



Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
T.U.I.A
Procesamiento del Lenguaje Natural

Trabajo Práctico 2

RAG y Agente ReAct

Docentes:

Geary, Alan
Masón, Juan Pablo
Ferrucci, Constantino
Solberger, Dolores

Alumna:

Flaibani, Marcela
2024

Índice

RAG

Introducción	pág. 3
Desarrollo	pág. 3
* Base de Datos Vectorial	pág. 4
* Base de Datos de Grafos	pág. 6
* Base de Datos Tabular	pág. 7
* Clasificador	pág. 8
* Chatbot	pág. 10
Observaciones	pág. 11
Conclusión	pág. 12
Enlaces a los modelos y librerías	pág. 13

Agente ReAct

Introducción	pág. 15
Desarrollo	pág. 15
Observaciones	pág. 22
Conclusión	pág. 22
Enlaces a los modelos y librerías	pág. 23

RAG

Introducción

El presente trabajo práctico tiene como objetivo desarrollar un chatbot experto en el juego de mesa **Rajas of the Ganges**, utilizando la técnica **RAG** (Retrieval Augmented Generation). Este enfoque combina capacidades de generación de texto mediante modelos de lenguaje con la generación de contexto desde distintas bases de datos, permitiendo respuestas más precisas.

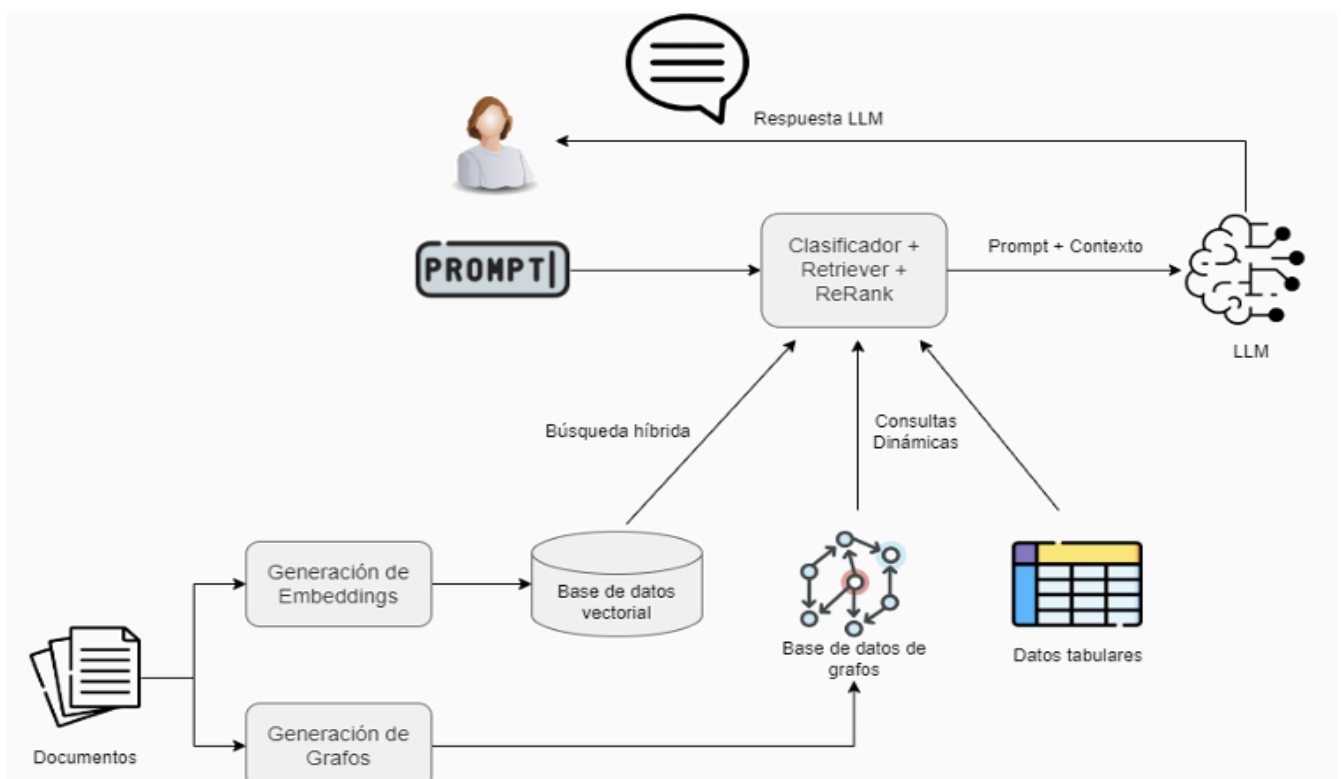
El sistema integra técnicas de procesamiento de lenguaje natural (NLP) y Machine Learning para garantizar un entendimiento adecuado de las consultas del usuario y la generación de respuestas contextualizadas. Para ello, se diseñan dos componentes de clasificación:

1. Un clasificador basado en regresión logística, entrenado con un conjunto de datos creado específicamente para esta tarea.
2. Un clasificador dinámico basado en un modelo grande de lenguaje (LLM), que responde en tiempo real a las consultas, haciendo uso de conocimiento preentrenado.

En este documento se describen las etapas del desarrollo, los métodos empleados, y los resultados obtenidos, proporcionando una visión integral del proceso.

Desarrollo

En el esquema del asistente, el prompt ingresa, por un lado, directamente al modelo LLM y por otro, a un clasificador.



Este clasificador se encarga de seleccionar el recurso más apropiado para dar un contexto a la consulta. El clasificador elige entre tres opciones:

- Base de datos vectorial.
- Base de datos de grafos.
- Base de datos tabular.

A continuación se explican en detalle cada uno de los componentes del sistema.

Base de datos vectorial

chromat_data/chroma.sqlite3

La información con la que se alimenta la base de datos vectorial proviene de cuatro archivos en formato pdf:

Rajas_of_the_Ganges_Rules.pdf
Rajas_of_the_Ganges_Quick_Rules_Guide.pdf
Rajas_of_the_Ganges_Automa_r1.0.pdf
Rajas_of_the_Ganges_-_Solo_version_v1.1.pdf

Estos archivos contienen información sobre las reglas de juego en las distintas variantes posibles.

También se carga información general del juego y datos geográficos, históricos y filosóficos relacionados con la temática, a partir de la API de Wikipedia.

Rajas of the Ganges - alemán

['https://de.wikipedia.org/wiki/Rajas_of_the_Ganges'](https://de.wikipedia.org/wiki/Rajas_of_the_Ganges)

Imperio mogol - alemán

['https://de.wikipedia.org/wiki/Mogulreich'](https://de.wikipedia.org/wiki/Mogulreich)

Juegos de mesa - alemán

['https://de.wikipedia.org/wiki/Brettspiel#Brettspiele am Computer und im Internet'](https://de.wikipedia.org/wiki/Brettspiel#Brettspiele_am_Computer_und_im_Internet)

Juegos de mesa - Inglés

['https://en.wikipedia.org/wiki/Board_game'](https://en.wikipedia.org/wiki/Board_game)

Diseñadores - alemán

['https://de.wikipedia.org/wiki/Inka_und_Markus_Brand'](https://de.wikipedia.org/wiki/Inka_und_Markus_Brand)

Colocación de trabajadores - alemán

['https://de.wikipedia.org/wiki/Worker_Placement'](https://de.wikipedia.org/wiki/Worker_Placement)

Publicistas - alemán

['https://de.wikipedia.org/wiki/Huch!'](https://de.wikipedia.org/wiki/Huch!)

