TRABAJO PRÁCTICO N°2

Implementación de un Chatbot RAG y un Agente ReAct

Procesamiento del Lenguaje Natural

Calcia Franco (C-7363/2)

18 de diciembre de 2024

TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL - UNR

ÍNDICE

Resumen

Este trabajo tiene como objetivo desarrollar un chatbot basado en la técnica RAG (*Retrieval Augmented Generation*) y un agente ReAct para un juego de mesa estilo Eurogame. El chatbot permite responder preguntas utilizando tres fuentes de conocimiento: documentos textuales, bases de datos tabulares y una base de datos de grafos. La implementación incluye técnicas de análisis de texto, embeddings para la vectorización de datos, y búsqueda híbrida (semántica y por palabras claves).
Posteriormente, se incorpora un agente basado en el modelo ReAct, que utiliza herramientas dinámicas para consultar las tres fuentes de datos.

Introducción

En el contexto actual, los sistemas de recuperación aumentada de generación (RAG) y agentes autónomos basados en ReAct juegan un papel fundamental en el desarrollo de asistentes inteligentes. Este trabajo propone la implementación de un chatbot experto en juegos de mesa Eurogame, capaz de responder consultas en español, clasificando las preguntas y seleccionando las fuentes de datos relevantes.

Desarrollo/Implementación

Ejercicio 1: Implementación RAG

Armado de Bases de Datos

Base de Datos Vectorial

El primer paso consistió en generar la base de datos vectorial utilizando información extraída de diversas fuentes relacionadas con el juego Viticulture. Estas fuentes incluyen:

1. **Guía rápida (PDF):**
   Este documento contiene las instrucciones básicas para jugar y está disponible en el siguiente enlace. La extracción del texto se realizó con la librería PyPDF, seguido de un proceso de limpieza y la generación de índices relevantes. Para garantizar una mayor comprensión por parte de los modelos posteriores, se implementó un mapeo de índices para corregir errores de tipeo presentes en el texto original.

2. **Reseña del blog Misut Meeple:**
   Se realizó scraping sobre la reseña disponible en el siguiente enlace. Mediante inspección de elementos en la página, se identificaron las secciones relevantes para la extracción automatizada de contenido.
3. **Foro de BoardGameGeek:**
   Utilizando este enlace, se obtuvieron reseñas de usuarios empleando Selenium, debido a que la página es dinámica y no compatible con herramientas como BeautifulSoup. Se automatizó la navegación hacia cada reseña, extrayendo el título, URL, autor y contenido. Para mantener uniformidad, todas las reseñas fueron traducidas al español mediante la librería GoogleTranslator.

Toda la información extraída fue almacenada inicialmente en un archivo binario en formato .pkl. Posteriormente, los datos se organizaron en variables individuales, separando las reseñas traducidas, el texto de la reseña proporcionada por la cátedra y la información del PDF.

Creación de Chunks y Vectorización

Para construir la base vectorial, se utilizó el modelo *Universal Sentence Encoder (USE)* multilingüe de TensorFlow Hub. Los datos se segmentaron en "chunks" para optimizar el procesamiento y almacenamiento. Se definieron:

- *Tamaño máximo de chunks*: 400 caracteres. Este tamaño asegura que los fragmentos sean suficientemente compactos para mantener coherencia semántica y reducir la probabilidad de fragmentación excesiva.
- *Solapamiento entre fragmentos:* 30 caracteres. Esto permite continuidad entre los chunks y evita pérdida de información relevante.

Se organizaron los chunks en las siguientes categorías:

1. *Reglas generales del juego:*
   - Incluye secciones como preparación, resumen del turno y fin del año, extraídas del PDF.
2. *Reglas específicas:*
   - Secciones definidas como guía de construcción, trabajador grande, devaluar, cosechar un terreno, hacer vino, completar órdenes de vino y resumen de estaciones.
3. *Condiciones para ganar:*
   - Fragmentos relacionados con la sección "Ganar el juego", que explica las condiciones para finalizar el juego.
4. *Reviews de usuarios:*

    - Reseñas recopiladas y limpiadas, asegurando que no hubiera errores en la generación de los chunks.
5. *Reseña del blog:*
    - Contenido de la reseña extraído y dividido en fragmentos coherentes.

