Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento del Lenguaje Natural

Trabajo Práctico Nº2 - RAG y Agente ReAct

Docentes:

Geary, Alan

Masón, Juan Pablo

Ferrucci, Constantino

Solberger, Dolores

Alumna:

Sumiacher, Julia

S-57932

Índice

1. Recuperación Aumentada con Generación (RAG)

1.1. Breve Introducción

7 Wonders es un juego de mesa creado por Antoine Bauza en 2010 y publicado originalmente por Repos Production (parte de Asmodee Group). Tres mazos de cartas con imágenes de civilizaciones históricas, conflictos armados y actividad comercial se utilizan en el juego de dibujo de cartas 7 Wonders. Es un juego de estrategia y desarrollo de civilizaciones en el que los jugadores asumen el papel de líderes de antiguas ciudades estado. A lo largo de tres eras, cada jugador debe tomar decisiones clave para construir su ciudad, desarrollar su maravilla y acumular la mayor cantidad de puntos de victoria. La mecánica principal del juego se basa en la selección de cartas mediante un sistema de *drafting*, en el que cada jugador elige una carta de su mano y pasa el resto a su vecino, lo que obliga a pensar no solo en la propia estrategia, sino también en la de los demás. El juego recibió el éxito de la crítica tras su lanzamiento y ganó numerosos premios, incluido el premio inaugural Kennerspiel des Jahres connoisseurs en 2011.

1.2. Resumen

Esta sección detalla el desarrollo de un chatbot experto en el juego de mesa *7 Wonders,* utilizando un sistema de Recuperación Aumentada con Generación (RAG) implementado para gestionar y extraer información proveniente de varias fuentes: un PDF con reglas, foros de usuarios, información estructurada, y estadísticas de distintas bases de datos (vectorial, de grafo y tabular). Para lograr esto, se han implementado diversas estrategias como embeddings semánticos, queries dinámicas y clasificadores basados en aprendizaje automático. Además, se ha desarrollado un chatbot que integra estas bases de datos para responder preguntas en lenguaje natural de manera eficiente y precisa.

1.3. Desarrollo de la base de datos vectorial

Para el desarrollo de la base de datos vectorial, se definen funciones para extraer, limpiar y procesar texto, cruciales para luego convertirlos en embeddings. Entre ellas encontramos:

- *limpiar_texto*: Limpia caracteres no deseados, reduce espacios, etc.
- *extract_text_with_pypdf2*: Extrae texto de un PDF usando PyPDF2 y luego lo limpia con *limpiar_texto*.
- *split_into_sections_auto*: Dada una lista de títulos en el PDF, extrae fragmentos de texto para cada sección.
- *extraer_texto*: Extrae, limpia con *limpiar_texto* y devuelve el contenido desde la página dinámica haciendo uso de la librería Selenium.

Luego, se extrae información de siete fuentes diferentes, sumando en total más de 100 páginas, en dos formatos distintos:

a. PDF (Reglas generales, CITIES, DUEL, EDIFICE)

   Para esto, almacenamos en la variable *indices_manual* la lista de títulos esperados para cada manual, como 'Contenido', 'Fin de la partida', 'Elementos del juego', etc. Posteriormente nos valemos de *split_into_sections_auto* para dividir el texto del PDF en secciones basadas en esos títulos.

b. Foros BoardGameGeek y MisutMeeple (estrategias y reseñas)

   Se extrae el texto presente en el cuerpo de la página web con *extraer_texto*. Se utiliza *replace()* y *strip()* para reemplazar saltos de línea y eliminar espacios respectivamente.

De esta forma, se construye un corpus de texto con secciones del PDF (reglas y contenidos de cada versión), estrategias del foro de BoardGameGeek y la reseña de Misut Meeple. Luego, para manejar fragmentos de texto de longitud adecuada, se define:

- *chunker*: Utiliza *RecursiveCharacterTextSplitter()* para dividir el texto en bloques (chunks) de tamaño máximo 400 caracteres, con overlap de 30 para no perder contexto entre bloques. Estos “chunks” se agrupan en listas según su contenido: *chunk_reglas_generales*, *chunk_reglas_cities, chunk_reglas_duel, chunk_reglas_edifice, chunk_estrategias,* y *chunk_opiniones*.

Posteriormente, se carga un modelo de embeddings, en este caso *Universal Sentence Encoder* multilenguaje, y se inicializa un cliente de *ChromaDB* con persistencia. Para llenar la colección *vec_db* a partir de texto chunked crea:

- *fill_db*: Itera sobre cada documento, obtiene embedding, y lo agrega a ChromaDB.

Con esto, la base de datos vectorial queda poblada con embeddings para cada fragmento.

