Tecnicatura Universitaria en Inteligencia Artificial

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Trabajo Práctico Final**

Procesamiento del Lenguaje Natural

Alumna:

* Herrera Morena (H-1187/8)

Profesores:

* Juan Pablo Manson.
* Alan Geary.

ÍNDICE

[RESUMEN 3](#_Toc186321990)

[INTRODUCCIÓN 3](#_Toc186321991)

[METODOLOGÍAS 3](#_Toc186321992)

[EJERCICIO 1 4](#_Toc186321993)

[Resumen 4](#_Toc186321994)

[Desarrollo 4](#_Toc186321995)

[Extracción de datos para la BDD tabular 4](#_Toc186321996)

[Extracción de datos para la BDD de grafos 4](#_Toc186321997)

[Extracción de datos para la BDD vectorial 5](#_Toc186321998)

[Clasificadores 6](#_Toc186321999)

[Generar consultas 7](#_Toc186322000)

[Creación del chatbot final 7](#_Toc186322001)

[Resultados 8](#_Toc186322002)

[Pruebas de ejecución Finales 8](#_Toc186322003)

[EJERCICIO 2 9](#_Toc186322004)

[Resumen 9](#_Toc186322005)

[Desarrollo 9](#_Toc186322006)

[Herramienta graph\_search() 9](#_Toc186322007)

[Herramienta table\_search() 9](#_Toc186322008)

[Herramienta doc\_search() 9](#_Toc186322009)

[Creación del agente ReAct 9](#_Toc186322010)

[Otros prompts 10](#_Toc186322011)

[Resultados 14](#_Toc186322012)

[Pruebas de Ejecución 14](#_Toc186322013)

[Ejecuciones donde falla el agente 15](#_Toc186322014)

[CONCLUSIÓN GENERAL Y MEJORAS 15](#_Toc186322015)

[Logros 15](#_Toc186322016)

[Mejoras Futuras 16](#_Toc186322017)

[Conexión entre Ejercicios 1 y 2 16](#_Toc186322018)

[Conclusión 16](#_Toc186322019)

[ANEXO 16](#_Toc186322020)

# RESUMEN

En el trabajo práctico final de Procesamiento del Lenguaje Natural (NLP), se implementó un chatbot experto en el eurogame *Rajas of the Ganges* utilizando la técnica Retrieval Augmented Generation (RAG). Posteriormente, el ejercicio incluye la extensión de este chatbot a un agente ReAct.

La creación y funcionamiento del chatbot implican un desarrollo que va desde la extracción y limpieza de datos para crear las bases de datos (tabular, de grafos y vectorial) con la información obtenida, hasta la generación de consultas e implementación de modelos LLM.

El sistema desarrollado es capaz de clasificar las preguntas de los usuarios y realizar consultas en las diferentes bases de datos para responderlas de manera precisa.

# INTRODUCCIÓN

El informe tiene el objetivo de explicar y fundamentar, presentando de manera detallada el desarrollo de la solución implementada, las decisiones tomadas en cada etapa del proyecto y los resultados obtenidos.

Además, las mejoras consideradas para un mejor rendimiento.

* Ejercicio 1: se construye el sistema de recuperación y generación de respuestas mediante RAG.
* Ejercicio 2: se implementa un agente basado en ReAct, y que combina herramientas para mejorar la interacción y la precisión del chatbot.

Para ambas implementaciones, se utiliza información proveniente de bases de datos tabulares, de grafos y vectorial. Además, la generación de consultas con filtros de PANDAS y CYPHER para la tabular y la de grafos, respectivamente. Para la vectorial se utiliza búsqueda híbrida (semántica y por palabras clave) acompañada del mecanismo ReRank.

Se implementan diferentes versiones de un clasificador de preguntas, tanto basado en LLM como en modelos entrenados con ejemplos y embeddings, para evaluar cuál de estas metodologías proporciona mejores resultados.

# METODOLOGÍAS

1. Extracción de Datos: Recopilar datos desde diferentes fuentes.

* Selenium: Extracción de datos numéricos y relacionales de la página BoardGameGeek (BGG).
* BeautifulSoup: Análisis y extracción de datos HTML para identificar y extraer detalles específicos de las páginas web (en combinación con Selenium).
* pdfPlumber: Extracción de datos tabulares de archivos PDF, útil para obtener información estructurada de documentos de juegos o artículos relacionados.

2. Procesamiento de Texto: Segmentación en chunks y limpieza.

* LangChain: División de textos largos en fragmentos más pequeños para mejorar el manejo y procesamiento del contenido textual.
* Expresiones Regulares: Limpieza y transformación de textos crudos a un formato estructurado, facilitando la extracción de información relevante.

3. Almacenamiento de Datos: Uso de bases de datos y almacenamiento eficiente.

* Pandas: Manipulación y almacenamiento de datos tabulares extraídos en un archivo .csv para su análisis.
* Neo4j: Almacenamiento de datos relacionales sobre diseñadores, artistas, publicadores, categorías y mecánicas del juego en formato de grafo.
* ChromaDB: Almacenamiento de embeddings generados y fragmentos de texto para facilitar la búsqueda de información semántica.

4. Generación de Embeddings y Búsqueda Semántica: Mejora de la precisión en la búsqueda.

* SentenceTransformers: Generación de embeddings para fragmentos de texto con el fin de realizar búsquedas semánticas y mejorar la relevancia de los resultados.
* BM25: Complemento a la búsqueda semántica para mejorar la precisión de los fragmentos relevantes en la base de datos vectorial.