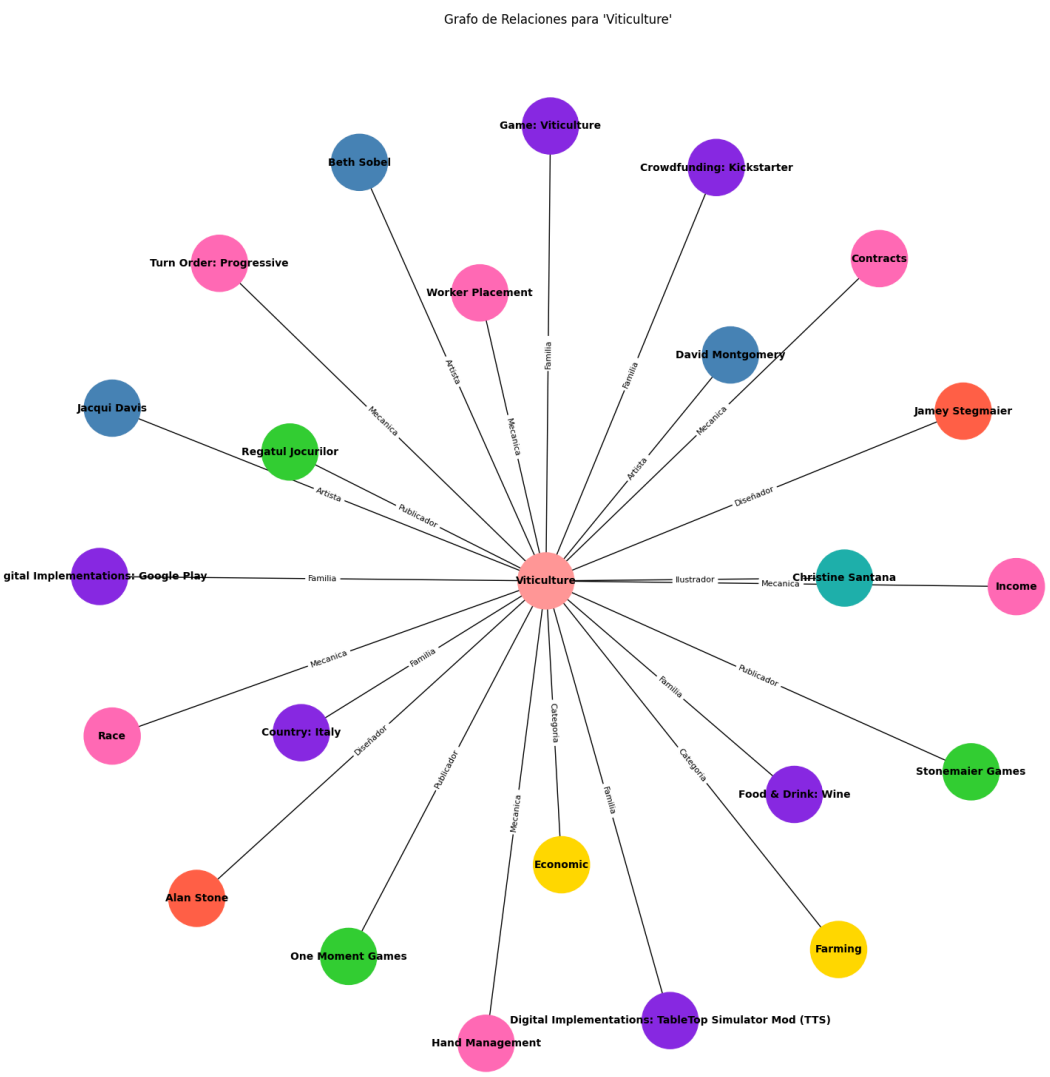
Finalmente, los chunks se almacenaron en la base vectorial ChromaDB, creando una colección denominada vec_db. A cada chunk se le asignó un identificador ("ID"), como por ejemplo: RG (Reglas Generales), RE (Reglas Específicas), GJ (Ganar el Juego), entre otros. Tras completar esta estructura, se probó una consulta para verificar su funcionamiento.

Base de Datos de Grafos

La información necesaria para la base de datos de grafos se obtuvo del siguiente enlace, donde se recopilaron datos sobre diseñadores, artistas, categorías del juego, diseñador gráfico, familias y mecánicas. Usando Selenium, se automatizó la navegación para recorrer los links de los diferentes nombres y extraer información clave.

Los datos recopilados se estructuraron en diccionarios separados (diseñadores, artistas, etc.). Para una mejor visualización, se utilizó NetworkX para construir un grafo, donde:

- El nodo central representa el juego (Viticulture).
- Los nodos conectados corresponden a elementos (diferenciados por color) como categorías ("Economía"), diseñadores ("Alan Stone"), y más.



Posteriormente, se creó un grafo RDF para estructurar las relaciones de manera semántica y permitir consultas sobre las entidades y sus relaciones. Se realizaron pruebas con consultas manuales para verificar la correcta recuperación de información.

Base de Datos Tabular

Finalmente, se generó la base de datos tabular extrayendo información numérica del enlace principal de BoardGameGeek. Mediante web scraping, se recopilaron datos como:

- Rating del juego.
- Año de lanzamiento.
- Precio del juego.
- Complejidad del juego.
- Cantidad de reviews del juego

- Mínimo de jugadores requeridos
- Máximo de jugadores necesarios
- Tiempo promedio del juego
- Edad mínima recomendada
- Complejidad
- Me gustas del juego

Los datos se almacenaron en un archivo CSV, permitiendo su consulta posterior mediante Pandas.