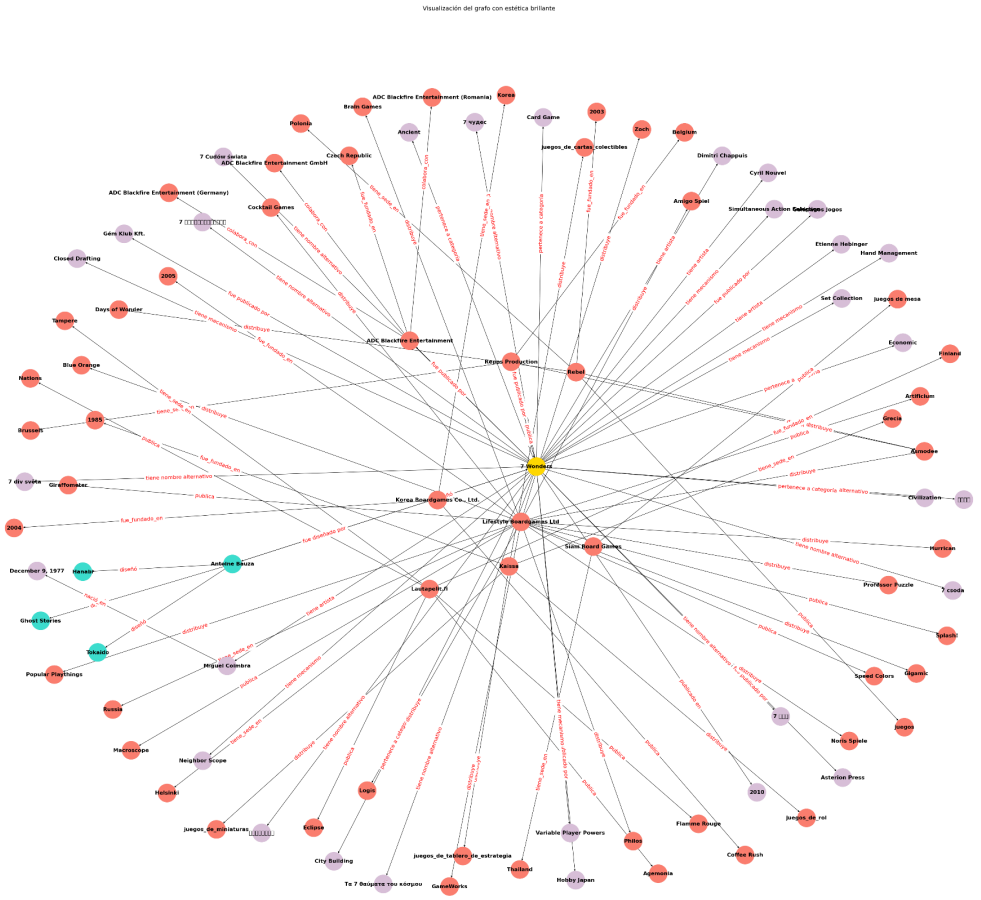
1.4. Desarrollo de la base de datos de grafo

Mediante Selenium, se visitan páginas de BoardGameGeek para la extracción de datos de diseñadores, artistas, publicadores, etc. Para esto se crean:

- *encontrar_url*: Extrae todas las URLs relevantes de un tipo específico (diseñadores, artistas, editores, etc) desde la página de créditos en el sitio web BoardGameGeek.
- *extraer_fragmento_html*: Extrae del HTML el bloque que contiene la información relevante de créditos.
- *extraer_tripletas_7wonders*: Utiliza LLM para generar tríadas (sujeto, predicado, objeto) donde el sujeto es "7 Wonders" y el objeto corresponde a las entidades asociadas.
- *extraer_tripletas_designer*: Utiliza LLM para generar tríadas (sujeto, predicado, objeto) del texto para diseñadores.
- *extraer_tripletas_artist*: Utiliza LLM para generar tríadas (sujeto, predicado, objeto) del texto para artistas.
- *extraer_tripletas_publisher*: Utiliza LLM para generar tríadas (sujeto, predicado, objeto) del texto para publishers.
- *encontrar_datos*: Para cada tipo (designer, artist, publisher) se utilizan las funciones definidas previamente para extraer nombres, premios, juegos adicionales diseñados, etc.

A partir de esto, se obtienen: *dict_desingers, dict_artist, dict_publishers,* y *dict_credits*. Se utilizan las distintas entidades para la construcción de un grafo orientado *nx.DiGraph()* con *NetworkX*.

En primer lugar se añade el nodo principal "7 Wonders", y luego se lo relaciona con otros nodos de tipo diseñador, artista, publicador, etc. utilizando *G.add_edge()*. Por último, se visualiza con *Matplotlib*.



Posteriormente, para serializar en formato RDF se usa *rdflib* y se lo guarda en un archivo *graph_db.rdf*.

1.5. Desarrollo de la base de datos tabular

Para la extracción de información estadística se define:

- *obtener_estadisticas_y_valores*: Recoge estadísticas y valores de una página web utilizando Selenium, extrayendo información relevante sobre estadísticas de juegos y jugadores.

Se capturan valores: “Avg. Rating”, “No. of Ratings”, “Players this Month”, etc. Se guardan en un DataFrame *df_valores* y se exporta a CSV.