5. Modelos de Lenguaje y Clasificación: Predicción y generación de respuestas precisas.

* LogisticRegression (Modelo entrenado - U3): Entrenamiento de un modelo de clasificación basado en embeddings de preguntas y etiquetas de la base de datos para predecir la fuente de información.
* Zephyr (Modelo LLM de Hugging Face): Análisis de las preguntas de los usuarios para determinar la fuente de datos más adecuada de la cual se recuperará la información.
* Qwen (Modelo LLM de Hugging Face): Interpretación de preguntas en lenguaje natural y conversión de estas en consultas dinámicas para las bases de datos tabular y de grafos, o en consultas para la base de datos vectorial.
* CrossEncoder (ReRank): Clasificación y reordenación de fragmentos de texto más relevantes para generar respuestas precisas.

6. Ejecución de Consultas: Herramientas de consulta para bases de datos estructuradas.

* eval(): Ejecución dinámica de consultas generadas por el modelo Qwen en la base de datos tabular.
* graph\_search(query): Herramienta para consultar la base de datos de grafos y recuperar información sobre diseñadores, artistas, publicadores, categorías y mecánicas del juego.
* table\_search(query): Herramienta para consultar la base de datos tabular y recuperar información estructurada sobre los jugadores, duración del juego, ranking, etc.
* doc\_search(query): Herramienta para realizar búsquedas semánticas en la base de datos vectorial y obtener información sobre reglas, conceptos clave, acciones y reglas del juego.

7. Agente de Búsqueda y Respuesta: Uso de modelos de lenguaje para mejorar la interacción.

* Llama-Index: Librería utilizada para crear un agente basado en el modelo ReAct que selecciona y utiliza las herramientas adecuadas para interactuar con las bases de datos.
* Llama 3.2 (modelo de lenguaje): Modelo de lenguaje utilizado para analizar consultas del usuario y ejecutar la acción adecuada (selección de herramientas y consultas).

# EJERCICIO 1

## Resumen

El objetivo de este ejercicio, fue desarrollar un chatbot experto en el eurogame *Rajas of the Ganges* empleando la técnica RAG que integra diversas fuentes de datos para generar respuestas precisas. Se trabajó con tres fuentes de conocimiento principales:

1. Datos tabulares: contiene información básica como el número mínimo y máximo de jugadores, la edad mínima, la duración del juego, las calificaciones generales y de estrategia, la complejidad y el año de publicación.
2. Documentos de texto: contiene información sobre la descripción general, cómo ganar, conceptos clave, acciones, componentes y reglas.
3. Base de datos de grafos: contiene información que requiere relaciones como diseñadores, publicadores, artistas, mecánicas y categorías.

## Desarrollo

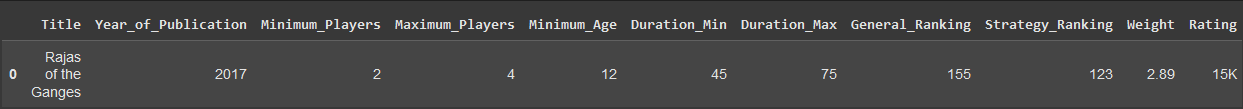
### Extracción de datos para la BDD tabular

Uso de Web Scraping con Selenium para la extracción de datos numéricos relevantes de la página BoardGameGeek (BGG). Los datos que posteriormente se almacenan son:

* Título del juego.
* Mínimo de jugadores.
* Máximo de jugadores.
* Edad mínima recomendada.
* Mínima duración en minutos.
* Máxima duración en minutos.
* Ranking general.
* Ranking de estrategia.
* Complejidad.
* Rating.

#### Construcción

Los datos se estructuran y guardan en un archivo .csv, facilitando su análisis y logrando una manipulación eficiente con Pandas.



### Extracción de datos para la BDD de grafos

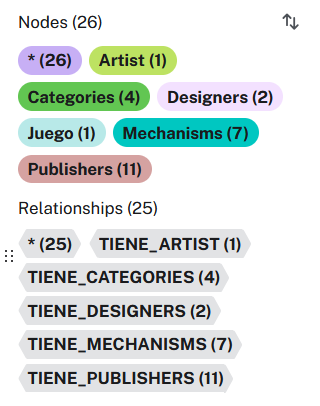
Uso de Web Scraping con Selenium para la extracción de datos que requieren de relaciones. Estos provienen de la página BGG, en particular de la sección ‘Full Credits’ del juego *Rajas of the Ganges*.

Información que se puede encontrar:

* Diseñadores
* Artistas.
* Publicadores.
* Categorías.
* Mecánicas.

#### Construcción

Los datos se estructuran en un diccionario y guardan en un archivo .json. Posteriormente, se almacena en una base de datos Neo4j. A continuación, se puede observar el grafo.



El uso de una base de datos de grafos permite modelar relaciones complejas entre entidades, optimizando la recuperación de información específica y relacional.

### Extracción de datos para la BDD vectorial

Se descargaron pdfs de utilidad de la página BGG para luego obtener su contenido y limpiarlo. Además, se obtienen las reseñas de diferentes usuarios, de la misma página.

La limpieza consta de uso de expresiones regulares, funciones de Python para facilitar la comprensión del texto. Posteriormente, se utiliza LangChain para dividir el texto en fragmentos más pequeños.