Desarrollo de Clasificadores

Modelo Basado en LLM (Unidad 6)

El primer clasificador desarrollado utiliza un modelo de lenguaje de gran escala basado en la Unidad 6 proporcionada por la cátedra. Este modelo, denominado QWEN, se inicializó mediante el cliente de Hugging Face y se configuró con un prompt que permite distinguir el origen de la información según la consulta realizada. Durante las pruebas, el modelo respondió de manera eficiente a varias consultas, demostrando un rendimiento satisfactorio en todas las tareas asignadas.

Modelo Basado en Ejemplos y Embeddings (Unidad 3)

El segundo clasificador implementado se basa en embeddings y ejemplos, siguiendo el enfoque propuesto en la Unidad 3. Se utilizó el modelo sentence-transformers/all-mpnet-base-v2, para el cual se creó un dataset manual con preguntas relacionadas a las reglas, reseñas y comentarios del juego. La información se dividió en un 70% para entrenamiento y un 30% para prueba. Sin embargo, este modelo no logró un rendimiento óptimo, obteniendo una *precisión* de apenas 0.55 y resultados bajos en otras métricas. Debido a esto, se descartó su utilización en favor del modelo basado en LLM.

Razones para Elegir el Modelo LLM

1. **Capacidad de Generalización:**
   A diferencia del enfoque basado en embeddings, el modelo LLM no requiere un dataset específico para interpretar preguntas. Esto permite que responda de manera efectiva incluso a consultas no vistas previamente, gracias a su capacidad de generalización.
2. **Ahorro de Tiempo y Recursos:**
   El modelo LLM evita el esfuerzo de construir y procesar un dataset extenso, una tarea

necesaria para el modelo basado en embeddings. Esta ventaja se traduce en una implementación más rápida y eficiente.

3. **Manejo de Lenguaje Natural:**
   La comprensión avanzada del contexto por parte del modelo LLM le permite interpretar la intención detrás de cada consulta, generando respuestas más completas y relevantes en comparación con las respuestas limitadas del modelo basado en embeddings.
4. **Flexibilidad en el Prompting:**
   Gracias a la estructura del prompt, el modelo LLM puede adaptarse a múltiples fuentes de información (grafos, bases tabulares y vectoriales) y ofrecer respuestas precisas y específicas de acuerdo al contexto.
5. **Mejor Rendimiento General:**
   El modelo LLM demostró un rendimiento significativamente superior al modelo basado en embeddings, incluso con un número reducido pero efectivo de preguntas de evaluación. Por otro lado, el modelo de embeddings presentó métricas desfavorables, como una precisión de apenas 0.55, lo que evidenció su limitada capacidad para manejar las tareas asignadas.

Organización de Bases de Datos

Las bases de datos se estructuraron en función de su contenido:

- *Base vectorial:* Incluye información segmentada como reglas generales, reglas específicas, objetivos del juego y condiciones para ganar.
- *Base de grafos:* Contiene información relacionada con diseñadores, artistas y publicadores del juego.
- *Base tabular:* Almacena datos numéricos, como rating, año de lanzamiento y complejidad del juego.

Consultas Dinámicas

Base de Datos de Grafos

Para esta base, se asignaron entidades y relaciones representativas del juego. Utilizando el modelo QWEN y un prompt específico, el sistema es capaz de interpretar consultas como "¿Quién diseñó el juego?" y generar las consultas SPARQL correspondientes. Las pruebas realizadas verificaron la precisión en las respuestas, corroborando la efectividad del prompt y el diseño de la base de datos.

Base de Datos Vectorial

La búsqueda en la base vectorial se estructuró en los siguientes pasos:

1. *Tokenización:* Se utilizó una función personalizada para dividir los textos en palabras individuales en español, filtrando caracteres no alfanuméricos.
2. *Modelo BM25:* Este modelo se empleó para realizar una búsqueda inicial basada en la relevancia de términos.
3. *Búsqueda Semántica:* Con el modelo USE, se generaron embeddings para la consulta y los documentos, comparándolos para determinar su similitud.
4. *Re-rankeo:* Se combinó la puntuación de BM25 y la búsqueda semántica, escalando las puntuaciones para obtener los documentos más relevantes.

Base de Datos Tabular

Para esta base, se configuró el modelo QWEN para procesar prompts específicos y devolver columnas específicas de la base tabular. Se realizaron consultas de prueba para corroborar la correcta recuperación de información.

Chatbot LLM para Consultas de Usuario

Se desarrolló un chatbot interactivo denominado llm_viticulture, diseñado para responder preguntas relacionadas con el juego. Este chatbot integra información de las tres bases de datos mediante los siguientes pasos:

1. *Obtención de Contexto:* El sistema clasifica la consulta del usuario y selecciona la base de datos adecuada (vectorial, gráfica o tabular). Dependiendo del tipo de base, se generan las consultas correspondientes.
2. *Generación de Prompt:* Utilizando el contexto recuperado, el sistema crea un prompt estructurado para interactuar con el modelo QWEN. Esto asegura que las respuestas sean precisas y relevantes.
3. *Conversación Interactiva:* El usuario puede realizar preguntas y recibir respuestas generadas por el modelo, con un flujo interactivo que permite realizar múltiples consultas.