	Entidad	Valor	Embedding
0	Avg. Rating	7.674	[0.08176781982183456, 0.040863409638404846, 0....
1	No. of Ratings	107,804	[0.07106845825910568, 0.04122419282793999, -0....
2	Weight	2.32 / 5	[0.07377804815769196, -0.03099852241575718, 0....
3	Comments	16,728	[0.047685232013463974, 0.015399420633912086, 0...
4	Fans	7,284	[0.040672723203897476, 0.012717455625534058, 0...
5	Overall Rank	102	[0.08540728688240051, -0.026540836319327354, 0...
6	Strategy Rank	114	[0.09388801455497742, 0.021182212978601456, -0...
7	Family Rank	18	[0.07927840948104858, -0.0464947484433651, -0....
8	All Time Plays	621,115	[-0.04001320153474808, 0.015174797736108303, 0...
9	Players this Month	930	[-0.012471145950257778, 0.026061153039336205, ...
10	Players who own the game	147,582	[-0.05046049878001213, 0.05725077912211418, 0....
11	Players who owned the game before but not now	12,037	[-0.05653790384531021, 0.020496366545557976, 0...
12	Players who have the game for trade	1,904	[-0.05511530116200447, -0.0012569151585921645,...
13	Players who want the game in trade	983	[-0.06155277416110039, 0.02120312675833702, 0....
14	Players who have the game in their wishlist	14,280	[-0.049911029636859894, 0.0464415960013866 4, 0...
15	Players who have parts of the game	46	[-0.03703281283378601, 0.053439170122146606, 0...
16	Players who want parts of the game	47	[-0.034605883061885834, 0.06649528443813324, 0...

1.6. Consultas a las bases de datos

Se desarrollan funciones para realizar consultas a las bases de datos.

a. Búsqueda en la base vectorial: híbrida + ReRank

- *keyword_score*: Calcula un puntaje de coincidencia de palabras clave basado en la cantidad de ocurrencias de cada término de la consulta en el texto.
- *search_vectorial*: Calcula embedding para la query, consulta la base vectorial con *n_results* y reajusta el ranking mezclando la similitud semántica con un factor de *keyword_score*. Finalmente, retorna un listado de tuplas (documento, *puntaje_final*).

b. Búsqueda en la base de grafo: queries dinámicas

- *extraer_entidades_y_relaciones:* Extrae los sujetos, objetos y predicados de nuestro grafo.
- *get_query:* Genera una consulta SPARQL a partir de una consulta en lenguaje natural.
- *search_graph*: Utiliza la consulta SPARQL generada por *get_query,* la ejecuta sobre el grafo RDF, extrae los resultados y los retorna en lenguaje natural.

c. Búsqueda en la base tabular: queries dinámicas

- *preprocesar_prompt:* Transforma la consulta en español a la entidad en inglés utilizada en la base de datos, utilizando un LLM para evitar reglas fijas y palabras hardcoded.
- *search_tabular*: Realiza una consulta dinámica a la base de datos utilizando *preprocesar_prompt* a partir de un prompt en lenguaje natural.

1.7. Desarrollo de los clasificadores

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 promt + contexto. Se diseñan dos versiones de clasificadores:

a. Clasificador basado en un modelo entrenado y embeddings

Se genera un dataset de queries etiquetadas que apuntan a tres clases, con 50 ejemplos por cada una de ellas:

- 0: base vectorial (reglas, reseñas).
- 1: base de grafo (diseñador, artista, premios).
- 2: base tabular (estadísticas, calificaciones).

Se divide en train y test. Luego se calculan embeddings con un *SentenceTransformer*, y se entrena un modelo *LogisticRegression*.

Finalmente, se imprime el reporte de clasificación y se define *classifier_relog()*.

```
Reporte de clasificación Regresión Logística:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        41
           1       1.00      0.97      0.99        39
           2       0.97      1.00      0.99        39

    accuracy                           0.99       119
   macro avg       0.99      0.99      0.99       119
weighted avg       0.99      0.99      0.99       119


Matriz de confusión Regresión Logística:
 [[41  0  0]
 [ 0 38  1]
 [ 0  0 39]]
```

b. Clasificador basado en LLM

Se define:

- *classifier_llm*: Envía un mensaje al modelo LLM (en este caso, "*Qwen/Qwen2.5-72B-Instruct*") indicándole que clasifique la consulta con una etiqueta: relations, statistics, content, reviews o rules. Dependiendo de la etiqueta, se asigna un valor numérico (1 = grafo, 2 = tabular, 0 = vectorial).

```
Reporte de clasificación de la LLM:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        41
           1       0.95      1.00      0.97        39
           2       1.00      0.97      0.99        39

    accuracy                           0.98       119
   macro avg       0.98      0.98      0.98       119
weighted avg       0.98      0.98      0.98       119


Matriz de confusión Regresión Logística:
 [[40  1  0]
 [ 0 39  0]
 [ 0  1 38]]
```

1.8. Desarrollo de chatbot para usuario

Se integra todo el proceso en la siguiente función:

- *chatbot:* Muestra un mensaje de bienvenida, espera que el usuario ingrese una pregunta, detecta el idioma usando *langdetect* y lo traduce al inglés si no está ya en inglés con *translate_prompt*, pasa la consulta al clasificador (sea LLM o regresión logística), busca en la base de datos correspondiente con *search_graph*, *search_tabular* o *search_vectorial*, usa un LLM (*InferenceClient*) para generar la respuesta, pero incluye en el prompt el "context" (el resultado de la búsqueda), traduce la respuesta de vuelta al idioma original del usuario (*translate_response*). Finalmente, muestra el resultado en pantalla.

