TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL

Procesamiento del Lenguaje Natural

INFORME TP N°2

Estudiantes:

Valeri, Lara Rita

Docentes:

Juan Pablo Manson

Alan Geary

Juego asignado:

Lost Ruins of Arnak

Repositorio:

https://github.com/LaraV15/NLP

Índice

Introducción

El Trabajo Práctico Final consiste en desarrollar un sistema inteligente dividido en dos etapas complementarias. En la primera, se implementa un chatbot experto en un Eurogame, este caso Lost Ruins of Arnack, utilizando la técnica Retrieval Augmented Generation (RAG). Este chatbot integra múltiples fuentes de conocimiento, como documentos de texto, datos tabulares y bases de datos de grafos, para responder preguntas en español o inglés de manera precisa y contextual.

En la segunda etapa, se amplía la funcionalidad mediante un agente basado en ReAct, capaz de combinar información de diferentes fuentes para responder consultas complejas. El proyecto busca aplicar técnicas avanzadas de procesamiento de lenguaje natural, implementar búsquedas dinámicas y eficientes, y evaluar la calidad del sistema con ejemplos prácticos y casos de uso. Todo el desarrollo se realiza en Google Colab.

En el presente informe se describen las resoluciones de ambos ejercicios, así como también errores y limitaciones. Complementando con ejemplos e imágenes de los resultados obtenidos.

En el anexo del presente informe se encuentran ejemplos para poder probar el asistente en caso de que sea necesario.

Metodología

Fuentes de datos utilizadas

Para realizar la base de datos se utilizaron archivos pdf, archivos, txt e información de la siguiente sitios web sobre el juego:

- bgg: https://boardgamegeek.com/boardgame/312484/lost-ruins-of-arnak
- Reseña: https://misutmeeple.com/2021/04/resena-las-ruinas-perdidas-de-arnak/
- WikiData: https://www.wikidata.org/wiki/Q107835072
- Don Meeple: https://donmeeple.com/ruinas-perdidas-arnak-juego-mesa/

Herramientas y tecnologías

Software:

- Google Colab como entorno de desarrollo.
- LangChain para la división de textos.
- ChromaDB para la base de datos vectorial.
- RedisGraph para grafos.
- Llama-Index para la construcción del agente ReAct.
- Google Translate para realizar traducciones.
- Google Drive para guardar los archivos de texto para ser utilizados en l base de datos.

Modelos:

- Modelos de embeddings y LLM seleccionados según rendimiento y compatibilidad.

Hardware:

- Infraestructura proporcionada por Google Colab Pro para entrenamiento y pruebas.

Pasos seguidos para cumplir los objetivos

- Análisis y recolección de datos relevantes desde las fuentes mencionadas.
- Limpieza y preprocesamiento de los datos para garantizar su calidad.
- Implementación de técnicas de vectorización y creación de bases de datos.
- Desarrollo de clasificadores y sistemas de recuperación.
- Construcción e integración del agente basado en ReAct.
- Pruebas y evaluación de los resultados para identificar áreas de mejora.

Juego

El juego en el que se basa este trabajo es **‘Lost Ruins of Arnak’,** Las Ruinas Pérdidas de Arnak en español.

Es un juego de mesa estilo Eurogame que combina exploración, gestión de recursos y construcción de mazos. Ambientado en una isla misteriosa, los jugadores asumen el rol de

exploradores que buscan descubrir sus secretos antiguos mientras compiten por puntos de victoria. A lo largo del juego, deben recolectar artefactos, investigar templos y enfrentarse a guardianes mientras optimizan sus recursos y estrategias. Con una mezcla única de mecánicas, este juego ofrece una experiencia desafiante y temática para los amantes de la estrategia.

Ejercicio 1

El presente ejercicio tiene como objetivo implementar un chatbot especializado en el juego de mesa utilizando la técnica Retrieval Augmented Generation (RAG). El sistema integra tres fuentes principales de conocimiento:

1. **Documentos de texto**
2. **Datos tabulares**
3. **Base de datos de grafos**

Se espera que el chatbot sea capaz de clasificar preguntas y recuperar información relevante de las fuentes adecuadas, generando respuestas en español o inglés según el idioma de la consulta. Los documentos son procesados con técnicas de *chunking* y embeddings almacenados en ChromaDB. Se desarrollarán dos clasificadores: uno basado en LLM y otro con Random Forest, comparando sus resultados.

Base de datos

La información para la base de datos se recopiló principalmente a partir de los enlaces proporcionados por la cátedra. En particular, la sección de "Files"[1] de uno de estos enlaces resultó especialmente útil, ya que contenía diversos archivos relevantes para esta parte del ejercicio.