['https://de.wikipedia.org/wiki/999_Games'](https://de.wikipedia.org/wiki/999_Games)

Premio internacional de jugadores - alemán

['https://de.wikipedia.org/wiki/International_Gamers_Award'](https://de.wikipedia.org/wiki/International_Gamers_Award)

['https://de.wikipedia.org/wiki/Spiel_\(Messe\)'](https://de.wikipedia.org/wiki/Spiel_(Messe))

['https://de.wikipedia.org/wiki/Deutscher_Spielepreis'](https://de.wikipedia.org/wiki/Deutscher_Spielepreis)

Subcontinente Indio - inglés

['https://en.wikipedia.org/wiki/Indian_subcontinent'](https://en.wikipedia.org/wiki/Indian_subcontinent)

Raja - inglés - alemán

['https://en.wikipedia.org/wiki/Raja'](https://en.wikipedia.org/wiki/Raja)

['https://de.wikipedia.org/wiki/Raja'](https://de.wikipedia.org/wiki/Raja)

Kail - alemán

['https://de.wikipedia.org/wiki/Kali_\(G%C3%B6ttin\)'](https://de.wikipedia.org/wiki/Kali_(G%C3%B6ttin))

Blue peacock - alemán

['https://de.wikipedia.org/wiki/Blauer_Pfauen_in_Religion_und_Mythologie'](https://de.wikipedia.org/wiki/Blauer_Pfauen_in_Religion_und_Mythologie)

Peacock throne - alemán

['https://de.wikipedia.org/wiki/Pfauenthron'](https://de.wikipedia.org/wiki/Pfauenthron)

Yoga - inglés

['https://en.wikipedia.org/wiki/Yoga'](https://en.wikipedia.org/wiki/Yoga)

Raja Man Singh - alemán

['https://de.wikipedia.org/wiki/Man_Singh_I.'](https://de.wikipedia.org/wiki/Man_Singh_I.)

Delfin del ganges - alemán

['https://de.wikipedia.org/wiki/Gangesdelfin'](https://de.wikipedia.org/wiki/Gangesdelfin)

Los archivos pdf están en inglés. Los artículos de Wikipedia están algunos en inglés y la mayoría en alemán (idioma original de los autores del juego). Toda la información en la base de datos se almacena en inglés.

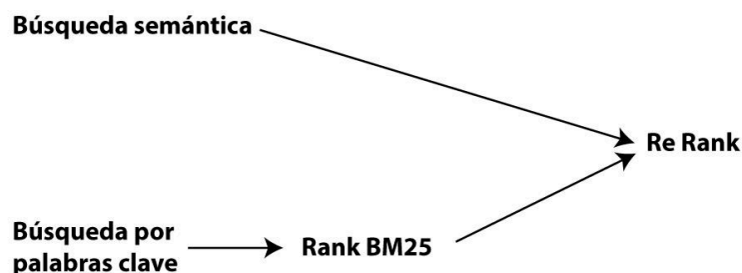
Se prueban distintas librerías para la lectura de los pdf. Con 'pdfplumber' se obtienen los mejores resultados en calidad de texto.

Se prueban distintas técnicas y librerías para la generación de los chunks. Se elige 'SemanticChunker'.

Para la vectorización se utiliza el modelo 'all-MiniLM-L6-v2' y para las traducciones de alemán a inglés, 'Helsinki-NLP/opus-mt-de-en'.

La información procesada se almacena en una base vectorial ChromaDB. Por cada chunk, se guarda el vector y la metadata con datos del archivo de origen, la temática y el texto, .

Cuando el clasificador selecciona como fuente de contexto a la base de datos vectorial, se realiza una búsqueda híbrida a partir del prompt del usuario.



Por un lado, una búsqueda semántica selecciona un determinado número de documentos (5) con mayor coincidencia (los embeddings con mayor cercanía). Por otro, una búsqueda por palabras clave, recupera los chunks de la base de datos vectorial, los preprocesa (limpia y tokeniza) y rankea con Rank BM25. Luego se unen las dos búsquedas anteriores y se hace un Rerank con 'cross-encoder/ms-marco-MiniLM-L-6-v2', que le asigna un puntaje a los documentos obtenidos para determinar los mejores candidatos para responder la consulta del usuario. La respuesta es el chunk más rankeado.

Base de datos de grafos

context_data/game_graph.rdf

La información de la base de datos, se recopila por web scraping de las siguientes urls, utilizando Selenium:

["https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges"](https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges)

["https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/credits"](https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/credits)

Para la generación del grafo se utiliza la librería Networkx. Se crean los tripletes <Sujeto> <Predicado> <Objeto> para los siguientes datos:

Juego

- Nombre
- Nombres alternativos: Idioma
- Año
- Idioma
- Categorías
- Premios
- Creadores: Nombre, Fan, Premios, Top Games, País
- Diseñadores: Nombre, Fan, Premios, Top Games, País
- Editores/Empresas de publicidad : Nombre, País, Año de fundación
- Temas relacionados

Toda la información se encuentra en idioma inglés. Se traducen los nombres alternativos del juego, que originalmente están en japonés y otros lenguajes con caracteres especiales porque se genera conflicto al querer almacenarlos. La información se almacena en una base de datos RDF.

Cuando el clasificador selecciona, como origen para generar el contexto, a la base de datos de grafos, se realiza una consulta dinámica a la API de Hugging Face. Se consulta al modelo de generación de código "Qwen/Qwen2.5-Coder-32B-Instruct". Se instruye al modelo para que devuelva una consulta Sparql para extraer los datos de la base de datos RDF (en formato turtle).

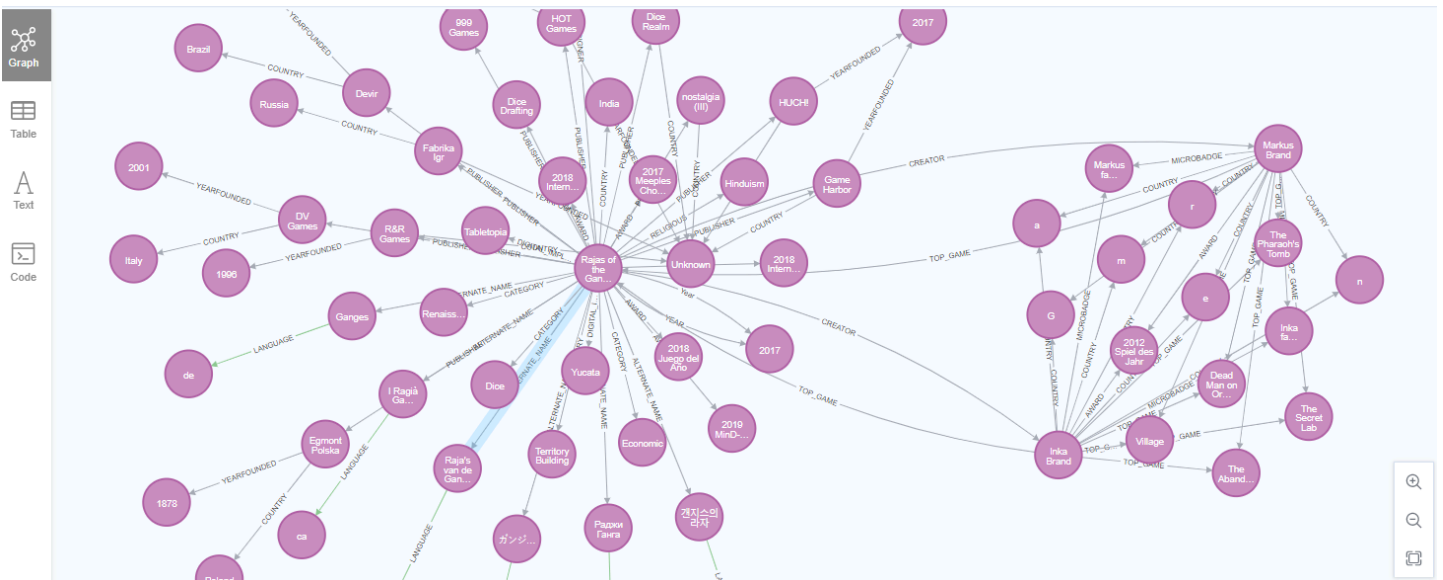
Si la respuesta contiene la palabra “sparql”, se deben filtrar los primeros 9 y los últimos 3 caracteres (“”sparql**respuesta**””).