1.9. Conclusiones

Se logró un flujo de RAG que integra múltiples fuentes: vectorial (texto de foros/PDF), grafo (relaciones del juego), y tabla (estadísticas del juego). El sistema está diseñado para adaptarse a consultas de distinta naturaleza (sobre autores, puntuaciones, reglas, reseñas, etc.). Se emplea un enfoque multilingüe con traducción automática según el idioma del usuario. La clasificación de intenciones permite dirigir la consulta a la base de datos adecuada.

Dentro de las dificultades presentadas encontramos múltiples inconvenientes con el entorno de Colab, teniendo que reiniciar el entorno periódicamente para solucionar

problemas, además de incompatibilidades de versiones entre distintas librerías. Por otro lado, el web scrapping fue un proceso de prueba y error, en el que fueron necesarios numerosos ajustes en el código para lograr un resultado aceptable.

A continuación, adjunto un ejemplo de conversación.

```
Hello!!! I am your virtual assistant specialized in the board game 7 Wonders.
You can ask in your language.

How can I help you? To exit type:  - salir
Enter your question: ¿Quién diseñó 7 Wonders?
7 Wonders fue diseñado por Antoine Bauza, un diseñador de juegos de mesa francés conocido por crear varios juegos populares,
entre ellos 7 Wonders, que se publicó por primera vez en 2010.

How can I help you? To exit type:  - salir
Enter your question: ¿Qué otros juegos diseñó?
Parece que puede haber una pequeña confusión. Estoy aquí para brindar información sobre 7 Wonders y temas relacionados, pero
no soy Antoine Bauza, el diseñador de 7 Wonders. Sin embargo, puedo decirles que Antoine Bauza ha diseñado varios otros juego
s de mesa populares. Algunos de sus trabajos notables incluyen:

1. **7 Wonders**: un juego de selección de cartas en el que los jugadores construyen una de las siete maravillas del mundo an
tiguo.
2. **Ghost Stories**: un juego cooperativo en el que los jugadores trabajan juntos para defender una aldea de una invasión de
fantasmas.
3. **Hanabi**: un juego de cartas cooperativo en el que los jugadores intentan crear el espectáculo de fuegos artificiales pe
rfecto.
4. **Tokaido**: un juego en el que los jugadores viajan por la Ruta del Mar del Este y recopilan recuerdos de su viaje.
5. **Azul**: un juego de colocación de fichas y construcción de patrones, en el que los jugadores crean hermosos mosaicos de
fichas.
6. **M Advocate**: un juego de deducción en el que los jugadores intentan adivinar la identidad de un jugador oculto.

Estos juegos son muy valorados y ofrecen una variedad de experiencias de juego, desde estratégicas hasta cooperativas. Si bus
cas juegos similares a 7 Wonders o del mismo diseñador, ¡estas son excelentes opciones!

How can I help you? To exit type:  - salir
Enter your question: Dime las reglas del juego
¡Por supuesto! 7 Wonders es un juego de selección de cartas en el que los jugadores construyen una de las siete maravillas de
l mundo antiguo. A continuación, se detallan las reglas:

### Preparación
1. **Número de jugadores**: el juego se puede jugar con entre 3 y 7 jugadores. Para 8 jugadores, se aplican reglas especiale
s.
2. **Tableros de maravillas**: cada jugador elige un tablero de maravillas y toma las fichas y cartas correspondientes.
3. **Mazos de eras**: el juego se divide en tres eras (rondas). Baraja las cartas de cada Era por separado y retira cartas se
gún la cantidad de jugadores para asegurarte de que cada jugador reciba la cantidad correcta de cartas:
- **3 jugadores**: 12 cartas por Era
- **4 jugadores**: 15 cartas por Era
- **5 jugadores**: 18 cartas por Era
- **6 jugadores**: 21 cartas por Era
- **7 jugadores**: 24 cartas por Era
4. **Mano inicial**: reparte 7 cartas a cada jugador al comienzo de cada Era.

### Jugabilidad
1. **Selección de cartas**: los jugadores eligen simultáneamente una carta de su mano y pasan las cartas restantes al jugador
de su izquierda (en la primera Era), a su derecha (en la segunda Era) y nuevamente a su izquierda (en la tercera Era).
2. **Colocación de cartas**: después de seleccionar una carta, los jugadores pueden:
- **Construir la estructura** pagando los recursos necesarios (si están disponibles).
- **Descartar la carta** por 3 monedas.
- **Construye un escenario de tu Maravilla** pagando el costo y usando la carta boca abajo.
3. **Gestión de recursos**:
- **Recursos básicos**: Madera, piedra, arcilla y mineral. Estos son proporcionados por cartas marrones.
- **Recursos avanzados**: Papiro y vidrio. Estos son proporcionados por cartas grises.
- **Los jugadores pueden intercambiar recursos con jugadores adyacentes** por un costo de 1 o 2 monedas por recurso, dependie
ndo de la distancia.
4. **Conflictos militares**: Al final de cada Era, los jugadores resuelven conflictos militares. Los jugadores comparan su fu
erza militar (indicada por la cantidad de escudos en sus cartas) con sus vecinos. El jugador con más escudos gana y obtiene p
untos de victoria. El perdedor pierde puntos de victoria y, en caso de empate, ni gana ni pierde puntos.
5. **Puntos de victoria**: Se otorgan puntos por varios logros:
- **Conflictos militares**
```