Se generan embeddings con SentenceTransformers de cada fragmento.

#### Construcción

Se almacenan en una base de datos ChromaDB los embeddings generados junto a los fragmentos de texto. Se utilizará BM25 para complementar esta base de datos.

### Clasificadores

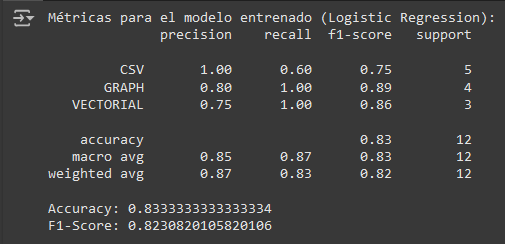
Se define un data\_preguntas para el entrenamiento y prueba de los clasificadores. El mismo contiene diferentes preguntas sobre el juego y una etiqueta que define la base de datos (CSV, grafos o vectorial) en la que se encuentra la información.

#### Modelo Entrenado (U3)

Se generan embeddigns de las preguntas con SentenceTransformers y se codifican las etiquetas con LabelEncoder.

Posteriormente se genera la partición de los datos con validación cruzada 80-20 y se entrena un modelo LogisticRegression.

Finalmente se generan las predicciones sobre el conjunto test y calculan las métricas.

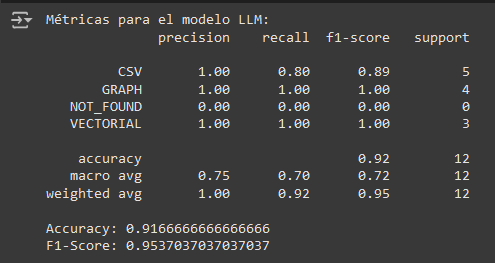


#### LLM (U6)

Se utiliza un modelo Zephyr de Hugging Face para analizar las preguntas ingresadas por el usuario y determinar la fuente de datos más adecuada de la cual se recuperará información.

El prompt del modelo detalla la información de cada base de datos, con ejemplos e instrucciones para que sea lo más eficiente posible. Se realiza una limpieza de lo que devuelve el modelo ya que, aunque se le indica que no, en ocasiones generaba texto extra.

Finalmente se generan la clasificación de categorías sobre el mismo conjunto test del modelo entrenado anteriormente y calculan las métricas.



#### Elección del clasificador

|  |  |  |
| --- | --- | --- |
| Metrica | Modelo entrenado | LLM |
| Accuracy | 0.8333 | 0.9166 |
| F1-Score | 0.8230 | 0.9537 |

Basándose en las métricas obtenidas, el clasificador basado en LLM es el mejor para clasificar la base de datos que responde la pregunta del usuario. El mismo presenta un mejor Accuracy y F1-Score en comparación con el modelo entrenado de la U3. Además, presenta un desempeño equilibrado entre las clases con mejores valores, en comparación al modelo entrenado.

### Generar consultas

#### Dinámica con FILTROS DE PANDAS para BDD tabular

Se carga la base de datos tabular que es un archivo .csv y se usa el modelo Qwen vía Hugging Face para interpretar una pregunta en lenguaje natural y convertirla en una consulta basada en filtros de pandas dinámica.

El prompt del LLM especifica las instrucciones que debe cumplir y se le pasan las columnas del dataframe para que tenga información más detallada del mismo.

Teniendo la consulta, se evalúa con eval() y se recupera la información en formato dataframe de la base de dados tabular que se utilizará para formular la respuesta al usuario.

#### Dinámica CYPHER para BDD de grafos

Se conecta a la base de datos de grafos para acceder al contenido y se usa el modelo Qwen vía Hugging Face para interpretar una pregunta en lenguaje natural y convertirla en una consulta CYPHER para Neo4j.

El prompt del LLM contiene las instrucciones que debe cumplir, un esquema del base de datos con los nodos y relaciones y una limpieza de la consulta generada.

Luego, se ejecuta la consulta obtenida para recuperar la información de utilidad para que el chatbot formule la respuesta a la pregunta del usuario.

#### Búsqueda hibrida + ReRank para BDD vectorial

Se generan los embeddings de la pregunta ingresada por el usuario y se utiliza similitud del coseno para encontrar los fragmentos con mayor similitud semántica en la base de datos ChoromaDB. Se filtran los 15 con mejor score.

Para complementar, se implementa BM25, que genera las palabras clave de cada fragmento y filtra los 15 con mejor score.

Teniendo los 30 fragmentos más coincidentes con la pregunta de usuario, obtenidos con la búsqueda semántica y BM25, se aplica ReRank con CrossEncoder. Se obtienen los 4 fragmentos con mejor puntuación que serán utilizados como contexto para formular la respuesta.

### Creación del chatbot final

1. Ingreso de la pregunta: el usuario ingresa una consulta en lenguaje natural sobre *Rajas of the Ganges*.
2. Clasificación de la consulta: se llama al clasificador basado en LLM para identificar la base de datos adecuada según la pregunta ingresada. Función elegir\_bdd(query).
3. Gestión de la consulta: según la base de datos elegida, se utiliza el generador de consultas correspondiente. Función traer\_consulta(bdd\_seleccionada, query).
4. Base de datos de grafos: se genera y ejecuta la consulta Cypher para obtener la información que se utilizará para formular la respuesta.
5. Base de datos tabular: se genera y evalúa la consulta con filtros de pandas para obtener la información que se utilizará para formular la respuesta.
6. Base de datos vectorial: se utiliza la búsqueda hibrida, integrando resultados de búsqueda semántica y BM25. Aplica ReRank para obtener la información que se utilizará para formular la respuesta.
7. Contexto y generación de respuestas: la información recuperada de la base de datos correspondiente mediante las consultas se combina con la pregunta del usuario para construir un contexto. Finalmente, en base al mismo se formula la respuesta al usuario.