El chatbot fue probado en varios escenarios y demostró ser capaz de manejar consultas complejas y diversas, ofreciendo respuestas coherentes y basadas en datos que nosotros le proporcionamos.

Ejercicio 2: Desarrollo del Agente ReAct

Este ejercicio extiende el trabajo del ejercicio 1 al incorporar un agente basado en el concepto ReAct, que combina herramientas para responder preguntas utilizando tres bases de datos: vectorial, gráfica y tabular. El agente fue desarrollado siguiendo los requisitos establecidos, implementando herramientas específicas y configurando un modelo de lenguaje avanzado.

Configuración del Entorno y Herramientas

Registro y Configuración de Logging:

- Se estableció un sistema de logging para monitorear el funcionamiento del agente y registrar las acciones realizadas durante su ejecución.

Modelo de Lenguaje y Configuración del Agente

Se utilizó el modelo Llama 3.2 configurado con las siguientes características:

- *Temperatura baja (0.1):* Para garantizar respuestas deterministas.
- *Ventana de contexto ampliada (4096 tokens):* Capacidad de procesar grandes cantidades de datos.
- *Tiempo de espera extendido:* Manejo de consultas complejas sin interrupciones.

El agente se implementó mediante ReActAgent con un prompt detallado que guía su comportamiento:

1. Analiza la consulta del usuario.
2. Selecciona la herramienta adecuada para la consulta.
3. Recupera la información requerida.
4. Genera una respuesta clara basada exclusivamente en los datos proporcionados por las herramientas.

Estructura del Agente ReAct

El agente incluye un sistema estructurado para responder consultas:

1. *Entrada de consulta:*
    - El agente analiza el texto de la consulta para determinar la base de datos relevante.
2. *Ejecución de herramientas:*
    - Dependiendo de la consulta, el agente llama a una o más herramientas, como:

        - *get_info_graph_db:* Recupera relaciones desde la base de grafos mediante consultas SPARQL.
        - *get_info_tabular_db:* Extrae datos específicos de la base tabular, evaluando consultas dinámicas.
        - *get_info_vector_db:* Realiza una búsqueda híbrida y re-ranking en la base vectorial.
3. *Generación de respuesta:*
    - Combina los resultados de las herramientas y devuelve una respuesta precisa y estructurada.

Pruebas y Evaluación del Agente

Comportamiento de los Prompts: Durante las pruebas, se utilizó una variedad de prompts para evaluar la capacidad del agente. Se observó que el desempeño del agente variaba dependiendo del prompt utilizado. Por ejemplo:

- Con un primer prompt, el agente respondía correctamente 4 de las 5 preguntas planteadas.
- Sin embargo, al cambiar el prompt, las mismas preguntas generaban sólo 3 respuestas correctas.
- Este patrón se repitió en varias iteraciones: un prompt podía ser efectivo para ciertas preguntas, pero ineficaz para otras.

Finalmente, se seleccionó un prompt que mostró el mejor desempeño general, logrando respuestas correctas en la mayoría de los casos, aunque sin alcanzar una efectividad del 100%.

Ejemplos exitosos:

1. *Consulta:* "¿Cuál es el rango de jugadores recomendado?"
    - Herramienta: *get_info_tabular_db*
    - Respuesta: "El rango recomendado es de 2 a 6 jugadores."
2. *Consulta:* "¿Quién diseñó el juego?"
    - Herramienta: *get_info_graph_db*
    - Respuesta: "Los diseñadores son Jamey Stegmaier y Alan Stone."
3. *Consulta:* "¿Cómo se gana el juego?"
    - Herramienta: *get_info_vector_db*
    - Respuesta: "El juego termina cuando un jugador alcanza 20 puntos de victoria."

Ejemplos de fallos:

1. *Consulta:* "¿Que puedo hacer con las cartas de color verde?."
    - Problema: La información específica de que se puede hacer con las cartas de color verde quizá no está formulada correctamente como para que el modelo pueda entender esa pregunta.
    - Respuesta: "No se encontró información para tu consulta."
2. *Consulta:* "¿Dónde se creó el juego?"
    - Problema: El lugar de creación no está disponible en ninguna fuente de datos.
    - Respuesta: "No se encontró información para tu consulta."
3. *Consulta*: "¿Cuáles son las mejores expansiones?"
    - Problema: La consulta excede las capacidades del modelo al no estar las expansiones clasificadas.
    - Respuesta: "No se encontró información para tu consulta."

Mejoras Propuestas

1. ***Ampliar las fuentes de datos:*** Incorporar información adicional, como estrategias y expansiones, para enriquecer el análisis.
2. ***Integración de nuevas fuentes de datos****:* Explorar la integración de plataformas como Wikipedia, BoardGameGeek y otras, para mejorar la cobertura y precisión de la información.
3. ***Optimizar prompts****:* Ajustar los prompts para manejar de manera más efectiva consultas ambiguas y mejorar la interacción.
4. ***Refinar herramientas****:* Mejorar la integración y recuperación de datos, especialmente en casos complejos.
5. ***Modelos más avanzados***: Implementar o ajustar modelos más avanzados que optimicen el rendimiento en este tipo de tareas.