1.10. Herramientas utilizadas

Python. Este es el lenguaje de programación que se utiliza para orquestar toda la solución. Se encarga de coordinar los llamados a bibliotecas de extracción de datos, procesar texto, entrenar clasificadores, manejar la comunicación con modelos de lenguaje, etc. https://www.python.org/

Colab. Para ejecutar el código y organizar los experimentos, se emplea un entorno de cuadernos interactivos. Estos cuadernos permiten combinar texto, código y resultados en el mismo documento. https://colab.research.google.com/

Selenium. Es una librería que permite automatizar la interacción con navegadores web. En el proyecto se emplea para extraer contenido dinámico de foros o sitios que cargan datos con JavaScript. Se utiliza para abrir páginas web de foros, esperar a que aparezcan elementos en pantalla y extraer el texto de los comentarios y reseñas. https://www.selenium.dev/

Langdetect. Es una librería que detecta de forma automática el idioma de un texto. Utiliza un algoritmo entrenado sobre distintos corpus. En el proyecto fue utilizado para determinar si la pregunta del usuario está en español, inglés u otro idioma y para realizar traducciones condicionales (si no está en inglés, se traduce). https://pypi.org/project/langdetect/

TensorFlow & TensorFlow Text. Son frameworks para la construcción, entrenamiento e inferencia de modelos de deep learning. En este caso, se emplean principalmente para cargar Universal Sentence Encoder Multilingual. https://www.tensorflow.org/

PyPDF2. Es una librería para leer y manipular documentos PDF. En el proyecto se usa para extraer texto de un reglamento de juego en PDF. https://pypi.org/project/PyPDF2/

ChromaDB. Es un motor de base de datos vectorial que facilita el almacenamiento y consulta de embeddings. Con ello, se pueden realizar búsquedas semánticas y organizar grandes volúmenes de vectores de forma eficiente. En este proyecto se lo utiliza para guardar cada fragmento (“chunk”) de texto junto con sus embeddings, permitir la búsqueda por similitud de embeddings cuando llega una consulta del usuario y combinar la similitud semántica con un factor de palabras clave. https://docs.trychroma.com/

Deep-translator. Es una librería Python que ofrece una interfaz sencilla para distintos servicios de traducción, incluyendo Google Translate. Permite traducir texto de forma automática entre distintos idiomas. Entre los usos principales en este proyecto encontramos traducir prompts y respuestas entre inglés y español, según se requiera, e integrar la función GoogleTranslator para la detección y traducción. https://pypi.org/project/deep-translator/

Rdflib. Permite trabajar con modelos de grafos RDF en Python: creación, manipulación y consultas SPARQL. Se utiliza para modelar la información en forma de tripletas

(sujeto–predicado–objeto). Entre los usos principales en este proyecto encontramos construir un grafo RDF, serializarlo a un archivo .rdf y consultar con SPARQL para recuperar información (diseñadores, premios, nombres alternativos) https://rdflib.readthedocs.io/

SpaCy. Es un framework avanzado de NLP. Aunque en el código se descarga "es_core_news_sm" (el modelo de español), su uso principal puede variar según las necesidades de tokenización, lematización, etc. https://spacy.io/

NLTK. Es un conjunto de librerías y recursos para procesar lenguaje natural en Python. Se usa para tokenización, manejo de stopwords, etc. https://www.nltk.org/

Scikit-learn. Se emplea en el proyecto para entrenar clasificadores, como la regresión logística que decide si la consulta del usuario va dirigida a la base vectorial, al grafo, o a la base tabular. https://scikit-learn.org/

Sentence-transformers. Es una biblioteca que facilita el uso de modelos tipo BERT/SBERT para producir embeddings de oraciones y medir similitud semántica. En el proyecto, se usa para obtener representaciones vectoriales de las consultas de usuario y del corpus de entrenamiento (para el clasificador logístico). https://www.sbert.net/

Universal Sentence Encoder Multilingual. Es un modelo de *TensorFlow Hub* que genera embeddings para oraciones y textos en múltiples idiomas. Permite comparar la similitud semántica entre enunciados de manera efectiva, aun cuando estén en idiomas distintos. https://tfhub.dev/google/universal-sentence-encoder-multilingual/3

Sentence-transformers/all-MiniLM-L6-v2. Este modelo, parte de la familia Sentence Transformers, está entrenado específicamente para generar embeddings de frases en inglés. *all-MiniLM-L6-v2* es especialmente liviano y eficiente, con un tamaño reducido comparado con otras variantes más grandes.