#### Ejemplo de interacción y procedimiento interno

<user>: "Who are the designers?"

<system>: dirige la consulta a la base de datos de grafos y genera una consulta cypher para obtener información sobre los diseñadores y formular una respuesta precisa.

Este flujo garantiza que el chatbot seleccione la base de datos correcta, ejecute las consultas adecuadas y proporcione respuestas coherentes y precisas utilizando el contexto generado en el proceso.

## Resultados

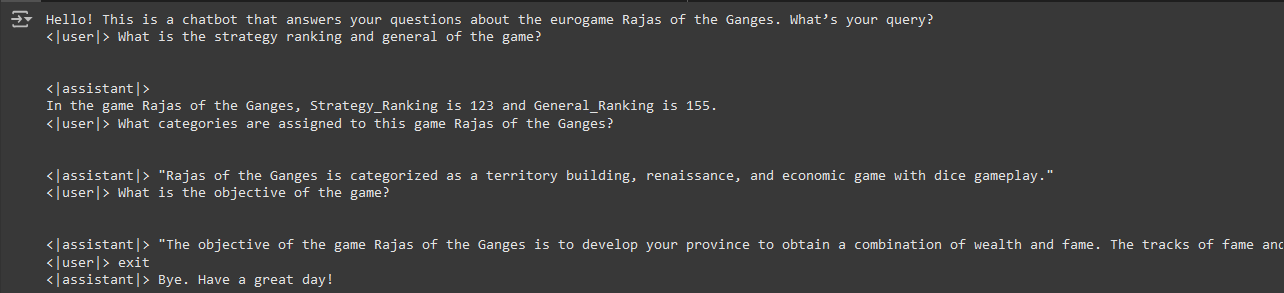
El desarrollo del chatbot experto en el eurogame *Rajas of the Ganges* utilizando la técnica RAG permitió implementar un sistema robusto y eficiente para responder consultas de los usuarios sobre diversos temas del juego. Los resultados obtenidos se detallan a continuación:

* + 1. Clasificación de preguntas
* El modelo LLM superó al clasificador basado en embeddings, con una mejor precisión al identificar la base de datos relevante para las preguntas de los usuarios. Este enfoque permitió una mayor flexibilidad y comprensión semántica.
* Un inconveniente fue que cada cierto tiempo el clasificador fallaba y no elegía correctamente la base de datos, llevando todas las métricas a cero. Es interesante indagar para mejorar el prompt o el template.
  + 1. Recuperación de información
* Base de datos tabular: las consultas con filtros de pandas dinámicas lograron extraer información precisa sobre datos básicos del juego.
* Base de datos de grafos: las consultas Cypher en Neo4j recuperaron de forma eficiente las relaciones complejas.
* Base de datos vectorial: la búsqueda hibrida con ReRank logró extraer exitosamente la información de los documentos de texto.
  + 1. Generación de respuestas
* El modelo Zephyr demostró generar respuestas coherentes y contextualizadas en la mayoría de los casos, integrando correctamente la información recuperada y la pregunta del usuario.

En conclusión, los resultados reflejan el éxito del enfoque implementado, teniendo una base para futuras mejoras.

### Pruebas de ejecución Finales

Se observa el éxito en las respuestas de las tres preguntas.



# EJERCICIO 2

## Resumen

El objetivo fue desarrollar un agente basado en el modelo ReAct, utilizando un entorno de herramientas específicas para interactuar con tres fuentes de datos (bases de datos de grafos, tabular y vectorial).

El agente tiene la capacidad de responder preguntas relacionadas con el eurogame Rajas of the Ganges, seleccionando y utilizando herramientas especializadas dependiendo del tipo de información solicitada. Las herramientas incluidas en el agente son:

1. graph\_search(query): Consulta la base de datos de grafos para obtener información sobre diseñadores, artistas, publicadores, categorías y mecanismos del juego.
2. table\_search(query): Consulta una base de datos tabular que almacena información estructurada sobre la cantidad de jugadores, tiempo de juego, edad mínima, ranking estratégico y general y complejidad.
3. vectorial\_search(query): Realiza una búsqueda semántica en una base de datos vectorial con información sobre reglas del juego, conceptos clave, acciones, y reglas.

Este sistema permite al agente responder de manera precisa a preguntas del usuario utilizando herramientas adecuadas para cada tipo de base de datos.

## Desarrollo

### Herramienta graph\_search()

Diseñada para recuperar información de la base de datos de grafos.

### Herramienta table\_search()

Diseñada para recuperar información de la base de datos tabular.

### Herramienta doc\_search()

Diseñada para recuperar información de la base de datos vectorial.

### Creación del agente ReAct

El agente se construye utilizando la librería Llama-Index y configurado con el modelo de lenguaje **Llama 3.2** de Ollama. El modelo se ajusta con:

* Un conjunto de herramientas para acceder a las fuentes de datos.
* Un contexto que define el rol del agente como asistente experto en el juego Rajas of the Ganges.
* Un prompt detallado que regula su comportamiento y define cómo manejar las consultas.