Una decisión clave en este proceso fue determinar el idioma de la base de datos. Opté por construirla en inglés, dado que este idioma ofrece una mayor compatibilidad y rendimiento con las herramientas y modelos de procesamiento de lenguaje natural disponibles, además de contar con una amplia variedad de recursos y datos que simplifican su integración.

Base de datos vectorial

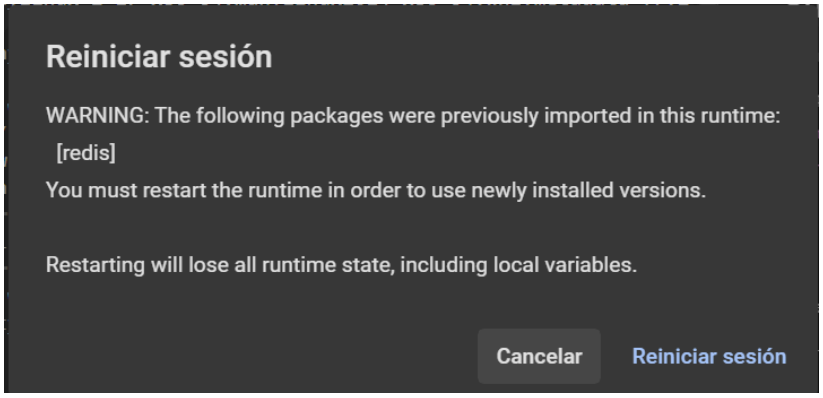
En primer lugar se crean las listas: *context = []* y *metadata = []*

La lista *context* almacena los textos procesados, como documentos, reseñas o cualquier contenido relevante extraído para el análisis, mientras que la lista metadata guarda información asociada a cada elemento en context, como etiquetas, fuentes o clasificaciones que describen el tipo de contenido o su procedencia.

Con respecto al link que contenía la reseña, se descarga la página web específica utilizando requests, se analiza su contenido HTML con BeautifulSoup y se extrae el texto de la reseña principal contenida en un elemento div con la clase entry-content. Finalmente, se tradujo el texto ya que el mismo se encontraba en español, utilizando la libería Google Translate.

[1] https://boardgamegeek.com/boardgame/312484/lost-ruins-of-arnak

Esto último trajo varios problemas por incompatibilidad de librerías, en donde nos saltaba el siguiente pop up:



Dicho problema se soluciono acomodando las librerías, pero derivó en otro problema: *'Warning: Can only detect less than 5000 characters'.* Para resolver este problema cree una función que traduce de a 4500 carácteres, este método se utilizó posteriormente para algunos pdfs que también estaban en español.

Para extraer datos de la página web de bgg se utiliza Selenium en lugar de Beautiful Soup porque el sitio incluye contenido dinámico que se carga mediante JavaScript. A diferencia de Beautiful Soup, que analiza el HTML estático directamente, Selenium permite interactuar con un navegador real, lo que garantiza que se cargue y procese todo el contenido generado dinámicamente. Esta capacidad fue fundamental para acceder a la información relevante de la página, especialmente en casos donde era necesario esperar la carga completa de elementos específicos.

La parte del Web Scraping tuvo sus limitaciones por time out y por incompatibilidad de librerías, más allá de eso se pudo obtener información importante para agregar a la base de datos tabular y vectorial. En el caso de la base de datos vectorial se agregó la descripción del juego.

Del último link mencionado, se obtuvieron algunos pdf que también fueron agregados a la base de datos.

Luego de tener toda la información de documentos y del Web Scrapping, se analizó la cantidad aproximada de páginas de toda la base de datos, ya que la misma tenía que estar cerca de 100, obtuvimos lo siguiente:

```
Total de caracteres: 165038
Páginas estimadas: 91.69
```

Luego de generar la base de datos de documentos, el conjunto de textos se dividió en fragmentos más pequeños utilizando el método *RecursiveCharacterTextSplitter*, asegurando que cada fragmento tenga un tamaño máximo definido *(chunk_size=1000)* y una superposición entre fragmentos *(chunk_overlap=100).* Luego, cada fragmento se almacena

en una lista, mientras que se registra información adicional (metadatos) en otra lista, asociando a cada fragmento un identificador único en la lista ids.

A partir de esto se realiza una base de datos vectorial usando ChromaDB. Luego, se utiliza el modelo de embeddings *SentenceTransformer ('all-MiniLM-L6-v2')* para generar representaciones vectoriales (embeddings) de los fragmentos de texto previamente divididos.