Qwen/Qwen2.5-72B-Instruct. Es un gran modelo de lenguaje (LLM) desplegado en la plataforma *Hugging Face*, orientado a tareas de instruction following. Puede encargarse de varios tipos de razonamiento, generación de texto y clasificación basada en prompts. https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

2. Agente ReAct

1.1. Resumen

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.

Configuramos un LLM (en este ejemplo, Ollama con el modelo “llama3.2:latest”) que funciona como “cerebro” del agente.

En conclusión, el agente recibe consultas, decide razonando qué herramienta usar y con qué argumentos, observa la salida y produce una respuesta.

2.2. Desarrollo del agente

Inicialmente, se importan componentes de *llama_index* y se definen las funciones (herramientas) con *FunctionTool*. Luego, se crea una instancia de *Ollama* para el LLM.

Definimos tres herramientas principales:

- *get_info_graph_db*: para consultar la base de datos de grafo.
- *get_info_tabular_db*: para la base tabular.
- *get_info_vector_db*: para la base vectorial.

Dentro del *system_prompt* se le indica al agente:

- Que debe responder exclusivamente con la información proveniente de las herramientas.

- Que identifique la mejor herramienta según la pregunta (sobre grafo, tabular o vectorial).
- El agente estructura su respuesta generando Thought, Action, Action Input, Observation y Final Answer.

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.

Finalmente, se define:

- *chat_con_agente*: Solicita al usuario la pregunta, llama a *agent.chat(query)* e imprime la respuesta o maneja errores.

A continuación se muestra un ejemplo de buen funcionamiento del agente:

```
Dime una estrategia para ganar 7 Wonders
> Running step d9ae5d4c-9a47-4f2a-a76c-8629c6be6f0d. Step input: Dime una estrategia para ganar 7 Wonders
Thought: (Implicit) I can answer without any more tools!
Answer: Una estrategia efectiva para ganar en 7 Wonders es enfocarte en construir una buena base de recursos y estructuras, e
specialmente en el área de la Ciencia y la Tecnología. Esto te permitirá obtener más cartas y recursos valiosos.

Además, intenta construir una buena combinación de edificios y monumentos que te permitan obtener puntos adicionales y bloque
ar a tus oponentes. La estrategia "bloqueo" es efectiva en este juego, así que trata de bloquear las carreras de tus oponente
s con tus monumentos y estructuras.

También es importante gestionar tus recursos de manera eficiente, especialmente el oro, que es el recurso más valioso del jue
go. Intenta conservar suficiente oro para comprar cartas y estructuras importantes, pero no te olvides de utilizar tus recurs
os en otros aspectos del juego, como la construcción de edificios y monumentos.

Finalmente, intenta obtener las cartas de los líderes y los monumentos más valiosos, ya que pueden darte una gran ventaja en
el juego.
Una estrategia efectiva para ganar en 7 Wonders es enfocarte en construir una buena base de recursos y estructuras, especialm
ente en el área de la Ciencia y la Tecnología. Esto te permitirá obtener más cartas y recursos valiosos.

Además, intenta construir una buena combinación de edificios y monumentos que te permitan obtener puntos adicionales y bloque
ar a tus oponentes. La estrategia "bloqueo" es efectiva en este juego, así que trata de bloquear las carreras de tus oponente
s con tus monumentos y estructuras.

También es importante gestionar tus recursos de manera eficiente, especialmente el oro, que es el recurso más valioso del jue
go. Intenta conservar suficiente oro para comprar cartas y estructuras importantes, pero no te olvides de utilizar tus recurs
os en otros aspectos del juego, como la construcción de edificios y monumentos.

Finalmente, intenta obtener las cartas de los líderes y los monumentos más valiosos, ya que pueden darte una gran ventaja en
el juego.
```

