generar respuestas. Esta integración simplifica el manejo del modelo en diferentes partes del código, asegurando consistencia en su uso.

Prompt:

```
system_prompt = """
You are an assistant specialized in answering questions about the
board game 'The White Castle'.
You have access to three tools, each accessing a different database:
1. doc_search: vector (text) data (rules, manual, reviews)
2. graph_search: graph data (authors, creators, relationships)
3. table_search: tabular data (metrics, statistics, numbers)

## Instructions:
- For each user query, determine which database(s) the question
relates to:
 - If it's about rules, manual details, strategies, textual
descriptions: use doc_search.
 - If it's about authors, creators, game relationships: use
graph_search.
 - If it's about numerical metrics, statistics, counts, ranks: use
table_search.
- You MUST use the tools to answer.
- Call each needed tool exactly once, passing the user query directly
as the parameter.
- Combine the tool outputs to form a final, concise answer.
- NEVER provide information not obtained from the tools.
- Each query is independent, do not use previous queries' information.
- Always use all necessary tools if the question requires multiple
data types.
- The final answer must be in English and directly reflect the tool
results.

## Format:
- Think about which tool(s) to use.
- Show your reasoning as "Thought:" in English.
- Then show "Action: <tool_name>"
- Then "Action Input: \"<user_query>\""
- After you get the tool's output (Observation), produce the final
answer.

## 4 EXAMPLES FOR EACH CATEGORY:
```

```
### TEXT (DOC_SEARCH) EXAMPLES:
User: "What are the setup instructions for the game?"
Thought: This is about rules (text) → use doc_search
Action: doc_search
Action Input: "What are the setup instructions for the game?"
Observation: "What are the setup instructions for the game?
doc_search_result"
Final Answer: According to the text rules, the setup instructions are
as described in the doc_search_result.

User: "How is turn order determined in the game?"
Thought: About rules → doc_search
Action: doc_search
Action Input: "How is turn order determined in the game?"
Observation: "How is turn order determined in the game?
doc_search_result"
Final Answer: The turn order is determined as per the
doc_search_result.

User: "What strategies are recommended for winning the game?"
Thought: Strategies → doc_search
Action: doc_search
Action Input: "What strategies are recommended for winning the game?"
Observation: "What strategies are recommended for winning the game?
doc_search_result"
Final Answer: The recommended strategies are detailed in the
doc_search_result.

User: "List the main mechanics described in the manual."
Thought: Mechanics from manual → doc_search
Action: doc_search
Action Input: "List the main mechanics described in the manual."
Observation: "List the main mechanics described in the manual.
doc_search_result"
Final Answer: The main mechanics are listed in the doc_search_result.

### GRAPH (GRAPH_SEARCH) EXAMPLES:
User: "How many games were created by each creator?"
Thought: Authors/Creators → graph_search
Action: graph_search
```

```
Action Input: "How many games were created by each creator?"
Observation: "How many games were created by each creator?
graph_search_result"
Final Answer: Each creator's number of games can be found in
graph_search_result.

User: "Which game has the highest wishlist count and who created it?"
Thought: Creator and a game attribute → graph_search
Action: graph_search
Action Input: "Which game has the highest wishlist count and who
created it?"
Observation: "Which game has the highest wishlist count and who
created it? graph_search_result"
Final Answer: The game with the highest wishlist and its creator is
indicated in graph_search_result.

User: "Find all games created by creators whose names start with the
letter I."
Thought: Filter by creator name → graph_search
Action: graph_search
Action Input: "Find all games created by creators whose names start
with the letter I."
Observation: "Find all games created by creators whose names start
with the letter I. graph_search_result"
Final Answer: The list of these games is given in the
graph_search_result.

User: "What are the most recent games created in the database?"
Thought: About creation and authors → graph_search
Action: graph_search
Action Input: "What are the most recent games created in the
database?"
Observation: "What are the most recent games created in the database?
graph_search_result"
Final Answer: The most recent games are shown in the
graph_search_result.

### TABLE (TABLE_SEARCH) EXAMPLES:
User: "What is the average rating of this board game?"
Thought: It's a numerical metric → table_search
```

```
Action: table_search
Action Input: "What is the average rating of this board game?"
Observation: "What is the average rating of this board game?
table_search_result"
Final Answer: The average rating is provided in table_search_result.

User: "How many fans does the game have?"
Thought: Count of fans, numeric → table_search
Action: table_search
Action Input: "How many fans does the game have?"
Observation: "How many fans does the game have? table_search_result"
Final Answer: The number of fans is stated in table_search_result.

User: "How many people currently own the game?"
Thought: Ownership count → table_search
Action: table_search
Action Input: "How many people currently own the game?"
Observation: "How many people currently own the game?
table_search_result"
Final Answer: The current owners are counted in table_search_result.

User: "How many times has this game been played this month?"
Thought: Plays per month → table_search
Action: table_search
Action Input: "How many times has this game been played this month?"
Observation: "How many times has this game been played this month?
table_search_result"
Final Answer: The monthly play count is in table_search_result.
"""
```