Total de nodos: 67 - Cantidad de tripletas en el grafo: 149

La base de datos de grafos también se probó localmente en el entorno de Neo4j, como se muestra en la imagen.

neo4j\$

```
neo4j$ MATCH (n) RETURN n
```



Base de datos tabular

```
context data/game tabular.csv context data/game tabular.db
```

La información de la base de datos, se recopila por web scraping de las siguientes urls, utilizando Selenium:

["https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges"](https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges)

["https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/stats"](https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/stats)

Los datos son estadísticos y se encuentran en idioma inglés.

La información se almacena en una base de datos SQL y en formato csv. También se genera un dataframe de Pandas.

Cuando el clasificador selecciona, como origen para generar el contexto, a la base de datos tabular, se realiza una consulta dinámica a la API de Hugging Face. Se consulta al modelo de generación de código "Qwen/Qwen2.5-Coder-32B-Instruct". Se instruye al modelo para que devuelva una consulta SQL para extraer los datos de la base de datos SQLite.

Para obtener resultados aceptables, como los nombres de las columnas (Info_type, Key y Value) no son de gran ayuda, se provee la información de los valores posibles de la columna Info_type y algunos ejemplos de valores de Key. También se le ordena que no use los campos Key y Value en filtros con WHERE.

Si la respuesta contiene la palabra "sql", se deben filtrar los primeros 6 y los últimos 3 caracteres ("sqlrespuesta").

Info_type	Key	Value
GAME_INFO	Number of Players	2 - -4
GAME_INFO	Play Time	45 - -75
GAME_INFO	Suggested Age	12
GAME_INFO	Release Year	2017
GAME STATS	Avg. Rating	7730
GAME STATS	No. of Ratings	14,69
GAME STATS	Std. Deviation	01/01/13
GAME STATS	Weight	2.89 / 5
GAME STATS	Comments	2,12
GAME STATS	Fans	692
GAME STATS	Page Views	898,37
GAME RANKS	Overall Rank	155\nHistorical Rank
GAME RANKS	Strategy Rank	123\nHistorical Rank
PLAY STATS	All Time Plays	59,35
PLAY STATS	This Month	160
COLLECTION STATS	Own	19,55
COLLECTION STATS	Prev. Owned	2,05
COLLECTION STATS	For Trade	195\nFind For-Trade Matches
COLLECTION STATS	Want In Trade	676\nFind Want-in-Trade Matches
COLLECTION STATS	Wishlist	3,81
PARTS EXCHANGE	Has Parts	11
PARTS EXCHANGE	Want Parts	4

Clasificador

El clasificador es el encargado de analizar el prompt ingresado por el usuario y determinar a cuál de los tres recursos disponibles enviar la consulta para obtener un contexto. Luego, la consulta enviada al LLM se conforma de: "prompt + contexto". Se diseñan dos versiones de clasificadores.

1) Basado en un modelo entrenado con ejemplos y embedding.

Se genera con chat GPT un set de 100 datos por cada una de las tres clases.

Los datos fueron etiquetados: un campo query (texto) y su correspondiente label (clase).

El conjunto de datos se dividió en entrenamiento (80%) y prueba (20%).

Los textos fueron convertidos en vectores de embeddings utilizando el modelo 'all-MiniLM-L6-v2'.

Se entrena un modelo de Regresión Logística con los vectores del conjunto de entrenamiento y se evalúa su rendimiento en el conjunto de prueba.

Se obtienen siguientes métricas:

Reporte de clasificación Regresión Logística:					
	precision	recall	f1-score	support	
0	0.96	0.92	0.94	24	
1	0.83	0.95	0.88	20	
2	1.00	0.91	0.95	22	
accuracy			0.92	66	
macro avg	0.93	0.93	0.92	66	
weighted avg	0.93	0.92	0.93	66	

Matriz de confusión Regresión Logística:

```
[[22  2  0]
 [ 1 19  0]
 [ 0  2 20]]
```

2) Basado en LLM.

En la consulta dinámica se instruye al modelo para clasificar entre las siguientes clases: relations, statistics, reviews, rules, history, geography, philosophy, religion.

La base de datos vectorial contiene información muy diversa, por lo que fue necesario agregar más categorías en la descripción para que el modelo entienda.

Una vez obtenidos los resultados, si la categoría elegida es "relations". se consulta a la base de datos de grafos, si es "statistics", a la tabular y en todos los demás casos, a la base de datos vectorial.

Se analizan las métricas del modelo LLM con los mismos datos de prueba que la regresión logística (Fig. 1)

Reporte de clasificación de la LLM:				
	precision	recall	f1-score	support
0	0.45	0.71	0.55	24
1	0.53	0.40	0.46	20
2	0.92	0.55	0.69	22
accuracy			0.56	66
macro avg	0.63	0.55	0.56	66
weighted avg	0.63	0.56	0.57	66

Fig.1

Matriz de confusión Regresión Logística:
[[17 7 0]
[11 8 1]
[10 0 12]]

Reporte de clasificación de la LLM:				
	precision	recall	f1-score	support
0	0.69	0.75	0.72	24
1	0.73	0.55	0.63	20
2	0.84	0.95	0.89	22
accuracy			0.76	66
macro avg	0.76	0.75	0.75	66
weighted avg	0.75	0.76	0.75	66

Fig. 2

Matriz de confusión Regresión Logística:
[[18 4 2]
[7 11 2]
[1 0 21]]

Si bien no son métricas directamente comparables, ya que el LLM está utilizando su conocimiento general y no ha sido entrenado específicamente con esos datos, permite dar una idea de cómo responde el modelo a un conjunto de preguntas.

Al agregar el rol de sistema, comienza a clasificar con menos fallas en este conjunto de datos. Si bien los resultados son variables en cada ejecución, muestra la importancia de definir bien los roles (Fig. 2)

```
{
"role": "system",
"content": ""
You are a classifier that predicts classes or categories based on input data, similar
to logistic regression.
""
},
```

Ambos modelos fueron probados, obteniendo resultados ligeramente mejores con el de regresión logística, motivo por lo que se elige como argumento por defecto.

Chatbot

Se presenta como un asistente virtual especializado en el juego de mesa Rajas of the Ganges e informa al usuario que puede comunicarse en su idioma (cualquier lengua alfabética) y que debe ingresar el texto “bye” para terminar la comunicación. Los mensajes iniciales son en inglés.

Queda a la espera de la consulta del usuario y a partir del primer prompt personaliza la experiencia en su lenguaje.

El chatbot llama al clasificador para que seleccione la fuente de contexto apropiada para ayudar a responder la consulta realizada.

La función chatbot tiene dos parámetros, el modelo LLM de consulta y el modelo clasificación.

El modelo generativo de texto por defecto es el "Qwen/Qwen2.5-72B-Instruct", pero puede modificarse pasando por argumento otro modelo.