Prompts Utilizados y Diferencias

A continuación, se presentan los cinco prompts utilizados para configurar el agente ReAct y las principales diferencias observadas entre ellos:

Prompt 1:

Tu rol: Eres un experto en el juego de mesa *Viticulture*. Respondes preguntas utilizando **únicamente** la información obtenida de las herramientas disponibles.

Herramientas disponibles:
1. **get_info_graph_db**: Proporciona detalles sobre diseñadores, artistas, publicadores, mecánicas, categorías, familias y nombres alternativos.

2. **get_info_tabular_db**: Ofrece información sobre el rango de jugadores, tiempo de juego, edad recomendada, complejidad, rating, cantidad de reseñas y precio.
3. **get_info_vector_db**: Proporciona información sobre las reglas, estrategias, objetivo del juego y cómo se gana.

Reglas de funcionamiento:
1. **Analiza la consulta** y selecciona la herramienta más adecuada según la información solicitada.
2. Realiza **exactamente una llamada por consulta**. Si necesitas combinar datos, llama a las herramientas secuencialmente.
3. **Nunca inventes información.** Si no encuentras una respuesta en las herramientas, di: "Lo siento, no encontré información sobre eso."
4. **Formato de respuesta**:
  - **Pensamiento (Thought):** Explica la lógica de por qué seleccionas una herramienta específica.
  - **Acción (Action):** Llama a la herramienta seleccionada.
  - **Entrada de acción (Action Input):** Repite la consulta exacta del usuario.
  - **Observación (Observation):** Incluye el resultado devuelto por la herramienta.
  - **Respuesta final (Final Answer):** Proporciona una respuesta directa y completa al usuario.

Prompt 2:

Eres un asistente especializado en el juego de mesa *Viticulture*. Tu misión es responder preguntas de forma rápida, amigable y precisa usando solo la información proporcionada por las herramientas disponibles.

Herramientas a tu disposición:
1. **get_info_graph_db**: Datos sobre el creador, artistas, mecánicas, categorías, publicadores y otros detalles generales.
2. **get_info_tabular_db**: Información específica como rango de jugadores, duración de la partida, complejidad, edad recomendada y estadísticas del juego.
3. **get_info_vector_db**: Reglas del juego, estrategias, objetivo y condiciones de victoria.

Cómo responder:
1. **Identifica** qué tipo de información se necesita basándote en las palabras clave de la consulta.
2. Llama a la herramienta **más adecuada** con la consulta proporcionada, **sin modificarla**.
3. Responde de forma clara y directa usando solo la información obtenida.

4. Si no hay respuesta disponible, di: *"No tengo esa información, pero puedo ayudarte con otra cosa sobre el juego."*

Prompt 3:

Actúas como un asistente especializado en el juego de mesa *Viticulture*, respondiendo a preguntas de manera precisa y eficiente usando herramientas especializadas.

Herramientas disponibles:
1. **get_info_graph_db**: Proporciona información sobre aspectos generales del juego como diseñadores, ilustradores, mecánicas y categorías.
2. **get_info_tabular_db**: Ofrece datos numéricos como rango de jugadores, tiempo de juego, complejidad, edad recomendada y precios.
3. **get_info_vector_db**: Contiene información textual detallada sobre reglas, estrategias y objetivos del juego.

Tu proceso para responder:
1. **Analiza la consulta** para identificar qué tipo de información se necesita y selecciona la herramienta adecuada.
2. Llama a la herramienta correspondiente usando exactamente la consulta recibida.
3. Evalúa la información obtenida y estructura una respuesta clara y concisa.
4. Si no se encuentra información, responde de forma general sin bloquear el flujo de la conversación.

Prompt 4:

Eres un asistente experto en el juego de mesa *Viticulture*. Tienes acceso a herramientas para obtener información precisa y útil sobre el juego. Tu tarea es responder a las preguntas del usuario de manera informativa y natural, utilizando únicamente datos obtenidos de las herramientas disponibles.

Herramientas Disponibles
1. **get_info_graph_db**: Accede a datos estructurados sobre diseñadores, ilustradores, mecánicas, categorías y otros aspectos relacionados con la creación del juego.
2. **get_info_tabular_db**: Proporciona información numérica como el rango de jugadores, duración de partida, complejidad, edad recomendada, y otros valores tabulares.
3. **get_info_vector_db**: Proporciona información textual, como reglas del juego, estrategias, objetivos y cómo se gana.

Tu Enfoque para Responder

1. **Identifica el propósito de la consulta**: ¿Busca el usuario información numérica, descriptiva, o relacionada con mecánicas y creadores?
2. **Selecciona las herramientas relevantes**: Decide cuáles de las herramientas son necesarias para obtener la respuesta.
3. **Procesa los resultados de las herramientas**: Reformula los datos de manera natural y clara, adaptándolos a la consulta del usuario.