Los embeddings, junto con los textos, metadatos e identificadores, se almacenan en la base de datos. Esto permite indexar y recuperar los textos en función de su significado semántico, facilitando las futuras tareas de búsquedas contextuales y generación de respuestas relevantes de nuestro sistema.

Base de datos tabular

Realizando Web Scrapping con Selenium sobre las estadísticas del juego y se realiza una tabla que contiene los siguientes datos:

Calificación y popularidad:

- Avg. Rating (8.071): Calificación promedio del juego.
- No. of Ratings (47,364): Cantidad de calificaciones recibidas.
- Page Views (3,626,686): Veces que la página del juego fue vista.
- Fans (4,338): Usuarios que lo marcaron como favorito.

Dificultad y opiniones:

- Weight (2.92/5): Complejidad percibida del juego.
- Std. Deviation (1.25): Dispersión de las calificaciones.
- Comments (5,892): Comentarios de usuarios.

Clasificación:

- Overall Rank (29): Posición general en el ranking.
- Strategy Rank (29): Posición en juegos estratégicos.
- Estadísticas de partidas:
- All Time Plays (245,388): Partidas registradas históricamente.
- This Month (2,823): Partidas registradas este mes.

Colección e intercambios:

- Own (71,720): Usuarios que lo poseen.
- Wishlist (13,802): En la lista de deseos de usuarios.
- For Trade (452) / Want In Trade (1,062): Disponibilidad e interés en intercambiarlo.

Partes del juego:

- Has Parts (26) / Want Parts (16): Intercambios de partes específicas.

Base de datos de grafos

Se construyen dos bases de datos de grafos.

Base de datos de grafos con WikiData

Se realiza un grafo que contiene la relación entre el juego y otros que han ganado los mismos premios o que comparten algún editor. A continuación se explica brevemente cada uno de los métodos utilizados.

1) get_wikidata_info_all(entity_id):

Consulta toda la información asociada al entity_id del juego en Wikidata y la presenta en un formato legible. Utiliza SPARQL para obtener propiedades y valores relacionados.

Muestra datos relevantes del juego, como fecha de publicación, editor, creador, etc.

2) get_wikidata_info(entity_id):

Filtra información específica del entity_id (como "instance of", "publisher", "developer", y "publication date") y la organiza en un diccionario.

Devuelve un diccionario con claves descriptivas (como "publisher") y sus valores correspondientes.

3) get_same_publisher():

Encuentra juegos publicados por el mismo editor que Lost Ruins of Arnak. Utiliza SPARQL para buscar relaciones con la propiedad P123 (publisher).

Devuelve una lista de juegos que comparten el mismo editor.

4) get_same_awards():

Encuentra juegos que comparten premios con el juego consultado. Utiliza SPARQL para buscar relaciones con la propiedad P1411 (premios).

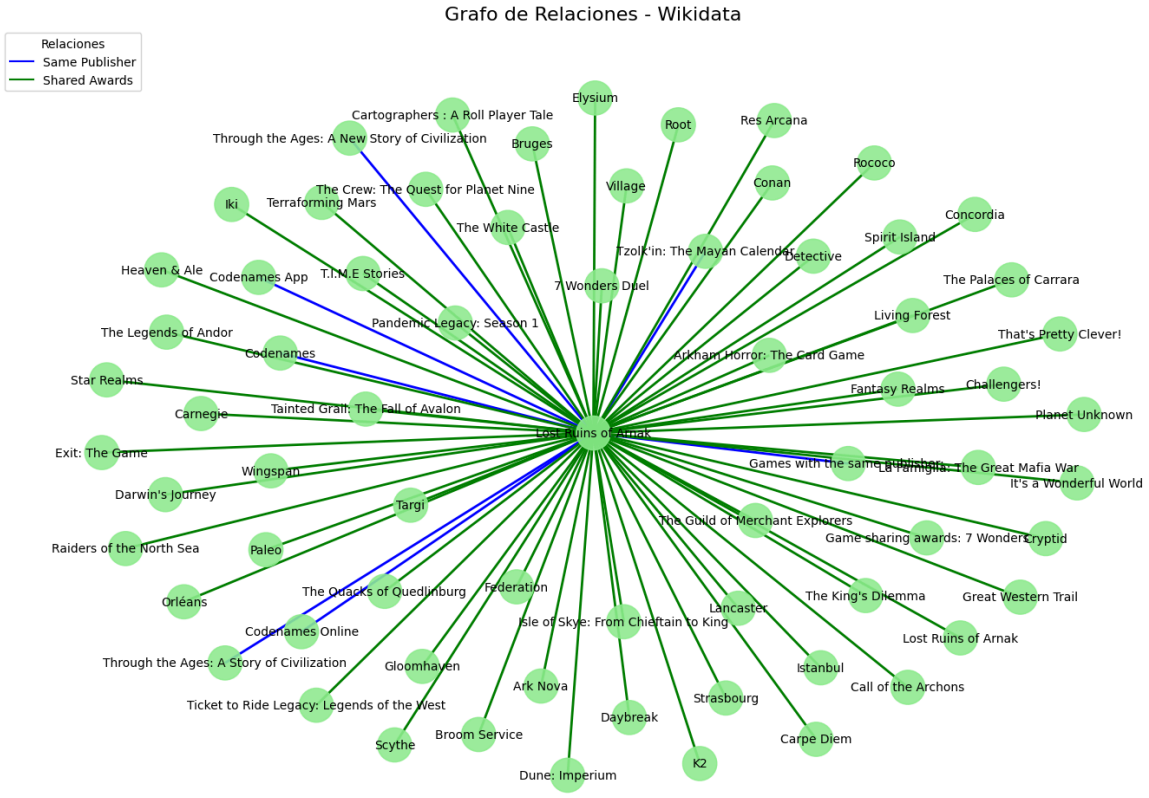
Devuelve una lista de juegos que comparten uno o más premios con el juego principal.