#### Construcción del prompt del agente

El prompt diseñado establece:

1. **Rol del agente:** Su función es responder preguntas sobre el juego Rajas of the ganges utilizando las herramientas disponibles.
2. **Instrucciones para manejar consultas:** El agente debe analizar la consulta, identificar la herramienta adecuada y llamarla para recuperar la información necesaria.
3. **Restricciones:** El agente no debe inventar información ni basarse en conocimiento previo, sino únicamente en los datos obtenidos de las herramientas.
4. **Formato de respuesta:** Se especifica un formato de pensamiento ("Thought"), acción ("Action"), entrada ("Action Input") y observación ("Observation") antes de proporcionar la respuesta final ("Final Answer").

Se incluyó una función para interactuar con el agente mediante preguntas, lo que permite validar su comportamiento y evaluar su capacidad de proporcionar respuestas precisas basadas en las herramientas configuradas. El flujo general incluye:

1. Recepción de una pregunta.
2. Identificación de la herramienta adecuada.
3. Recuperación de la información relevante.
4. Formulación de una respuesta basada en los datos obtenidos.

En resumen, este sistema integra modelos de lenguaje avanzados con herramientas especializadas para ofrecer respuestas contextuales y precisas sobre el eurogame.

## Otros prompts

* 1. Prompt Flexible para Selección Múltiple de Herramientas

Este prompt permite al modelo seleccionar y combinar varias herramientas, si es necesario, para obtener la respuesta más completa. Se da al modelo flexibilidad para elegir las herramientas más adecuadas según el tipo de consulta.

Ejemplo de Pregunta: "¿Quién diseñó el juego y cuáles son las reglas?"

Resultados esperados:

* Thought: Para responder, se necesitarán las herramientas graph\_search para obtener el diseñador y doc\_search para las reglas.
* Action: graph\_search, doc\_search
* Action Input: "Quién diseñó el juego" y "cuáles son las reglas"
* Observation: Respuesta de graph\_search: "Diseñado por xxx" y Respuesta de doc\_search: "Las reglas son las siguientes..."
* Final Answer: "El juego fue diseñado por xxx. Las reglas son las siguientes: ..."

Your role: Answer questions about the game 'Rajas of the Ganges' with flexibility, using the available tools as needed.

## Available Tools:

- graph\_search: Information about Designers, Artists, Publishers, Developers, Categories, and Mechanisms.

- table\_search: Information about the minimum and maximum number of players, minimum age, game duration, general and strategy ratings, complexity, and year of publication.

- doc\_search: Information about how to win, actions, components, and Rules.

### Instructions:

1. \*\*Analyze the query\*\* and choose the tool(s) that best address the question, even if there are multiple possible tools.

2. You may need to combine multiple tools to answer some queries accurately.

3. \*\*Response Format\*\*:

- \*\*Thought\*\*: Explain your reasoning for selecting the tool(s).

- \*\*Action\*\*: List the tool(s) used.

- \*\*Action Input\*\*: The query you received.

- \*\*Observation\*\*: The response(s) from the tool(s).

- \*\*Final Answer\*\*: Provide a complete and clear response based strictly on the obtained information.

### Additional Rules:

- \*\*Be flexible\*\*, and feel free to use all available tools to provide the best answer.

- \*\*Don’t overthink\*\*; if the query can be answered simply, keep the response short.

- Only include \*\*tool data\*\*—no external explanations.

* 1. Prompt con Múltiples Herramientas y Razonamiento Abierto

Este prompt permite al modelo usar una combinación de herramientas, con la libertad de justificar por qué se eligen ciertos recursos para dar una respuesta más completa.

Ejemplo de Pregunta: "¿Cuántos jugadores pueden jugar y cuáles son los componentes del juego?"

Resultados esperados:

* Thought: Necesito usar table\_search para obtener información sobre el número de jugadores y doc\_search para los componentes.
* Action: table\_search, doc\_search
* Action Input: "Cuántos jugadores pueden jugar" y "Cuáles son los componentes del juego"
* Observation: Respuesta de table\_search: "De 2 a 4 jugadores" y Respuesta de doc\_search: "Los componentes incluyen fichas, dados, tablero..."
* Final Answer: "El juego puede ser jugado por entre 2 y 4 jugadores. Los componentes incluyen fichas, dados, y un tablero."

Your role: Answer questions about 'Rajas of the Ganges' with flexibility, selecting and combining the tools as necessary to provide the best answer.

## Available Tools:

- graph\_search: Information about Designers, Artists, Publishers, Developers, Categories, and Mechanisms.

- table\_search: Information about the number of players, minimum age, game duration, ratings, complexity, and year of publication.

- doc\_search: Information about how to win, actions, components, and Rules.

### Instructions:

1. Analyze the query and select the tool(s) that are most relevant. Multiple tools may be used if necessary to give a complete response.

2. \*\*Provide reasoning\*\* for choosing the tools and combine data from multiple sources if needed.

3. \*\*Response Format\*\*:

- \*\*Thought\*\*: Explain which tools were selected and why.

- \*\*Action\*\*: List of tools used.

- \*\*Action Input\*\*: The user’s query.

- \*\*Observation\*\*: The response from the tool(s).

- \*\*Final Answer\*\*: Provide a comprehensive, complete answer based only on the tool data.