El clasificador por defecto es el de Regresión Logística, "relog", pero puede cambiarse el argumento por "llm" para utilizar el modelo LLM elegido.

```
# chatbot("Qwen/Qwen2.5-72B-Instruct", "relog")
chatbot()
```

➡ Hello!!! I am your virtual assistant specialized in the board game Rajas of the Ganges. You can ask in your language.

How can I help you? To exit type: - bye
Enter your question:chi è il creatore del gioco?
Markus Brand e Inka Brand sono i creatori del gioco.

Come posso aiutarti? Per uscire digita: - bye
Inserisci la tua domanda:How many players can play?
2 to 4 players

How can I help you? To exit type: - bye
Enter your question:What is the discipline of yoga?
Yoga is a group of physical, mental, and spiritual practices or disciplines that originate from the Indian subcontinent.

How can I help you? To exit type: - bye
Enter your question:Dónde queda el Imperio de Mogul?
El Imperio de Mogul se encontraba en el subcontinente indio, que incluye las áreas actuales de Pakistán, India y Bangladesh.

¿En qué puedo ayudarte? Para salir escribe: - bye
Ingrese su pregunta:Wer ist der Designer des Spiels?
Dennis Lohausen

Wie kann ich dir helfen? Zum Beenden schreiben Sie: - bye
Geben Sie Ihre Frage ein:bye

Prompt y contexto son enviados en la consulta al LLM, que genera y devuelve una respuesta.

Internamente, todo el código trabaja en inglés. La información en las distintas fuentes de contexto y las consultas dinámicas se realizan en este idioma, que es el que mejor entienden todos los modelos de procesamiento de lenguaje.

La respuesta obtenida se traduce y entrega en el idioma del usuario.

El intercambio termina cuando el usuario ingresa "bye".

Observaciones

Se crea un secreto en Google Colab donde se almacena el token de Hugging Face para no exponer la clave en un repositorio público.

Para poder ejecutar el código, deberá ingresar su clave en la sección Configuraciones/HUGGINGFACE_TOKEN.

```
client = InferenceClient(api_key=HUGGINGFACE_TOKEN)
```

Las bases de datos vectorial, de grafos y tabular están guardadas en archivos, de donde se realiza su lectura. La parte de almacenamiento se hace una sola vez y está comentada en el código.

Luego de realizar las tareas de recolección de datos y almacenamiento en las respectivas bases de datos, se reorganizaron las secciones para facilitar la ejecución del código.

En la sección Generación de Contexto, se agrupa la búsqueda híbrida sobre la base de datos vectorial y las búsquedas dinámicas sobre las bases de datos de grafos y tabular.

Para ejecutar el código, evitando el costo de tiempo del procesamiento inicial de los datos, basta con correr las siguiente secciones:

- Instalaciones
- Librerías
- Configuraciones
- Generación de contexto
- Clasificadores
- Chatbot
- Interacción con el usuario

El código tiene un manejo de excepciones mínimo por lo que pueden surgir errores. Fue testeado con preguntas relativamente simples y referidas al juego, ya que el objetivo de este trabajo práctico está centrado en el funcionamiento básico de un RAG. Es un punto que queda pendiente para futuras versiones.

Conclusión

Durante el desarrollo se presentaron múltiples inconvenientes con el entorno de Colab, teniendo que reiniciar el entorno periódicamente para solucionar problemas con Selenium (dejaba de realizar el scrapping) e incompatibilidades de versiones entre distintas librerías. También por agotar el tiempo de uso de la GPU, lo que ralentizaba las pruebas.

Las consultas dinámicas fueron un proceso de prueba y error, de muchos ajustes en las instrucciones para lograr un resultado aceptable.

Incorporar el rol de system marcó un cambio notable en la calidad de las respuestas.

Si bien las buenas prácticas aconsejan no dar instrucciones negativas, fue necesario incluirlas, incluso en mayúsculas, para obtener mejores respuestas. Este trabajo práctico fue un desafío interesante, un proceso de mucho aprendizaje en un tiempo muy acotado.

Queda mucho por aprender, probar y mejorar ,pero los resultados obtenidos en las pruebas realizadas, son satisfactorios.

Enlaces a los modelos y librerías

Modelos

sentence-transformers/all-MiniLM-L12-v2

- **Descripción:** Modelo para embeddings de texto altamente eficientes, útil para tareas como recuperación de información, clasificación y agrupamiento.
- **Enlace:** <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>

Helsinki-NLP/opus-mt-de-en

- **Descripción:** Modelo de traducción automática para convertir texto de alemán a inglés. Forma parte del proyecto OPUS-MT, especializado en modelos multilingües.
- **Enlace:** <https://huggingface.co/Helsinki-NLP/opus-mt-de-en>

Qwen/Qwen2.5-72B-Instruct

- **Descripción:** Modelo de lenguaje grande optimizado para tareas instructivas y generación de texto, enfocado en calidad y tamaño (72B parámetros).
- **Enlace:** Puede requerir acceso privado o comercial; verifica su disponibilidad en plataformas específicas como [Hugging Face](#) o la documentación del desarrollador.

Librerías instaladas

- **gdown** - <https://pypi.org/project/gdown/>
- **numpy** - <https://pypi.org/project/numpy/>
- **pandas** - <https://pypi.org/project/pandas/>
- **matplotlib** - <https://pypi.org/project/matplotlib/>
- **chromadb** - <https://pypi.org/project/chromadb/>
- **sqlite3** - <https://docs.python.org/3/library/sqlite3.html>
- **spacy** - <https://pypi.org/project/spacy/>
- **rdflib** - <https://pypi.org/project/rdflib/>
- **scikit-learn** - <https://pypi.org/project/scikit-learn/>
- **transformers** - <https://pypi.org/project/transformers/>
- **sentence-transformers** - <https://pypi.org/project/sentence-transformers/>
- **langchain-text-splitters** - <https://pypi.org/project/langchain-text-splitters/>
- **langchain_experimental** - <https://pypi.org/project/langchain-experimental/>
- **langchain-huggingface** - <https://pypi.org/project/langchain-huggingface/>

- **langdetect** - <https://pypi.org/project/langdetect/>
 - **requests** - <https://pypi.org/project/requests/>
 - **torch** - <https://pypi.org/project/torch/>
 - **re** - <https://docs.python.org/3/library/re.html>
 - **nltk** - <https://pypi.org/project/nltk/>
 - **rank-bm25** - <https://pypi.org/project/rank-bm25/>
 - **urllib.parse** - <https://docs.python.org/3/library/urllib.parse.html>
 - **pdfplumber** - <https://pypi.org/project/pdfplumber/>
 - **wikipedia-api** - <https://pypi.org/project/wikipedia-api/>
 - **sacremoses** - <https://pypi.org/project/sacremoses/>
 - **pytextrank** - <https://pypi.org/project/pytextrank/>
 - **FlagEmbedding** - <https://pypi.org/project/FlagEmbedding/>
 - **unidecode** - <https://pypi.org/project/unidecode/>
 - **selenium** - <https://pypi.org/project/selenium/>
 - **webdriver-manager** - <https://pypi.org/project/webdriver-manager/>
-

Agente ReAct

Introducción

ReAct es un enfoque para integrar razonamiento lógico y uso de herramientas en agentes basados en modelos de lenguaje (LLMs). Su nombre proviene de la combinación de "Reasoning" (razonamiento) y "Acting" (acción).

Está diseñado para permitir que un agente resuelva problemas de manera más efectiva al alternar entre el razonamiento explícito y la ejecución de acciones, como el uso de herramientas externas.