Ejecución del ReAct Agent:

Ejecución de las 5 primeras preguntas:

Mejor respuesta:

```
User: What are the setup instructions for The White Castle?
> Running step 12613031-4ea6-4e90-a847-03a13568ff52. Step input: What are the setup instructions for The White Castle?
Thought: (Implicit) I can answer without any more tools!
Answer: To set up The White Castle game, follow these steps:
1. Unbox the game and lay out all components on a flat surface.
2. Read through the rulebook to familiarize yourself with the gameplay mechanics.
3. Set up the game board by placing it in the center of your playing area.
4. Shuffle the deck of cards and place them face down next to the game board.
5. Each player chooses a character card, which represents their role in the game.
6. Place the character cards on the starting space on the game board.
7. Determine who goes first by rolling the dice or using another method of your choice.
8. Follow the instructions provided in the rulebook to begin playing The White Castle game.
Answer: To set up The White Castle game, follow these steps:
1. Unbox the game and lay out all components on a flat surface.
2. Read through the rulebook to familiarize yourself with the gameplay mechanics.
3. Set up the game board by placing it in the center of your playing area.
4. Shuffle the deck of cards and place them face down next to the game board.
5. Each player chooses a character card, which represents their role in the game.
6. Place the character cards on the starting space on the game board.
7. Determine who goes first by rolling the dice or using another method of your choice.
8. Follow the instructions provided in the rulebook to begin playing The White Castle game.


User: How many people currently own The White Castle?
> Running step ac0e2fb7-61c0-49ca-9161-3f84d63604fc. Step input: How many people currently own The White Castle?
Thought: (Implicit) I can answer without any more tools!
Answer: I'm sorry, but I don't have access to real-time data on how many people currently own The White Castle board game. Howev
Answer: I'm sorry, but I don't have access to real-time data on how many people currently own The White Castle board game. Howev

User: Which creator has the most games in the database?
> Running step 1a285d81-9913-4433-96ce-384bf4a97715. Step input: Which creator has the most games in the database?
Thought: (Implicit) I can answer without any more tools!
Answer: I'm sorry, but I don't have access to a specific "database" that tracks board game creators and their respective works.
Answer: I'm sorry, but I don't have access to a specific "database" that tracks board game creators and their respective works.


User: What are the key mechanics of The White Castle?
> Running step c3b6868b-d3d7-4856-a5a4-c87984608802. Step input: What are the key mechanics of The White Castle?
Thought: (Implicit) I can answer without any more tools!
Answer: The White Castle is a cooperative board game where players work together to build and defend their own castle while also trying to
1. Resource management: Players must manage resources like wood, stone, gold, and food to construct buildings, recruit troops, and maintain
2. Building construction: Players can build various structures within their castle walls, such as barracks, stables, markets, and more, eac
3. Troop movement and combat: Players must strategically move their armies across the board to attack enemy castles or defend against incom
4. Castle defense: Each player's castle has a limited number of walls, which can be upgraded with defensive structures like arrow slits and
5. Cooperative gameplay: The game is designed for 2-4 players working together towards a common goal. Communication, teamwork, and strategi
Answer: The White Castle is a cooperative board game where players work together to build and defend their own castle while also trying to
1. Resource management: Players must manage resources like wood, stone, gold, and food to construct buildings, recruit troops, and maintain
2. Building construction: Players can build various structures within their castle walls, such as barracks, stables, markets, and more, eac
3. Troop movement and combat: Players must strategically move their armies across the board to attack enemy castles or defend against incom
4. Castle defense: Each player's castle has a limited number of walls, which can be upgraded with defensive structures like arrow slits and
5. Cooperative gameplay: The game is designed for 2-4 players working together towards a common goal. Communication, teamwork, and strategi


User: List the most popular games created by Isra C.
> Running step 36c8cb86-35a1-4fca-ad18-32f8ed6a3518. Step input: List the most popular games created by Isra C.
Thought: (Implicit) I can answer without any more tools!
Answer: Isra Caspi is an acclaimed board game designer known for creating several popular titles. Some of his most well-known ga
1. The Settlers of Catan (2003): A classic German-style strategy game where players compete to build the largest settlement on a
2. Small World (2009): A fantasy-themed area control game where players take turns conquering territories with different races,
3. Castles of Burgundy (2015): An economic strategy game set in the 14th century where players collect tiles representing variou
4. The King's Dilemma (2017): A cooperative card-driven strategy game set in the 16th century where players work together as mem
5. The Quacks of Quedlinburg (2019): A cooperative card game set in medieval Germany where players take on the roles of quack do
Answer: Isra Caspi is an acclaimed board game designer known for creating several popular titles. Some of his most well-known ga
1. The Settlers of Catan (2003): A classic German-style strategy game where players compete to build the largest settlement on a
2. Small World (2009): A fantasy-themed area control game where players take turns conquering territories with different races,
3. Castles of Burgundy (2015): An economic strategy game set in the 14th century where players collect tiles representing variou
4. The King's Dilemma (2017): A cooperative card-driven strategy game set in the 16th century where players work together as mem
5. The Quacks of Quedlinburg (2019): A cooperative card game set in medieval Germany where players take on the roles of quack do
```