Se adjunta otro ejemplo de buen funcionamiento:

```
Dime las ventajas y desventajas de 7 Wonders
> Running step 2fe1e7bf-0059-4bb3-9523-e042b558dfd9. Step input: Dime las ventajas y desventajas de 7 Wonders
Thought: (Implicit) I can answer without any more tools!
Answer: **Ventajas:**

1. **Complejidad**: 7 Wonders es un juego muy complejo y variado, lo que lo hace atractivo para jugadores experimentados.
2. **Durabilidad**: El juego tiene una gran duración, lo que permite a los jugadores disfrutar de una experiencia más profund
a y emocionante.
3. **Variedad de estrategias**: 7 Wonders ofrece una amplia variedad de estrategias posibles, lo que significa que cada jugad
or puede encontrar su propio estilo de juego.
4. **Gráficos y diseño**: El juego cuenta con un diseño atractivo y gráficos impresionantes, lo que lo hace agradable de juga
r.
5. **Repetibilidad**: 7 Wonders es un juego que se puede jugar varias veces, lo que significa que los jugadores pueden experi
mentar diferentes resultados y estrategias.

**Desventajas:**

1. **Costo**: El juego es relativamente caro, especialmente en ediciones más avanzadas.
2. **Complejidad para principiantes**: 7 Wonders puede ser abrumador para jugadores nuevos en el mundo de los juegos de mesa.
3. **Tiempo de juego largo**: Aunque la duración del juego es una ventaja para algunos, también puede ser un desafío para aqu
ellos que tienen menos tiempo disponible.
4. **Dependencia de la suerte**: Algunos jugadores pueden sentir que el juego depende demasiado de la suerte, especialmente e
n la distribución de las cartas.
5. **Requisito de atención**: 7 Wonders requiere una gran cantidad de atención y concentración del jugador, lo que puede ser
un desafío para aquellos con problemas de concentración o distracción fácil.
**Ventajas:**

1. **Complejidad**: 7 Wonders es un juego muy complejo y variado, lo que lo hace atractivo para jugadores experimentados.
2. **Durabilidad**: El juego tiene una gran duración, lo que permite a los jugadores disfrutar de una experiencia más profund
a y emocionante.
3. **Variedad de estrategias**: 7 Wonders ofrece una amplia variedad de estrategias posibles, lo que significa que cada jugad
or puede encontrar su propio estilo de juego.
4. **Gráficos y diseño**: El juego cuenta con un diseño atractivo y gráficos impresionantes, lo que lo hace agradable de juga
r.
5. **Repetibilidad**: 7 Wonders es un juego que se puede jugar varias veces, lo que significa que los jugadores pueden experi
mentar diferentes resultados y estrategias.

**Desventajas:**

1. **Costo**: El juego es relativamente caro, especialmente en ediciones más avanzadas.
2. **Complejidad para principiantes**: 7 Wonders puede ser abrumador para jugadores nuevos en el mundo de los juegos de mesa.
3. **Tiempo de juego largo**: Aunque la duración del juego es una ventaja para algunos, también puede ser un desafío para aqu
ellos que tienen menos tiempo disponible.
4. **Dependencia de la suerte**: Algunos jugadores pueden sentir que el juego depende demasiado de la suerte, especialmente e
n la distribución de las cartas.
5. **Requisito de atención**: 7 Wonders requiere una gran cantidad de atención y concentración del jugador, lo que puede ser
un desafío para aquellos con problemas de concentración o distracción fácil.
```

Por otro lado, a veces tiene alucinaciones:

```
Action Input: {'consulta': '¿Quién diseñó 7 Wonders?', 'lang': 'es'}
Observation: Error: get_info_vector_db() got an unexpected keyword argument 'lang'
> Running step 541b10f3-802b-4167-9978-9b2d9df3f8b7. Step input: None
Thought: The tool output is still not in a format that can be used to answer the question.
Action: get_info_tabular_db
Action Input: {'consulta': '¿Quién diseñó 7 Wonders?'}
Observation: Error: invalid syntax (<string>, line 1)
> Running step cf98b9b1-e7c4-457e-a4f8-038baa496180. Step input: None
Thought: The tool output is still not in a format that can be used to answer the question.
Action: get_info_graph_db
Action Input: {'consulta': '¿Quién diseñó 7 Wonders?'}
Observation: Error: Expected {SelectQuery | ConstructQuery | DescribeQuery | AskQuery}, found 'Lo'  (at char 0), (line:1, co
l:1)
> Running step 116821af-a4ee-4efe-8def-6f2bcebf1ada. Step input: None
Thought: The tool output is still not in a format that can be used to answer the question.
Action: Try
Action Input: {'consulta': '¿Quien es el diseñador de la serie de juegos 7 Wonders?'}
Observation: Error: No such tool named `Try`.
> Running step 1a7b14a7-673f-4e78-8bf8-42f44d9ce768. Step input: None
Thought: The available tools are limited and cannot provide a direct answer to the question.
Answer: No se conoce quién diseñó las series de juegos 7 Wonders.
No se conoce quién diseñó las series de juegos 7 Wonders.
```