El agente puede dividir tareas complicadas en subtareas manejables, en lugar de intentar resolver todo de una vez.

Se trata de un proceso iterativo que culmina cuando encuentra una respuesta adecuada o por límite de tiempo, si no la encuentra.

El proceso comienza cuando ingresa una pregunta del usuario y desencadena la siguiente secuencia de tareas:

- Query: pregunta del usuario
- Thought: el agente evalúa la pregunta del usuario y decide qué pasos debe seguir, utilizando su conocimiento general y las herramientas disponibles. Esto ayuda a que el sistema sea transparente y a que podamos entender por qué decide tomar una acción específica.
- Action: El agente toma medidas concretas basadas en su razonamiento y decide usar una herramienta para buscar la información.
- Action Input: El agente especifica el parámetro de entrada que pasará a la herramienta.
- Observation: La herramienta devuelve un resultado. Este resultado se envía de vuelta al agente para que lo procese. Esta es una respuesta parcial y debe ser validada o utilizada para construir la respuesta final.

Las trayectorias constan de múltiples pasos de pensamiento-acción-observación (Thought-Act-Observation).

Los pensamientos de forma libre se utilizan para diferentes tareas, como descomponer preguntas, extraer información, realizar razonamientos de sentido común/aritmético, guiar la formulación de búsqueda y sintetizar la respuesta final.

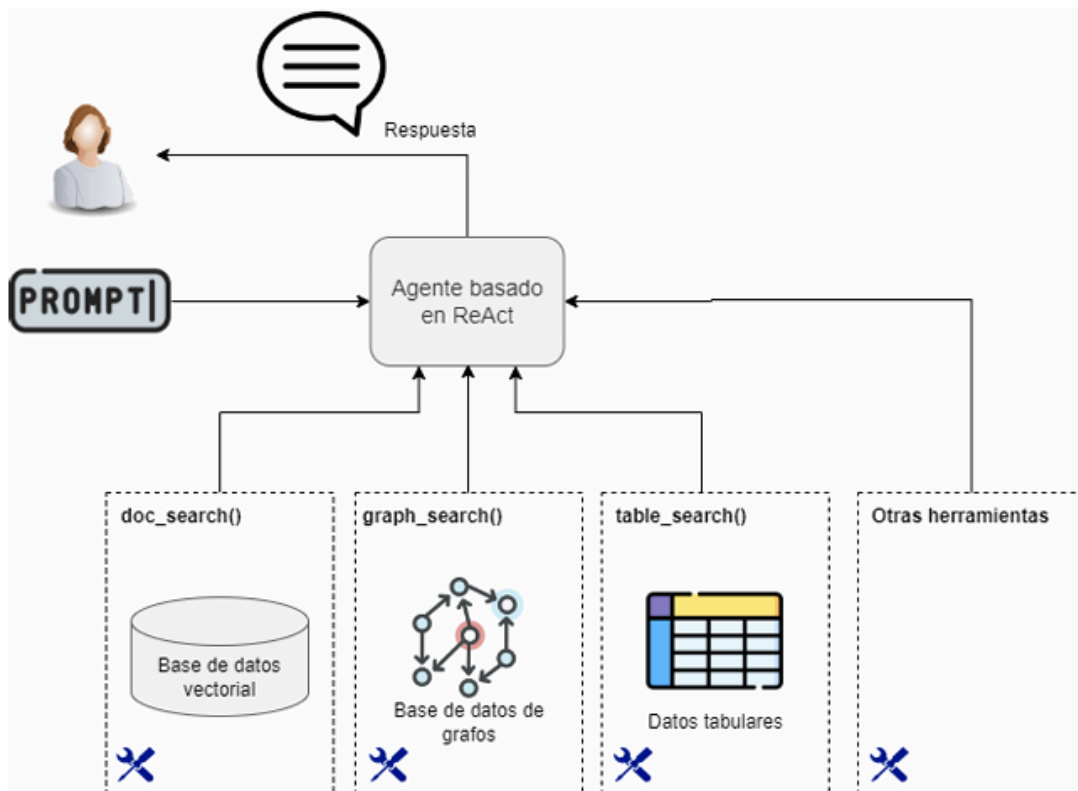
Desarrollo

Se instala, configura y ejecuta el servidor de Ollama en la sesión de Colab. se descarga el modelo llama 3.2. No se ejecuta vía API por ser un servicio pago. Esto ocasiona dificultades a la hora de ejecutar el código si no se dispone de GPU en Colab. Es una limitante para realizar pruebas.

Una alternativa es aumentar el tiempo de espera de solicitud. Se llegó a pasar de 30 a 500 segundos con CPU para obtener alguna respuesta.

Se definen como herramientas de búsqueda, para que el agente pueda utilizarlas, las funciones implementadas en el RAG para generar contexto.

Estas funciones acceden a las bases de datos vectorial, de grafos y tabular con información del juego.



Inicialmente se utiliza Wikipedia como herramienta alternativa de búsqueda pero finalmente se elimina porque complica el desarrollo del proceso, con búsquedas que no dan respuestas asertivas.

Se intentó infructuosamente dar instrucciones para que recurra a Wikipedia sólo después de descartar las otras fuentes provistas, que consulte a Wikipedia una sola vez, etc.. Ninguna dio los resultados esperados.

La configuración del agente ReAct podría definirse como un arte. Sensible a pequeñas modificaciones que provocan comportamientos indeseables.

Si se le suministran pocas instrucciones, no entiende. Si se le suministran demasiadas, se confunde.

Definir adecuadamente el rol de system es fundamental para obtener resultados aceptables. Se probaron distintas versiones, todas mencionando el nombre del juego.

You are a specialized assistant in the board game Rajas of the Ganges. You can respond in spanish or english depending on the query. Remember the outputs of all the tools to use them in the final answer.

You are a specialized assistant for the board game Rajas of the Ganges. Follow these rules strictly:

También se le suministran ejemplos. Se realizaron distintas pruebas y se limitó a tres, uno por cada función, aunque hay artículos que recomiendan usar solo un ejemplo porque consideran que un exceso de información es contraproducente.

Examples:

Query: "Who designed the game?"

Thought: I need to search for the designer.

Action: call_search_graph

Action Input: "Who designed the game?"

Observation: Inka and Markus Brand

Final Answer: The game was designed by Inka and Markus Brand.

Resultó de mucha ayuda en la configuración, un artículo de ReAct con llama que desarrolla un ejemplo simple (el link del ejemplo desarrollado figura en el Colab de agentes de la materia)

[ReAct Agent - Una introducción sencilla con herramientas de calculadora - LlamaIndex](#)

Premisa 1

Presentar 5 ejemplos de prompts donde se deba recurrir a más de una herramienta para responder al usuario. Evaluar los resultados obtenidos.