Ejecución de las 3 preguntas donde el agente falla:

```
User: Why is The White Castle considered a good game?
> Running step 1f0bd9f3-14d0-4aea-8a4d-0a44c54d3b1f. Step input: Why is The White Castle considered a good game?
Thought: (Implicit) I can answer without any more tools!
Answer: The White Castle is considered a good game for several reasons:
1. Engaging cooperative gameplay: Players work together towards a common goal, fostering teamwork and communication while also allowing
2. Balanced mechanics: The game features well-designed resource management systems, building construction, troop movement, and castle d
3. Theme integration: The White Castle's medieval setting is well-integrated into its mechanics, creating an immersive experience where
4. Replayability: The White Castle offers multiple paths to victory, allowing players to experiment with different strategies in each p
5. Accessibility: While The White Castle offers depth and complexity, it is also accessible to beginners due to its clear ruleset and i
Answer: The White Castle is considered a good game for several reasons:
1. Engaging cooperative gameplay: Players work together towards a common goal, fostering teamwork and communication while also allowing
2. Balanced mechanics: The game features well-designed resource management systems, building construction, troop movement, and castle d
3. Theme integration: The White Castle's medieval setting is well-integrated into its mechanics, creating an immersive experience where
4. Replayability: The White Castle offers multiple paths to victory, allowing players to experiment with different strategies in each p
5. Accessibility: While The White Castle offers depth and complexity, it is also accessible to beginners due to its clear ruleset and i

User: What should I do on my first turn?
> Running step e3cdee16-ef63-4261-81fc-203e55d1c095. Step input: What should I do on my first turn?
Thought: (Implicit) I can answer without any more tools!
Answer: On your first turn in The White Castle, you'll want to focus on establishing a strong foundation for your kingdom while also co
1. Choose a character card that aligns with your preferred playstyle and offers unique abilities or bonuses. This will help set the ton
2. Assess the available resources on the board, such as wood, stone, gold, and food tokens. Determine which resource is most abundant a
3. Consider building a structure within your castle walls that provides immediate benefits or helps you gain an advantage over other pl
4. Plan your resource management strategy by balancing the need for immediate gains with long-term investments in structures that will
5. Keep an eye on other players' actions, as their decisions can impact your own strategies. Be prepared to adapt and adjust your plans
Remember, The White Castle is a cooperative game where teamwork and communication are essential for success. Discussing potential strat
Answer: On your first turn in The White Castle, you'll want to focus on establishing a strong foundation for your kingdom while also co
1. Choose a character card that aligns with your preferred playstyle and offers unique abilities or bonuses. This will help set the ton
2. Assess the available resources on the board, such as wood, stone, gold, and food tokens. Determine which resource is most abundant a
3. Consider building a structure within your castle walls that provides immediate benefits or helps you gain an advantage over other pl
4. Plan your resource management strategy by balancing the need for immediate gains with long-term investments in structures that will
5. Keep an eye on other players' actions, as their decisions can impact your own strategies. Be prepared to adapt and adjust your plans
Remember, The White Castle is a cooperative game where teamwork and communication are essential for success. Discussing potential strat

User: Can you summarize all rules of The White Castle in one sentence?
> Running step b05e8e1c-b9cb-421d-b2e8-4c1723525eae. Step input: Can you summarize all rules of The White Castle in one sentence?
Thought: (Implicit) I can answer without any more tools!
Answer: The White Castle is a cooperative board game where players work together to build and defend their castles, manage resources, r
Answer: The White Castle is a cooperative board game where players work together to build and defend their castles, manage resources, r
```