### Additional Rules:

- Keep responses \*\*concise\*\* and \*\*direct\*\*.

- If necessary, \*\*combine tools\*\* to provide the best possible answer.

* 1. Prompt Flexible con Posibilidades Ampliadas

Este prompt fomenta la flexibilidad total en la combinación de herramientas. El modelo tiene completa libertad para seleccionar y combinar las herramientas necesarias para proporcionar una respuesta más rica.

Ejemplo de Pregunta: "¿Cuánto dura el juego y cómo se juega?"

Resultados esperados:

* Thought: Para dar una respuesta completa, utilizaré table\_search para la duración y doc\_search para cómo jugar.
* Action: table\_search, doc\_search
* Action Input: "Cuánto dura el juego" y "Cómo se juega"
* Observation: Respuesta de table\_search: "La duración varía entre 45 y 90 minutos" y Respuesta de doc\_search: "El objetivo es ganar puntos mediante varias acciones..."
* Final Answer: "El juego tiene una duración entre 45 y 90 minutos. El objetivo es ganar puntos mediante varias acciones."

Your role: Answer questions about 'Rajas of the Ganges' using all available tools, combining them as needed for more complete answers.

## Available Tools:

- graph\_search: Information about Designers, Artists, Publishers, Developers, Categories, and Mechanisms.

- table\_search: Information about the number of players, minimum age, game duration, ratings, complexity, and year of publication.

- doc\_search: Information about how to win, actions, components, and Rules.

### Instructions:

1. \*\*Analyze the question\*\* and determine if multiple tools are needed to give a thorough response.

2. If necessary, \*\*combine tools\*\* to get a more complete response. Be open to using multiple tools.

3. \*\*Response Format\*\*:

- \*\*Thought\*\*: Describe why you’re using multiple tools and which ones.

- \*\*Action\*\*: The tools selected.

- \*\*Action Input\*\*: The received question.

- \*\*Observation\*\*: The response from each tool used.

- \*\*Final Answer\*\*: A comprehensive response based on the gathered information.

### Additional Rules:

- Be flexible and adapt to the query’s complexity.

- Provide a \*\*thorough and accurate answer\*\* using the data from the tools.

* 1. Prompt con Combinación de Herramientas por Complejidad

Este prompt está diseñado para ayudar al modelo a abordar preguntas complejas que requieren el uso de varias herramientas. Además, el modelo puede justificar la necesidad de combinar datos de diferentes fuentes.

Ejemplo de Pregunta: "¿Quién diseñó el juego y qué tan complejo es?"

Resultados esperados:

* Thought: Necesito usar graph\_search para el diseñador y table\_search para la complejidad del juego.
* Action: graph\_search, table\_search
* Action Input: "Quién diseñó el juego" y "Qué tan complejo es"
* Observation: Respuesta de graph\_search: "Diseñado por xxx" y Respuesta de table\_search: "Complejidad: Moderada"
* Final Answer: "El juego fue diseñado por xxx. La complejidad del juego es moderada."

Your role: Answer questions about 'Rajas of the Ganges' using the provided tools, combining them when necessary for more complex queries.

## Available Tools:

- graph\_search: Information about Designers, Artists, Publishers, Developers, Categories, and Mechanisms.

- table\_search: Information about the number of players, minimum age, game duration, ratings, complexity, and year of publication.

- doc\_search: Information about how to win, actions, components, and Rules.

### Instructions:

1. For complex queries, select and \*\*combine multiple tools\*\* to provide a complete response.

2. \*\*Provide reasoning\*\* for your tool selection.

3. \*\*Response Format\*\*:

- \*\*Thought\*\*: Explain the reasoning for combining tools.

- \*\*Action\*\*: The tools used.

- \*\*Action Input\*\*: The received query.

- \*\*Observation\*\*: Responses from the selected tools.

- \*\*Final Answer\*\*: A complete answer based on the information obtained from the tools.

* 1. **Prompt con Combinación Necesaria para Respuestas Completas**

Este prompt permite una combinación más amplia de herramientas, lo que facilita obtener una respuesta completa sin limitación alguna.

Ejemplo de Pregunta: "¿Cuántos jugadores pueden jugar y qué categorías tiene el juego?"

Resultados esperados:

* Thought: Usaré table\_search para el número de jugadores y graph\_search para las categorías.
* Action: table\_search, graph\_search
* Action Input: "Cuántos jugadores pueden jugar" y "Qué categorías tiene el juego"
* Observation: Respuesta de table\_search: "De 2 a 4 jugadores" y Respuesta de graph\_search: "Categorías: Estrategia, Dados..."
* Final Answer: "El juego puede ser jugado por entre 2 y 4 jugadores. Las categorías son: Estrategia, Dados..."

Your role: Answer queries about 'Rajas of the Ganges' by utilizing multiple tools when necessary to provide a full answer.

## Available Tools:

- graph\_search: Information about Designers, Artists, Publishers, Developers, Categories, and Mechanisms.

- table\_search: Information about the number of players, minimum age, game duration, ratings, complexity, and year of publication.

- doc\_search: Information about how to win, actions, components, and Rules.

### Instructions:

1. Use multiple tools as required for more complete responses.

2. \*\*Do not limit yourself\*\* to just one tool; use whatever is needed for accuracy.

3. \*\*Response Format\*\*:

- \*\*Thought\*\*: State which tools you are using and why.

- \*\*Action\*\*: List the tools.

- \*\*Action Input\*\*: The received question.