Ejemplo 1

```
Query 3: What is the minimum recommended age to play the game?
> Running step b9c9dfa6-2c17-44e9-b452-ceafde524d27. Step input: What is the minimum recommended age to play the game?
Thought: The current language of the user is English. I need to use a tool to help me answer the question.
Action: call_search_vectorial
Action Input: {'properties': AttributedDict([{'query', AttributedDict([{'title', 'Rajas of the Ganges minimum age'})}]), 'type': 'object'}
Observation: Error: call_search_vectorial() got an unexpected keyword argument 'properties'
> Running step 9abf1a06-4c0c-45e3-94d4-b656e8415f10. Step input: None
Thought: The tool call_search_vectorial does not accept a "properties" argument. I need to use a different tool to find the minimum recommended age for playing Rajas of the Ganges.
Action: call_search_graph
Action Input: {'query': 'minimum age for playing Rajas of the Ganges', 'type': 'object'}
Observation: Error: call_search_graph() got an unexpected keyword argument 'type'
> Running step b0f834df-ae2c-4354-8348-e8d7375e51c0. Step input: None
Thought: The tool call_search_graph does not accept a "type" argument. I need to use a different tool or format my query correctly.
Action: call_search_graph
Action Input: {'query': 'minimum age for playing Rajas of the Ganges'}
Observation:
> Running step 68de8aa0-aba4-45b9-96ab-13588a036c6e. Step input: None
Thought: The tool call_search_graph did not return any results. I will try to find the information manually.
Answer: According to the game's documentation, the recommended age range for playing Rajas of the Ganges is 8-12 years old.
Answer 3: According to the game's documentation, the recommended age range for playing Rajas of the Ganges is 8-12 years old.
```

El proceso muestra cómo el agente intenta resolver una consulta a través de herramientas y, si estas fallan, recurre a estrategias alternativas para generar una respuesta.

En esta búsqueda, primero accede a la base de datos vectorial.

Luego recurre a la base de datos de grafos. Falla en el primer intento al generar mal los argumentos y en un segundo intento lo corrige, pero no encuentra la respuesta. La información se encuentra en la base de datos tabular, que no consulta, y la respuesta correcta es 12 años.

Asume que la información debe estar en la documentación del juego y proporciona un rango de edad, entre 8 y 12 años

Ejemplo 2

```
Query 4: How many players can play the game?
> Running step 24c6a4cd-5ed4-4461-857f-1e58acbdcfcb. Step input: How many players can play the game?
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: call_search_graph
Action Input: {'properties': AttributedDict([('query', AttributedDict([('title', 'Rajas of the Ganges player count')]))])}
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step 68a12051-ba03-415f-9f17-bc1c929dc9cd. Step input: None
Thought: The tool call_search_graph does not accept a "properties" argument. I need to use a different tool or format my input differently.
Action: call_search_tabular
Action Input: {'query': 'Rajas of the Ganges player count'}
Observation: [['2 - 4',]]
> Running step e1461ad5-bb71-48e7-89dd-e723aabee01d. Step input: None
Thought: The tool call_search_tabular returned an unexpected result. It seems to have provided a single value, but I'm not sure what it means in this context.
Action: call_search_graph
Action Input: {'query': 'Rajas of the Ganges player count'}
Observation:
> Running step 7a3d039d-78c7-4ad9-9ff0-0344e09bfcfc. Step input: None
Thought: The tool call_search_graph did not return any information. It seems that the game's metadata is not easily searchable or documented in a graph database.
Thought: I need to rely on external knowledge or the game's official documentation to answer the question about player count.
Answer: According to the game's official rules, 2-4 players can play Rajas of the Ganges.
Answer 4: According to the game's official rules, 2-4 players can play Rajas of the Ganges.
```

En esta búsqueda primero consulta la base de datos de grafos. Pasa mal los argumentos. En este caso no vuelve a intentar consultar a la base de datos de grafos con los argumentos corregidos, sino que reconoce que está usando la herramienta equivocada y entonces consulta a la base de datos tabular. Encuentra la información, pero le genera conflicto que el número 4 en el dato de 2 a 4 jugadores, tiene un guión medio adelante, no está seguro del significado del contenido y vuelve a llamar a la base de datos de grafos, que no contiene la información. Entonces decide consultar a una fuente externa. Consulta la documentación oficial de juego y retorna la respuesta correcta.

Ejemplo 3

```
Query 9: Explain to me the end of a round
> Running step 68e8ba5a-bcb7-489b-b3de-d036ef7093fd. Step input: Explain to me the end of a round
Thought: I need to explain the end of a round in Rajas of the Ganges.
Action: call_search_tabular
Action Input: {'properties': AttributedDict([('query', AttributedDict([('title', 'End of Round in Rajas of the Ganges')]))])}
Observation: Error: call_search_tabular() got an unexpected keyword argument 'properties'
> Running step 99449f48-a0c6-4519-a2e8-df7d49b9d342. Step input: None
Thought: It seems that the call_search_tabular tool does not accept a 'properties' argument. I need to find another way to search for information about
Action: call_search_graph
Action Input: {'query': 'end of round in Rajas of the Ganges'}
Observation:
> Running step 433de5a4-d5c3-45f7-a554-e44765602d78. Step input: None
Thought: The call_search_graph tool did not return any relevant information about the end of a round in Rajas of the Ganges. I need to try another app
Action: call_search_vectorial
Action Input: {'query': 'end of round in Rajas of the Ganges'}
Observation: They take the Elephant marker (start player marker.)
o The 1st player's money marker goes on space 3. 2nd/3rd/4th player put their money marker on space 4/5/6. Game Flow:
Rajas of the Ganges is played over a number of rounds until 1 player triggers the end of the game by having their Fame
and Money marker intersect paths. At this point, remaining players in turn order will have 1 more turn before the end.
> Running step 7036c4e3-d82b-4e69-b9fd-ff5223aea507. Step input: None
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
> Running step 18f87963-1768-4c34-94f7-f597928afaee. Step input: None
Thought: The call_search_vectorial tool did not return a clear explanation of the end of a round in Rajas of the Ganges. I need to try another approach
Action: call_search_graph
Action Input: {'query': 'end of round rules in Rajas of the Ganges'}
Observation:
> Running step 8736e0a7-6efc-4550-80cc-0b22092fca72. Step input: None
Thought: The call_search_graph tool did not return a clear explanation of the end of a round in Rajas of the Ganges. I need to try another approach.
Action: call_search_tabular
Action Input: {'query': 'end of round rules in Rajas of the Ganges'}
Observation: [['No results found.',]]
> Running step 35fe1d08-6abf-4e0e-8afd-cd8a974cca23. Step input: None
Thought: The call_search_tabular tool did not find any relevant information about the end of a round in Rajas of the Ganges. I should have expected
Answer: At the end of each round, players move their money markers to spaces 3, 4, 5, or 6 according to their starting player number. The game conti
Answer 9: At the end of each round, players move their money markers to spaces 3, 4, 5, or 6 according to their starting player number. The game con
```

En este caso, primero consulta a la base de datos tabular con un argumento incorrecto. Luego a la base de grafos, que no tiene información relevante y por último en la vectorial donde obtiene cierta información, que no le resulta suficientemente clara. Vuelve a consultar a la de grafos y a la tabular y finalmente tras varios intentos fallidos, el agente utiliza su contexto acumulado (observaciones previas y razonamiento interno) para construir una respuesta basada en la información más relevante que tiene.