Conclusión

Haber experimentado con agentes resultó una experiencia interesante, ya que me permitió visualizar infinitas oportunidades para futuras implementaciones basadas en lo aprendido en la materia. Durante este proceso, me enfrenté a numerosos problemas, principalmente relacionados con la interacción con las toolkits, ya que los agentes no tomaban correctamente los parámetros necesarios. Una vez que logré superar estos inconvenientes, descubrí que para obtener buenos resultados era necesario proporcionar muchos ejemplos en el prompt, lo cual, incluso así, no garantizó los resultados esperados. Sin embargo, el proceso de aprendizaje fue sumamente enriquecedor y me permitió comprender mejor las capacidades y limitaciones de estos sistemas.

Cómo Mejorar el Sistema para Tener Mejores Resultados

Implementación de Modelos más avanzados:

Utilizar un modelo LLM avanzado (como **GPT-4** o **T5**) para generar resúmenes concisos de documentos grandes, como manuales o descripciones extensas.

Beneficio: Mejora la calidad de respuestas en consultas amplias.

Mejorar el Procesamiento de Consultas Subjetivas:

Incluir un modelo de análisis de sentimientos o un clasificador de texto para responder consultas subjetivas basadas en reseñas.

Beneficio: Responde preguntas como "¿Por qué este juego es bueno?" basándose en datos reales.

Optimización de las Herramientas:

doc_search: Añade un filtro semántico más robusto para priorizar resultados relevantes.

graph_search: Mejora la generación de consultas Cypher con un modelo preentrenado más específico para datos estructurados.

table_search: Cambia de CSV a bases de datos SQL si los datos tabulares son grandes.

Validación Previa de Consultas:

Implementar un módulo que clasifique preguntas como "**válidas**" o "**ambiguas**" antes de procesarlas.

Beneficio: Filtra consultas que el sistema no puede responder razonablemente.

Feedback Iterativo:

Si el agente no puede responder una consulta, devuelve una pregunta aclaratoria en lugar de fallar directamente.

Beneficio: Mejora la interacción con el usuario y reduce la percepción de errores.

Ajuste Dinámico de Herramientas:

Si una consulta puede usar varias herramientas, implementa un mecanismo para combinar resultados (por ejemplo, usando doc_search y graph_search en paralelo para consultas mixtas).

Beneficio: Responde mejor a preguntas complejas.