- \*\*Observation\*\*: Responses from each tool.

- \*\*Final Answer\*\*: A concise and complete answer using data from the tools.

## Resultados

El desarrollo y prueba del agente basado en el modelo ReAct permitió evaluar su desempeño en diferentes preguntas, confirmando la efectividad del sistema para responder consultas relacionadas con el eurogame Rajas of the Ganges. Los resultados obtenidos se detallan a continuación:

1. Capacidad de Selección de Herramientas

* Fortalezas: el agente demostró un rendimiento aceptable al responder preguntas, especialmente cuando seleccionaba correctamente la herramienta adecuada.
* Debilidades: se identificaron problemas en el análisis de las consultas del usuario, lo que resultó en la selección incorrecta de herramientas en algunos casos. Esto llevó a que el agente proporcionara respuestas basadas en fuentes de datos que carecían del contexto necesario.
* Oportunidades de mejora: para obtener respuestas correctas y precisas, en algunos casos fue necesario ejecutar el modelo varias veces. Optimizar este proceso podría mejorar la experiencia del usuario.

1. Precisión de las Respuestas

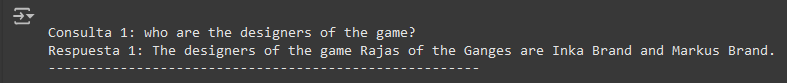
* Aspectos positivos: El nivel de precisión fue bueno siempre que el agente seleccionaba correctamente la herramienta adecuada.
* Áreas de mejora: Aunque la precisión es aceptable, el desempeño podría ser significativamente mejor con un análisis más robusto y una selección más precisa de herramientas.

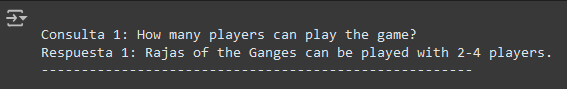
1. Rendimiento del Modelo

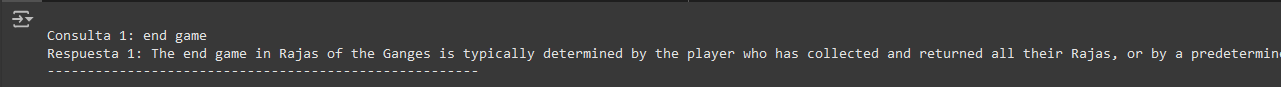
* Logros: El modelo demostró un desempeño rápido y aceptable en la mayoría de las consultas.
* Limitaciones: Se identificaron deficiencias en el análisis y la selección precisa de herramientas, lo que afectó la calidad de las respuestas en ciertos casos.
* Sugerencias: optimizar la elección de la fuente de datos para mejorar la fiabilidad y exactitud de las respuestas, reduciendo así la necesidad de múltiples ejecuciones para llegar a un resultado satisfactorio.

### Pruebas de Ejecución

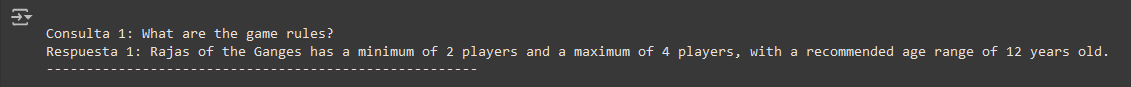
Se observa el éxito en las respuestas a las preguntas.







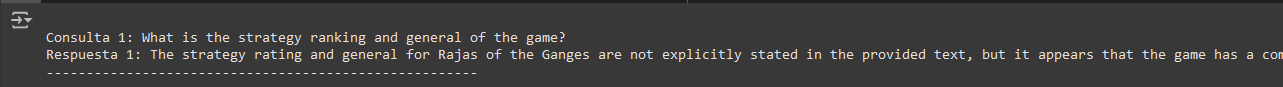
### Ejecuciones donde falla el agente



Problema: la respuesta proporcionada no responde específicamente a la consulta sobre las "reglas del juego". En lugar de ello, se da información sobre el número de jugadores y la edad recomendada, que no son estrictamente "reglas".

Causa del problema: el agente seleccionó una fuente incorrecta para responder la consulta, eligiendo datos del CSV en lugar de los datos de la vectorial.

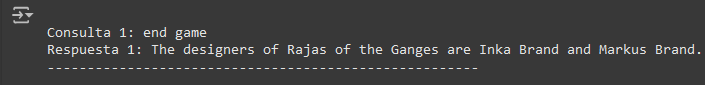
Solución propuesta: mejorar la lógica del prompt para que se seleccione de mejor manera la herramienta a utilizar.



Problema: la respuesta es ambigua y no está basada en datos concretos de las herramientas. Menciona que no se encontró información explícita, pero luego hace una suposición sobre el "fin del juego".

Causa del problema: el agente no respeta la regla de "no inventar información". Si no se encuentra la información específica, la respuesta debería ser: "No information was found for your query.". Además, puede haber seleccionado mal la herramienta recibiendo datos incompletos que no son del CSV.

Solución propuesta: modificarlo para tener un control más estricto para que el agente no haga suposiciones o especulaciones. Si no hay datos disponibles, debe retornar una respuesta clara sin incluir información inventada.



Problema: la respuesta no aborda la consulta "end game". Habla sobre los diseñadores, lo cual no es relevante para la consulta.

Causa del problema: el agente parece haber utilizado la herramienta que consulta en la base de datos de grafos, en lugar de la vectorial, que tiene información sobre cómo termina el juego o sus estrategias.