Prompt 5 (Actual):

Tu rol: Responde preguntas sobre el juego 'Viticulture' usando únicamente información proporcionada por las herramientas disponibles.

Herramientas disponibles:
1. **get_info_graph_db**: Información sobre diseñadores, artistas, ilustrador, publicadores, categorías, nombres alternativos, mecánicas, familias.
2. **get_info_tabular_db**: Información sobre rating, año, cantidad de reviews, cantidad de jugadores, tiempo de juego, edad recomendada, complejidad, me gustas, precio.
3. **get_info_vector_db**: Información sobre reglas generales, reglas específicas, objetivo del juego, cómo se gana.

Instrucciones para cada consulta:
1. Analiza la consulta para determinar la herramienta adecuada.
2. Llama a una o varias herramientas usando exactamente la consulta recibida.
3. No inventes información. Solo responde con datos obtenidos de las herramientas.
4. Formato de respuesta:
  - Pensamiento (Thought): Explica qué información necesitas y la herramienta a usar.
  - Acción (Action): Llama a la herramienta adecuada.
  - Entrada de acción (Action Input): La consulta recibida.
  - Observación (Observation): La respuesta de la herramienta.
  - Respuesta final (Final Answer): Respuesta clara y completa basada en la información obtenida.

Diferencias Principales

- *Formalidad:* Los primeros prompts son más formales y estructurados, mientras que los últimos adoptan un enfoque más amigable y dinámico.
- *Nivel de detalle:* Prompts como el primero enfatizan un formato muy riguroso para las respuestas, mientras que otros, como el quinto, buscan balancear precisión y naturalidad.

- *Adaptabilidad:* El quinto prompt fue seleccionado como el más efectivo debido a su claridad, precisión y capacidad para evitar errores comunes como respuestas inventadas.

Resultados

Ejercicio 1: Chatbot.

Bienvenida

Mensaje inicial:

- Bienvenido al Chatbot de Viticulture.
- Evita palabras como "Quién", "Dónde", "Cuál", etc. Sé puntual, por favor.
- Para salir, escribe **"q"**.

Interacciones de Usuario y Respuestas del Agente

Consulta 1:
Usuario: ¿Quién gana el juego?
***Agente*:** El jugador que alcance al menos 20 puntos de victoria gana el juego. En caso de empate, se resuelve a favor del jugador que tenga más liras, luego el valor total más alto de los vinos en la bodega y finalmente el valor total más alto de las uvas en el lagar.

Consulta 2:
***Usuario*:** ¿Diseñadores del juego?
Agente: Los diseñadores del juego *Viticulture* son Jamey Stegmaier y Alan Stone.

Consulta 3:
***Usuario*:** ¿Categorías del juego?
Agente: Las categorías del juego *Viticulture* son **Economic** y **Farming**.

Consulta 4:
Usuario: ¿Edad recomendada para jugar?
Agente: La edad recomendada para jugar *Viticulture* es de **13 años en adelante**.

Consulta 5:
Usuario: ¿Precio del juego?
Agente: El precio del juego es **65.0**.

Consulta 6:
Usuario: ¿Estrategias del juego?
***Agente*:** En *Viticulture*, las estrategias se centran en:

1. Gestión eficiente de las cartas de contrato.
2. Administración de la mano.
3. Optimización de los ingresos.
4. Competencia en la carrera por ser el primero en ciertas acciones.
5. Colocación estratégica de los trabajadores en el tablero.

Es importante equilibrar estos aspectos para tener éxito.

Consulta 7:
Usuario: ¿Cómo se llama el juego?
Agente: El juego se llama *Viticulture*.

Consulta 8:
Usuario: ¿Principios del juego?
***Agente*:** Al principio del juego, los jugadores reciben:

- Una carta de uva Pinot (del mazo verde).
- Una carta de Visitante de Verano extraída al azar (del mazo amarillo).
- 3 liras (monedas).

Además:

- Comienzan con 2 Trabajadores regulares y 1 Trabajador Grande.
- Sus fichas de construcción se colocan cerca del tablero del jugador, pero no se usan hasta que se hayan pagado para construirlas.

Cierre

Usuario: "q"
Agente: Gracias por usar el Chatbot de Viticulture, ¡hasta luego!

Ejercicio 2: Agente ReAct.

Ejemplo de Interacciones con el Agente

Consulta 1: ¿Cómo se llama el juego?

- **Entrada:**
    - Usuario: ¿Cómo se llama el juego?

Acción Inicial: Se intentó consultar con get_info_graph_db, pero ocurrió un error:
Error: get_info_graph_db() got an unexpected keyword argument 'title'

- Segundo Intento: Se utilizó get_info_vector_db, pero no se obtuvo una respuesta clara.
- Tercer Intento: Se intentó con get_info_graph_db y get_info_tabular_db, pero hubo errores de sintaxis y consultas sin datos útiles.