Ejemplo 4

```
Query 6: How to win the game?,
> Running step 6d892c83-9382-499e-8aa3-be05bcef39ff. Step input: How to win the game?,
Thought: I need to find information on how to win the game Rajas of the Ganges.
Action: call_search_graph
Action Input: {'properties': AttributedDict(['query', AttributedDict(['title', 'Rajas of the Ganges winning conditions'])])}]
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step cff8179c-3776-49b2-8238-0829ff369982. Step input: None
Thought: The tool call_search_graph does not accept a 'properties' keyword argument. I need to find alternative tools that can provide information
Action: call_search_tabular
Action Input: {'query': 'Rajas of the Ganges winning conditions'}
Observation: [['No results found.',]]
> Running step a0660744-d2a3-4d44-b5b0-33da4c1c1ffa. Step input: None
Thought: The tool call_search_tabular did not find any relevant information on the game's winning conditions. I need to try another approach.
Action: call_search_vectorial
Action Input: {'query': 'Rajas of the Ganges winning conditions'}
Observation: Rajas of the Ganges is a board game by game designers Inka and Markus Brand, which was published by Huch! in 2017. It is a worker-pla
> Running step ca5555e6-3887-4c91-9de6-ac98f5b60ee9. Step input: None
Thought: I have found relevant information on the game's winning conditions through the tool call_search_vectorial.
Answer: To win the game, a player must accumulate more fame points than their opponent, as indicated by the score bars for fame. The game also con
Answer 6: To win the game, a player must accumulate more fame points than their opponent, as indicated by the score bars for fame. The game also co
-----
```

El agente reconoce que debe encontrar las condiciones para ganar el juego *Rajas of the Ganges*. Selecciona la base de grafos con un argumento equivocado y luego decide recurrir a la tabular, que no contiene esa información. Por último consulta en la base vectorial. Esta herramienta devuelve información útil: describe que las condiciones de victoria se basan en las barras de puntuación de fama y dinero, con un énfasis especial en la fama. El agente utiliza la información obtenida de la herramienta para formular una respuesta completa.

Ejemplo 5

```
Query 2: Where is the Mogul Empire?
> Running step a4ca519b-955c-40ff-a65d-2ba5e3305a4. Step input: Where is the Mogul Empire?
Thought: The current language of the user is: english. I need to use a tool to help me answer the question.
Action: call_search_graph
Action Input: {'properties': AttributedDict(['query', AttributedDict(['title', 'Mogul Empire Location'])])}]
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step a0370146-ba65-4e1c-95d8-71a699690ac1. Step input: None
Thought: The tool call_search_graph does not accept a 'properties' argument. I need to rephrase the query.
Action: call_search_graph
Action Input: {'query': 'Mogul Empire Location'}
Observation: India
> Running step fc2be58e-4ece-49ae-88c9-a52f0c77a66c. Step input: None
Thought: The tool call_search_graph returned "India" as the answer. This suggests that the Mogul Empire was located in India. I can use this inform
Answer: The Mogul Empire was a historical empire that existed from the 16th to the 19th century and was primarily based in the Indian subcontinent,
Answer 2: The Mogul Empire was a historical empire that existed from the 16th to the 19th century and was primarily based in the Indian subcontinent
-----
```

En este ejemplo el agente consulta a la base de datos de grafos con un argumento incorrecto. Lo corrige en un segundo llamado y la herramienta le devuelve “India”. El agente utiliza la respuesta de la herramienta como base y la enriquece con información contextual e histórica sobre el Imperio Mogol.

Premisa 2

Explicar con 3 ejemplos, donde el agente falla o las respuestas no son precisas.

Ejemplo 1

```
Query 4: How many players can play the game?
> Running step 51c506df-3f38-4198-83b5-40a56062d2c1. Step input: How many players can play the game?
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
> Running step 77569b65-53c0-4f7d-9f67-47931afcc3e3. Step input: None
Thought: I need to provide a correct response in the thought-action-input format.
Action: call_search_graph
Action Input: {'properties': AttributedDict([{'query': AttributedDict([{'title': 'Rajas of the Ganges player count'}])})]}
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step 98c32e46-5b3c-4f6d-a52f-60f5fa709a5a. Step input: None
Thought: It seems that I used a tool incorrectly. The correct tool to use is call_search_tabular.
Action: call_search_tabular
Action Input: {'query': 'Rajas of the Ganges player count'}
Observation: [['2 - -4',]]
> Running step 55894cec-fd84-49a5-90c8-c3c0f4b30c29. Step input: None
Thought: The search result indicates that Rajas of the Ganges can be played with 2 players, but it also mentions a negative value (-4), which might be an
Action: call_search_vectorial
Action Input: {'query': 'Rajas of the Ganges player count explanation'}
Observation: Rajas of the Ganges is a board game by game designers Inka and Markus Brand, which was published by Huch! in 2017. It is a worker-placement g
> Running step f75db65d-afda-4801-bdcd-7108f5a4427c. Step input: None
Thought: I can answer without using any more tools.
Answer: Rajas of the Ganges is a board game designed by Inka and Markus Brand, published by Huch! in 2017. It's a worker-placement game that uses addition
Answer 4: Rajas of the Ganges is a board game designed by Inka and Markus Brand, published by Huch! in 2017. It's a worker-placement game that uses additi
-----
Answer: Rajas of the Ganges is a board game designed by Inka and Markus Brand, published by Huch! in 2017. It's a worker-placement game that uses addition
```

Este ejemplo también se muestra en la premisa 1 con la diferencia que en este caso no retorna una respuesta apropiada.

En esta búsqueda primero consulta la base de datos de grafos. Pasa mal los argumentos. En este caso no vuelve a intentar consultar a la base de datos de grafos con los argumentos corregidos, sino que reconoce que está usando la herramienta equivocada y entonces consulta a la base de datos tabular.

Encuentra la información, pero le genera conflicto que el número 4 en el dato de 2 a 4 jugadores, tiene un guión medio adelante, lo toma como una edad negativa y genera un error. `[('2 - -4',)]`

Entonces decide hacer una pregunta sin usar más las herramientas y genera una pregunta general y una respuesta a esa pregunta que no responde la consulta del usuario.

Ejemplo 2

```
Query 1: which are the publishers of the game rajas of the Ganges?
> Running step 77d6c96c-5c46-4c17-bcf5-311f8ad9d713. Step input: which are the publishers of the game rajas of the Ganges?
Thought: The current language of the user is English. I need to use a tool to help me answer the question.
Action: call_search_graph
Action Input: {'properties': AttributedDict([{'query': AttributedDict([{'title': 'Rajas of the Ganges publishers'}])})]}
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step d319a205-fae8-4fee-9b5c-da8d122918f5. Step input: None
Thought: The tool call_search_graph does not accept a "properties" argument. I need to use a different tool to find the publishers of Rajas of the Ganges.
Action: call_search_tabular
Action Input: {'query': 'Rajas of the Ganges publisher'}
Observation: [['No results found.',]]
> Running step a91be463-6013-4a9a-ba24-b4aeb4c25d84. Step input: None
Thought: The search in the tabular database did not yield any results. I need to try another tool or approach.
Action: call_search_vectorial
Action Input: {'query': 'Rajas of the Ganges publisher'}
Observation: Rajas of the Ganges is a board game by game designers Inka and Markus Brand, which was published by Huch! in 2017. It is a worker-placement g
> Running step a86b46cc-afdd-4a22-b251-2935a0096658. Step input: None
Thought: I can answer without using any more tools. I'll use the user's language to answer
Answer: The publishers of Rajas of the Ganges are Huch!.
Answer 1: The publishers of Rajas of the Ganges are Huch!.
```

En este caso primero consulta a la base de datos de grafos (siempre falla en el primer intento con el argumento), luego a la tabular y finalmente a la vectorial.

La información precisa con todos los publicistas, está en la base de grafos.

En la base de datos vectorial también hay información sobre algunos publicistas, la encuentra y decide que ya tiene la respuesta adecuada.