5) Construcción del grafo (graph_new):

Crea un nodo principal que representa al juego consultado. Añade nodos para juegos relacionados por editor ("Same Publisher") y premios compartidos ("Shared Awards").

Conecta los nodos con aristas etiquetadas según la relación.

Resultado:



Grafo con bgg credits

Para obtener datos relevantes sobre el juego, se optó nuevamente por el uso de Selenium por los mismo motivos ya mencionados anteriormente.

Se decidió utilizar RedisGraph como base de datos para modelar relaciones entre los elementos extraídos. RedisGraph fue elegido por su eficiencia en consultas dinámicas y su integración con sistemas existentes de Redis, además de su capacidad para manejar datos semiestructurados como nodos y relaciones.

A continuación se detallan los métodos utilizados:

1) extract_credits():

Utiliza Selenium para acceder a la página de créditos del juego en BoardGameGeek. Extrae dinámicamente información sobre roles y personas asociadas utilizando selectores CSS con BeautifulSoup. Genera un diccionario con los créditos del juego. Ejemplo:

```
{
   "Designer": ["Min & Elwen"],
   "Illustrator": ["David Cochard"],
   "Publisher": ["Czech Games Edition"]
}
```

2) Construcción de la Base de Datos de Grafos

- add_node_and_connect(role, person, i):

Crea nodos en RedisGraph para cada persona asociada a un rol. Conecta los nodos al nodo central que representa al juego mediante relaciones etiquetadas.

Resultado: Nodos y relaciones dinámicamente creados en RedisGraph. Ejemplo:

Game "Lost Ruins of Arnak" has_been_designed_by Min & Elwen

- create_graph_from_credits():

Procesa el diccionario de créditos para generar nodos y conexiones en RedisGraph. También conecta entre sí nodos que comparten el mismo rol.

Devuelve un grafo en RedisGraph que modela las relaciones entre el juego, sus colaboradores, y roles compartidos.

3) visualize_graph():

Recupera nodos y relaciones desde RedisGraph y genera un grafo visual utilizando NetworkX. Aplica un diseño basado en spring_layout, asigna colores a las relaciones y crea una leyenda explicativa.

4) query_roles_and_persons():