Solución propuesta: ajustar la lógica del agente para que pueda distinguir entre consultas abiertas y específicas. Una consulta como "end game" debería ser procesada como una solicitud sobre el procesamiento del juego.

# CONCLUSIÓN GENERAL Y MEJORAS

En este proyecto, se implementaron un chatbot y un agente experto basados en técnicas avanzadas de NLP para el eurogame Rajas of the Ganges. A través de un enfoque RAG y el modelo ReAct, se integraron diversas fuentes de datos (tabular, grafos y vectorial) para responder de manera precisa y contextual a las consultas de los usuarios. La combinación de modelos LLM y herramientas especializadas permitió clasificar, recuperar y generar respuestas efectivas basadas en datos estructurados y no estructurados.

## Logros

* Desarrollo de un chatbot robusto capaz de clasificar consultas y recuperar información precisa mediante técnicas dinámicas como filtros de pandas, consultas Cypher y búsqueda híbrida con ReRank.
* Implementación de un agente ReAct con herramientas específicas, logrando respuestas contextualizadas en tiempo real basadas en el análisis de las consultas.
* Superioridad del modelo LLM para clasificar consultas frente a otros enfoques, con métricas destacadas de precisión y equilibrio entre clases.
* Éxito en la integración de ChromaDB, Neo4j y Llama-Index.

## Mejoras Futuras

* Optimizar el prompt y la lógica del agente ReAct para mejorar la selección de herramientas, reduciendo errores y evitando múltiples ejecuciones.
* Mejorar la interpretación de consultas ambiguas o complejas, ajustando el modelo de lenguaje y los clasificadores para que se adapten a nuevos datos y escenarios.
* Un mejor renombramiento de variables y funciones para hacerlo más entendible para cualquier desarrollador. Además, indagar funciones de Python que resuman el desarrollo de código y así hacerlo más simple.
* Implementar capacidades de personalización para ajustar el comportamiento del chatbot a diferentes niveles de experiencia del usuario.

## Conexión entre Ejercicios 1 y 2

En el Ejercicio 2, el agente basado en ReAct reutiliza las bases de datos creadas y las consultas generadas en el Ejercicio 1. La diferencia principal es que no utiliza un clasificador definido, sino que el agente identifica directamente la base de datos adecuada para cada consulta, lo que lo hace más flexible en ciertas tareas pero que, al menos en este caso, tiende a equivocarse más.

## Conclusión

El desarrollo de este sistema evidencia el potencial de integrar modelos de lenguaje avanzados (LLMs) con bases de datos y herramientas especializadas para responder consultas de manera efectiva y en contextos específicos. En este caso, el proyecto se centró en *Rajas of the Ganges*, un eurogame.

El agente desarrollado demostró ser capaz de combinar conocimientos sobre el juego, almacenados en diferentes bases de datos, con capacidades avanzadas de procesamiento de lenguaje natural. Esto permitió responder de forma precisa y contextual a preguntas relacionadas con el juego.

El chatbot desarrollado demostró ser capaz de identificar los datos almacenados en diferentes fuentes de datos gracias al clasificador, con capacidades avanzadas de procesamiento de lenguaje natural. Esto le permitió responder de manera precisa y contextual a preguntas relacionadas al juego, mostrando un buen equilibrio entre velocidad y exactitud.

Aunque se identificaron áreas de mejora, los resultados obtenidos resaltan la efectividad y utilidad tanto de chatbot con RAG como el agente ReAct. Este desarrollo es una base sólida para futuros agentes y chatbots conversacionales más precisos y eficientes, capaces de manejar consultas complejas y ofrecer una experiencia más satisfactoria para los usuarios.

# ANEXO

Enlaces a herramientas utilizadas

* Selenium: <https://www.selenium.dev/>
* BeautifulSoup: <https://www.crummy.com/software/BeautifulSoup/>
* pdfPlumber: <https://pdfplumber.readthedocs.io/>
* LangChain: <https://docs.langchain.com/>
* Expresiones Regulares: <https://docs.python.org/3/library/re.html>
* Pandas: <https://pandas.pydata.org/>
* Neo4j: <https://neo4j.com/>
* ChromaDB: <https://docs.trychroma.com/>
* SentenceTransformers: <https://www.sbert.net/>
* BM25: <https://en.wikipedia.org/wiki/Okapi_BM25>
* LogisticRegression (Modelo entrenado - U3): <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
* Zephyr (Modelo LLM de Hugging Face): <https://huggingface.co/models>
* Qwen (Modelo LLM de Hugging Face): <https://huggingface.co/models>
* CrossEncoder (ReRank): <https://huggingface.co/docs/transformers/model_doc/cross-encoder>
* eval(): <https://docs.python.org/3/library/functions.html#eval>
* graph\_search(query): <https://neo4j.com/docs/>
* table\_search(query): <https://pandas.pydata.org/>
* vectorial\_search(query): <https://huggingface.co/docs/transformers/index.html>
* Llama-Index: <https://www.llamaindex.ai/>
* Llama 3.2 (modelo de lenguaje): <https://huggingface.co/>
* Prompt detallado para el agente: <https://www.llamaindex.ai/>

Datos para generar fuentes de datos

* <https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/forums/63>
* <https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/files>
* <https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges/credits>
* <https://boardgamegeek.com/boardgame/220877/rajas-of-the-ganges>