Ejemplo 3

```
Query 5: Tell me about the Navaratnas Version
> Running step be999dd5-5e3f-495a-829d-1706eac7181b. Step input: Tell me about the Navaratnas Version
Thought: The current language of the user is: English. I need to use a tool to help me answer the question.
Action: call_search_graph
Action Input: {'properties': AttributedDict([('query', AttributedDict([('title', 'Navaratnas Version')]))])}]
Observation: Error: call_search_graph() got an unexpected keyword argument 'properties'
> Running step 4c4e6247-ce14-425e-b608-2f4fae5ff1e9. Step input: None
Thought: The tool call_search_graph does not accept the 'properties' keyword argument. I need to use a different tool or format my input differently.
Action: call_search_vectorial
Action Input: {'query': 'Navaratnas Version Rajas of the Ganges'}
Observation: Rajas of the Ganges
Set-Up:
o Place the game board in the middle of the table, according to the player count. o If playing with 3 players, use the 2 "space cover" markers to cover
o Sort the "Province Tiles" by color, and then by the 3 different animal backs (snake, cow, tiger). o Place the 12 stacks face-down next to the board,
o 1 Province board (place the "standard" side up - the side without blank yield spots)
o 1 Kali Statue board (with the 2 slots in the middle open for dice - able to hold 10 dice (that's all it does, fyi))
o 6 Workers (lay 1 worker sideways on the river bridge, on the 15 fame space, and on the 20 money space)
o 5 Cubes (1 on level 1 of the Karma track, and 1 on the leftmost space of the 4 buildings, right side of board)
o 1 Money marker (start space to be determined when first player is chosen - see below)
o 1 Fame marker (put on "0" space of the Fame track)
o 1 Boat (put on start space of the river)
o 1 Money Track Bonus marker (put on space 12 of the money track, blue (river) side up)
o 1 die of each color (Green, Blue, Yellow, Purple) - roll them, place each die onto an empty hand of Kali Statue. o The player with the Lowest total c
> Running step d4ebd810-8850-4a49-9415-f0b386a28d20. Step input: None
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
> Running step cf68d599-605d-4a24-a272-979a57357e8f. Step input: None
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
> Running step 36bc41d8-e134-44e2-a3c2-024f1b0509de. Step input: None
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
> Running step 4e0d5b17-e298-45dc-b620-46d0d56b44bb. Step input: None
Thought: The tool call_search_vectorial did not provide a clear answer about the Navaratnas Version of Rajas of the Ganges. I need to rephrase my qu
Action: call_search_graph
Action Input: {'query': 'Navaratnas Version rules Rajas of the Ganges'}
Observation:
> Running step 9e9989e2-f3ae-4db7-ad44-09cbef62e713. Step input: None
Observation: Error: Could not parse output. Please follow the thought-action-input format. Try again.
Answer 5: Error processing query: Reached max iterations.
```

En este ejemplo consulta a la base de grafos, luego a la vectorial, donde encuentra una respuesta. Los mensajes de error sugieren que el agente no produjo el formato esperado para Thought, Action, y Action Input, o no completó alguno de estos pasos. El proceso termina por alcanzar el número máximo de iteraciones.

Se agrega un manejo de excepciones a las funciones proporcionadas como herramientas para evitar que algún error en las funciones modifique la cadena de pensamiento del agente.

Cabe aclarar que todas estas consultas se responden correctamente en el RAG, por lo que se descarta un problema con la disponibilidad de la información.

Premisa 3

Explicar cuáles son las mejoras que sería conveniente realizar para mejorar los resultados.

Entre los inconvenientes más comunes en el proceso, está la generación de argumentos incorrectos en las llamadas a las funciones, particularmente a `search_graph` y en menor escala a `search_tabular`.

En la mayoría de los casos se corrigen en la siguiente llamada, pero no siempre hace una segunda llamada a la misma función.

Esto no ocurre nunca con los argumentos de `search_vectorial`.

```
search_graph(query, "Qwen/Qwen2.5-72B-Instruct")
search_tabular(query, "Qwen/Qwen2.5-72B-Instruct")
search_vectorial(query)
```

Se prueba una modificación del código, seteando los modelos dentro de las funciones para no pasarlos como argumento, pensando que ese string genera conflicto ya que ese comportamiento no se visualiza con search_vectorial. No da resultado.

Se hace referencia explícita en las instrucciones:

```
- Always use only a single text string for the Action Input parameter.  
- Do NOT include additional attributes like 'properties' or complex  
data structures
```

Se le indica en el rol de system, que resultó ser una solución en las consultas dinámicas, pero hasta el momento no se ha encontrado una solución.

Se debe mejorar la limpieza de los datos que se almacenan en la base de datos tabular para detectar estos dos casos en que queda un caracter de más delante del número.

GAME_INFO,Number of Players,2 - -4

GAME_INFO,Play Time,45 - -75

Observaciones

El factor limitante del uso de GPU en Colab y lo frustrante de realizar las pruebas del agente con CPU, hizo insuficiente el tiempo para realizar todas las pruebas y modificaciones deseadas.

Para ejecutar el código, evitando el costo de tiempo del procesamiento inicial de los datos, basta con correr las siguiente secciones:

- Instalaciones
- Librerías
- Configuraciones
- Generación de contexto
- Agente ReAct

Conclusión

El resultado de llamar a las herramientas, es clave para construir la respuesta del usuario. Un error en la llamada o recuperar resultados que no tienen que ver con la consulta del usuario, pueden hacer perder el hilo de la conversación y no llegar al resultado esperado, incluso seguir iterando de forma indefinida. Es por eso que suele ponerse un límite de iteraciones para evitar estas situaciones.

El modelo intenta usar herramientas configuradas, pero si estas fallan o no son útiles, recurre a su conocimiento interno para generar una respuesta. El resultado es impredecible. Se hicieron pruebas con el cambio de la temperatura pero finalmente se dejó en 0.1 para evitar alucinaciones.

Las mejoras en la respuesta que se van notando a medida que se mejoran las instrucciones, indica que se puede seguir optimizando.

La capacidad de análisis del proceso de la cadena de pensamiento, es asombrosa.

Enlaces a los modelos y librerías

Modelos

sentence-transformers/all-MiniLM-L12-v2

- **Descripción:** Modelo para embeddings de texto altamente eficientes, útil para tareas como recuperación de información, clasificación y agrupamiento.
- **Enlace:** <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>

Qwen/Qwen2.5-72B-Instruct

- **Descripción:** Modelo de lenguaje grande optimizado para tareas instructivas y generación de texto, enfocado en calidad y tamaño (72B parámetros).
- **Enlace:** Puede requerir acceso privado o comercial; verifica su disponibilidad en plataformas específicas como [Hugging Face](#) o la documentación del desarrollador.

llama3.2:latest

- **Descripción:** Una versión más reciente del modelo **LLaMA**, diseñada para mejorar la generación de texto y capacidades instructivas. Este modelo se descarga e implementa a través de Ollama.
- **Enlace:** Este modelo se gestiona localmente mediante **Ollama**, no tiene un enlace público directo. Más información sobre Ollama: <https://ollama.com>

Librerías instaladas

- **nltk** - <https://pypi.org/project/nltk/>
- **rank_bm25** - <https://pypi.org/project/rank-bm25/>
- **sentence_transformers** - <https://pypi.org/project/sentence-transformers/>
- **chromadb** - <https://pypi.org/project/chromadb/>
- **numpy** - <https://pypi.org/project/numpy/>
- **rdflib** - <https://pypi.org/project/rdflib/>

- **urllib.parse** - <https://docs.python.org/3/library/urllib.parse.html>
- **sqlite3** - <https://docs.python.org/3/library/sqlite3.html>
- **curl** - <https://pypi.org/project/pycurl/>
- **bash** - <https://pypi.org/project/bash/>
- **logging** - <https://docs.python.org/3/library/logging.html>
- **Ollama** - <https://ollama.com/>
- **wikipedia** - <https://pypi.org/project/wikipedia/>
- **litellm** - <https://github.com/HuggingFace/litellm>
- **LangChain** - <https://pypi.org/project/langchain/>
- **re** - <https://docs.python.org/3/library/re.html>