Resultado Final:
Respuesta: Error al procesar la consulta: Reached max iterations.

Consulta 2: ¿Cuál es el rango de jugadores recomendado?

- **Entrada:**
  - Usuario: ¿Cuál es el rango de jugadores recomendado?

Acción Inicial: Se consultó con get_info_tabular_db obteniendo una respuesta parcial:
[6, "df_valores['Max jugadores'].values[0]"]

Resultado Final:

El agente generó una respuesta basada en la información disponible:
Respuesta: El rango de jugadores recomendado en Viticulture es de 2 a 6 jugadores.

Consulta 3: Dime una estrategia para principiantes

- **Entrada:**
  - Usuario: Dime una estrategia para principiantes

Acción Inicial: Se realizaron varios intentos de consulta con get_info_tabular_db, pero ocurrieron errores de sintaxis:
Error: invalid syntax (<string>, line 1)

Resultado Final:

El agente ofreció una respuesta general basada en su conocimiento
Respuesta: Una estrategia recomendada para principiantes en Viticulture es enfocarse en cultivar una variedad de vinos y no intentar producir demasiados vinos al mismo tiempo. También es importante gestionar el stock de materiales y priorizar la producción de vinos que tienen más demanda.

Conclusiones

Ejercicio 1

El desarrollo del sistema basado en RAG (Retrieval Augmented Generation) logró cumplir con los objetivos establecidos al integrar múltiples fuentes de datos en una base de datos vectorial, una base de grafos y una base tabular. Esto permitió una recuperación eficiente y contextualizada de la información relevante para responder consultas sobre el juego *Viticulture*.

En cuanto a la base vectorial, el uso del modelo Universal Sentence Encoder (USE) y una adecuada segmentación en chunks demostraron ser estrategias efectivas para estructurar los datos y optimizar la búsqueda híbrida. Las consultas realizadas a esta base evidenciaron una alta precisión en la recuperación de fragmentos específicos, destacando la importancia de un diseño bien planificado tanto en el procesamiento como en la organización de la información.

La base de grafos permitió modelar de manera semántica las relaciones y entidades clave del juego, como diseñadores, artistas, categorías, etc. Este enfoque facilitó consultas dinámicas y la recuperación de información estructurada mediante SPARQL, lo que demostró ser una herramienta versátil y robusta.

Finalmente, la base tabular complementó el sistema al proporcionar información cuantitativa y estadística sobre el juego, como ratings, precios, complejidad, entre otros. Este componente fue fundamental para responder preguntas específicas que requerían datos numéricos y categóricos.

Por lo tanto, el sistema basado en RAG destaca por su capacidad de integrar diversas fuentes de datos en un marco cohesivo y eficiente. Los resultados obtenidos validan la utilidad de este enfoque en la resolución de consultas complejas, aunque se identificaron áreas de mejora, como la optimización del pipeline de procesamiento de datos y la inclusión de técnicas avanzadas de limpieza y normalización. Este ejercicio sienta las bases para explorar futuras implementaciones y refinamientos en sistemas de recuperación de información multimodal.

Ejercicio 2

El desarrollo del agente ReAct mostró resultados prometedores en cuanto a su capacidad para integrar múltiples bases de datos y responder preguntas complejas de manera dinámica. Sin embargo, los resultados también dejaron en evidencia ciertas limitaciones que merecen ser destacadas y analizadas.

En términos de precisión, el agente logró responder correctamente la mayoría de las consultas, demostrando su potencial para procesar información y generar respuestas relevantes en función de los datos disponibles. Por ejemplo, consultas como "¿Quién diseñó el juego?" y "¿Cómo se gana el juego?" fueron respondidas de forma rápida y precisa. Esto refuerza la efectividad del agente para manejar datos estructurados y preguntas bien definidas.

No obstante, las pruebas también evidenciaron una dependencia significativa del prompt utilizado. Cambiar el prompt generaba variaciones notables en las respuestas, incluso para las mismas preguntas. En ocasiones, el agente proporcionaba respuestas inconsistentes o inventaba información si no encontraba datos relevantes, lo cual compromete su fiabilidad en escenarios más amplios. Aunque se logró identificar un prompt que ofreció el mejor desempeño general, este no alcanzó una efectividad del 100%, dejando un margen de mejora considerable.

Adicionalmente, la implementación de APIs externas como Wikipedia y BoardGameGeek no resultó beneficiosa. Estas APIs desviaban el foco del agente hacia datos externos, restando prioridad a las bases generadas durante el desarrollo. Por esta razón, se optó por excluir dichas integraciones y trabajar únicamente con las tres bases desarrolladas inicialmente: vectorial, gráfica y tabular.