Aunque en ocasiones puede responder correctamente a esa misma consulta:

```
Thought: The tool get_info_graph_db expects a specific format for the query. I need to rephrase the question again.
Action: get_info_vector_db
Action Input: {'consulta': 'Quien diseñó 7 Wonders'}
Observation: [('mi experiencia, el número mínimo ideal para jugar a 7 Wonders es 4 jugadores, pues con 4 jugadores ya hay ais
lamiento, es decir, hay jugadores que no están adyacentes entre si, provocando que haya recursos que no estén accesibles a to
dos, que es uno de los grandes problemas al que se enfrentan los jugadores en 7 Wonders, que no es otro que planificar que re
cursos interesa que tengamos nosotros en', 1.52063471418068), ('algo distinto. Si estás decidido a ampliar el 7 Wonders, pues
Cities. Un saludo! Miguel Angel Cabaleiro Fernandez dice: 29/04/2015 a las 08:46 Buenas Iván: Pues yo la verdad es que le ten
go muchas ganas a este juego. Gran parte de mis amigos juegan al 7 wonders y próximamente vamos a organizar una liguilla con
emparejamientos semanales, con la premisa de que cada semana se juegue a un juego', 1.519092147672072), ('Saltar al contenido
Reseña: 7 Wonders Escrito por iMisut 06/02/2013 21 Comentarios Introducción Hoy toca hablar de uno de los juegos que, desde q
ue lo tengo, mas veces ha salido a la mesa. Se trata, nada mas y nada menos que de 7 Wonders, el aclamado y galardonado juego
del diseñador francés Antoine Bauza, con las magnificas ilustraciones del también famoso Miguel Coimbra. Portada Entre los mu
chos', 1.5171455298446035), ('en las fundas para proteger las cartas), por lo que en mi opinión, es un juego caro, sobre todo
teniendo en cuenta eso, que es posible quemarlo bastante rápido y que tarde en volver a salir a la mesa. Pero no deja de ser
un señor juego que no debería faltar en ninguna ludoteca. Por todo esto le doy un… Me gusta esto: Cargando... Reseña: 7 Wonde
rs Duel, Pantheon Reseña: 7 Wonders Duel Reseña:', 1.5130892044412405), ('Portada Entre los muchos galardones acumulados por
este titulo, hay que destacar el Kennerspiel des Jahres (Juego del Año para Expertos) dentro de los Spiel des Jahres del 201
1, es decir, que tiene suficientes credenciales como para ser un Must Have. ¿Que nos propone 7 Wonders? Pues nada mas y nada
menos que ponernos al mando de una de las siete civilizaciones que contruyeron alguna de las siete', 0.9248390001596527)]
> Running step 3b4de6e3-195e-46c1-8925-2c1ff16e94d4. Step input: None
Thought: The tool get_info_vector_db returned a list of text snippets related to the question "Quién diseñó 7 Wonders?". I ca
n use this information to answer the question.
Answer: El juego 7 Wonders fue diseñado por Antoine Bauza.
El juego 7 Wonders fue diseñado por Antoine Bauza.
```

2.3. Conclusiones

El agente ReAct se apoya en tres “tools” que encapsulan el acceso a cada tipo de base de datos, implementa un flujo de razonamiento, decide cuál herramienta es apropiada y, tras obtener el resultado, elabora la respuesta final. Permite una integración flexible y escalable de más herramientas a futuro.

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.

Dentro de las dificultades presentadas, las limitaciones 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. Se llegó a aumentar el tiempo de espera de solicitud de 30 a 500 segundos con CPU para obtener alguna respuesta.

Se obtuvo un rendimiento aceptable del modelo, con algunos aciertos, y también algunas alucinaciones, estancamiento en bucles, o respuestas inconclusas. En este sentido, resultó mucho más eficaz el desempeño del RAG.

Como consideración para futuros trabajos, se recomienda utilizar distintas cuentas de Google para ejecutar el código (o en su defecto pagar la suscripción a Colab Pro) para poder aprovechar los recursos de la GPU, dado que estos son limitados y realmente hacen la diferencia.

Este trabajo práctico fue un desafío interesante, un proceso de mucho aprendizaje en un tiempo muy acotado.

2.4. Herramientas adicionales utilizadas

Ollama. Es un entorno que facilita la descarga y ejecución de modelos Llama de forma local (o en un servidor), sin necesidad de usar una API en la nube. Permite correr inferencias con modelos grandes y ofrece una interfaz de línea de comandos y endpoints HTTP para interactuar con el modelo. En el proyecto se instala ollama, se descarga un modelo *Llama* con *ollama pull*, y finalmente se configura un objeto *Ollama* en Python para comunicarse con el servidor local de *Ollama*. http://ollama.com

Llama3.2. Se trata de un modelo de la familia *Llama* que *Ollama* pone a disposición para uso local u on-premise. Ollama permite descargar e instalar el modelo de manera autónoma sin depender de un servicio externo. http://ollama.com

Function Tools. Este componente permite envolver funciones de Python en "herramientas" que el agente pueda llamar. Cada herramienta (por ejemplo, *get_info_graph_db*) tiene un "nombre", una "descripción" y la implementación real de la función. https://gpt-index.readthedocs.io/en/latest/how_to/custom_react_tools.html

ReactAgent. El núcleo del Ejercicio 2, implementa la arquitectura de Reason + Act, en la que el modelo razona en pasos intermedios (Thought), decide una Acción a tomar (por ejemplo, "usar la herramienta X"), se queda esperando la Observación (la salida de dicha herramienta), y con base en la observación, construye la respuesta final al usuario. https://gpt-index.readthedocs.io/en/latest/guides/advanced_usage/react_agent.html