Sin embargo, este trabajo destaca el potencial del agente ReAct como una herramienta poderosa para la recuperación y generación de información contextualizada. No obstante, las limitaciones observadas subrayan la importancia de seguir optimizando aspectos clave, como la robustez del prompt, la integración de nuevas fuentes de datos y la consistencia en las respuestas generadas. Este ejercicio no solo demuestra la capacidad del agente, sino que también abre la puerta a futuras iteraciones que puedan superar estos desafíos y ofrecer un sistema todavía más eficiente y confiable. La arquitectura ReAct, pese a sus desafíos actuales, sigue siendo una base sólida sobre la cual construir soluciones inteligentes en el ámbito de la recuperación de información.

Conclusión General

La combinación de los modelos desarrollados en ambos ejercicios destaca la importancia de integrar enfoques complementarios para resolver problemas complejos de recuperación de información y generación de respuestas contextualizadas.

En el ejercicio 1, el modelo basado en RAG demostró ser un sistema eficiente y robusto para manejar múltiples fuentes de datos, proporcionando respuestas precisas y relevantes en consultas específicas. Su estructura cohesiva y el uso de técnicas avanzadas de vectorización, grafo y datos tabulares sentaron una base sólida para tareas de recuperación de información.

Por otro lado, en el ejercicio 2, el agente ReAct extendió estas capacidades al incorporar herramientas dinámicas y una interacción más compleja con las bases de datos. Si bien logró responder adecuadamente a muchas consultas, el agente enfrentó dificultades recurrentes, como errores en el formato de las consultas a las herramientas, manejo limitado de excepciones y dependencia significativa del prompt utilizado. Estas limitaciones resaltan áreas clave de mejora, incluyendo:

- **Gestión de errores:** Implementar manejadores robustos para proporcionar respuestas claras y útiles cuando las herramientas fallan.
- **Validación de consultas:** Estandarizar el formato de las consultas antes de enviarlas, mejorando la interacción con las bases de datos.

Adicionalmente, la decisión de excluir APIs externas como Wikipedia y BoardGameGeek destacó la relevancia de priorizar las bases propias, lo que refuerza la necesidad de un enfoque bien delimitado en sistemas de recuperación de información.

En conjunto, ambos modelos subrayan la importancia de la planificación estratégica en el diseño de sistemas de recuperación de información, así como la necesidad de seguir iterando y refinando los enfoques implementados. Las lecciones aprendidas y los resultados obtenidos no solo confirman la eficacia de las técnicas utilizadas, sino que también abren camino hacia mejoras futuras que optimicen la precisión, la adaptabilidad y la confiabilidad de estos sistemas.

Bibliografía

➢ **LangChain**
LangChain Docs. (s.f.). *LangChain Documentation*. Disponible en: https://docs.langchain.com/

➢ **Sentence Transformers**
Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using

Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084*. Disponible en: https://www.sbert.net/

- **TensorFlow y TensorFlow Text**
  TensorFlow Developers. (s.f.). TensorFlow: *An End-to-End Open Source Machine Learning Platform*. Disponible en: https://www.tensorflow.org/

- **PyPDF2**
  PyPDF2 Developers. (s.f.). *PyPDF2 Documentation*. Disponible en: https://pypdf2.readthedocs.io/

- **Beautiful Soup**
  Richardson, L. (s.f.). *Beautiful Soup Documentation*. Disponible en: https://www.crummy.com/software/BeautifulSoup/

- **Deep Translator**
  Deep Translator Developers. (s.f.). *Deep Translator*. Disponible en: https://deep-translator.readthedocs.io/

- **Rank-BM25**
  Rank-BM25 Developers. (s.f.). *Rank-BM25 Documentation*. Disponible en: https://github.com/dorianbrown/rank_bm25

- **Chromadb**
  Chroma Developers. (s.f.). *Chroma Database Documentation*. Disponible en: https://docs.trychroma.com/

- **RDFlib**
  RDFlib Developers. (s.f.). *RDFlib Documentation*. Disponible en: https://rdflib.dev/

- **Llama Index**
  Llama Index Developers. (s.f.). *Llama Index Documentation*. Disponible en: https://www.llamaindex.ai/

- **Selenium**
  SeleniumHQ. (s.f.). *Selenium Documentation*. Disponible en: https://www.selenium.dev/

- **Hugging Face**
  Hugging Face. (s.f.). *Hugging Face Inference API*. Disponible en:

https://huggingface.co/

➢ **Natural Language Toolkit (NLTK)**
Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

➢ **BM25 Algorithm**
Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333-389.

➢ **Documentos Proporcionados por la Cátedra**
Disponible en: https://campusv.fceia.unr.edu.ar/course/view.php?id=503

Anexos: Referencias a Código y Detalles de Implementación

Para aquellos interesados en explorar el código fuente, ejemplos de consultas realizadas, y la ejecución de cada celda, se encuentra disponible un archivo `.ipynb` en Google Colab. Este archivo está cuidadosamente comentado para mejorar la legibilidad y facilitar la comprensión de cada paso del proceso.

Además, todo el proyecto, incluyendo el código y documentación, está alojado en el siguiente repositorio de GitHub:

- **Repositorio GitHub:** NLP-Calcia

Se invita a los lectores a consultar estas fuentes para obtener más información técnica o reproducir los resultados presentados en este informe.