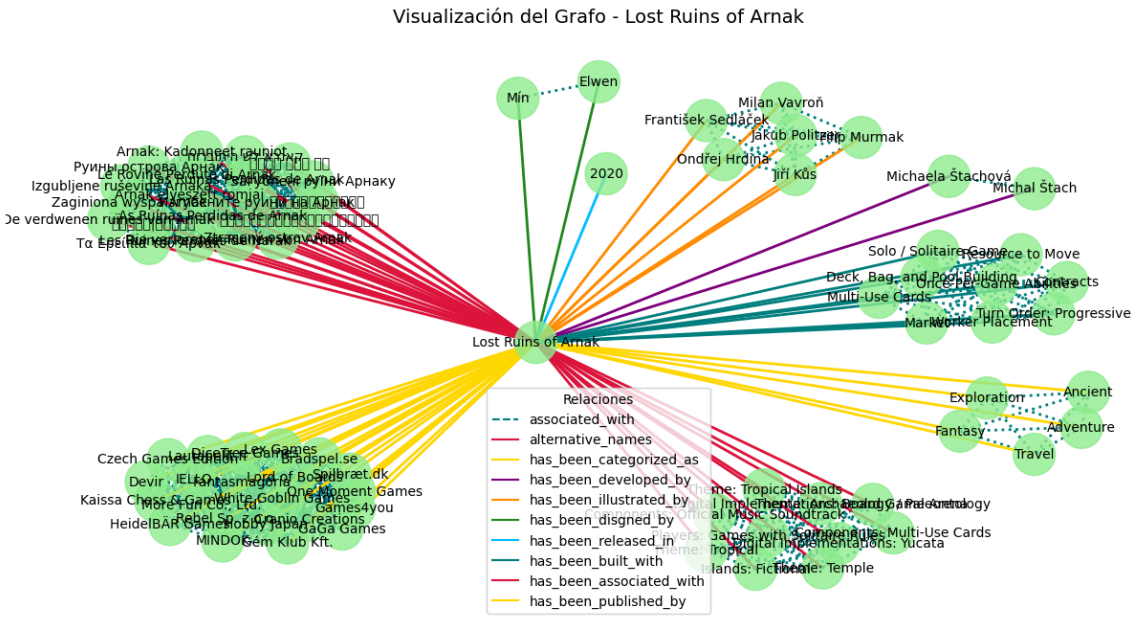
Realiza una consulta en RedisGraph para contar el número de personas asociadas a cada rol. Ordena los resultados por cantidad.

5) Generación de Triadas

Primero se genera una lista de listas con el contenido, por ejemplo los elementos de la lista serán de esta forma: *['Lost Ruins of Arnak', 'has_been_published_by', 'Hobby Japan']*

Luego se crea un único string uniendo todos los elementos de la lista que se guarda en *triadas_string_combined.*

Resultado del grafo obtenido:



Clasificador

En primer lugar se genera un conjunto de datos estructurado para entrenar los modelos de clasificación de preguntas relacionadas con el juego de mesa. Cada entrada contiene un prompt (pregunta) y una categoría asociada.

Se crean preguntas (con ayuda de Chat GPT) categorizadas en cuatro grupos:

- review: Preguntas sobre opiniones, recomendaciones y análisis del juego.
- stats: Preguntas sobre datos numéricos, características técnicas o estadísticas del juego.
- gameplay: Preguntas sobre mecánicas, reglas y estrategias del juego.
- miscellaneous: Preguntas sobre juegos similares a "Lost Ruins of Arnak" y los créditos del juego.

Con un total de 120 preguntas, 30 por cada categoría, estas preguntas cubren diversos aspectos del juego y sirven como ejemplos etiquetados.

Se usa la biblioteca pandas para convertir la lista questions_data en un DataFrame llamado training_data.

Las columnas del DataFrame son:

- prompt: Contiene las preguntas simuladas.

- type: Contiene la categoría correspondiente.

Random Forest

Elegí Random Forest como modelo inicial debido a su simplicidad y capacidad para manejar datos no lineales y desequilibrados. Es eficiente computacionalmente, lo que lo hace ideal para una primera aproximación, y permite interpretar la importancia de las características textuales (como palabras clave) de forma clara.

Para esto se entrena y evalúa un modelo de clasificación que asigna categorías a preguntas relacionadas con el juego. Se utiliza, como se mencio, un modelo de Random Forest para clasificar preguntas relacionadas con el juego en las categorías predefinidas (review, stats, gameplay, micellaneous).

1) train_random_forest()

Entrena un clasificador Random Forest para categorizar preguntas relacionadas con el juego en las categorías mencionadas: review, stats, gameplay, y miscellaneous_info.

Procedimiento:

- Vectorización del texto: Se utiliza TF-IDF para convertir las preguntas en vectores numéricos, eliminando palabras comunes con la opción stop_words='english'.
- División de datos: Los datos se dividen en un 80% para entrenamiento y un 20% para prueba, utilizando train_test_split.
- Se utiliza un modelo Random Forest Classifier con 100 árboles (n_estimators=100) y una semilla aleatoria para reproducibilidad (random_state=42).
- Se evalúa el modelo en el conjunto de prueba utilizando métricas como precisión, recall, y F1-score, generadas por classification_report.

Resultado:

	precision	recall	f1-score	support
gameplay	0.20	0.50	0.29	4
micellaneous_info	0.17	0.25	0.20	4
review	0.00	0.00	0.00	7
stats	0.29	0.22	0.25	9
accuracy			0.21	24
macro avg	0.16	0.24	0.18	24
weighted avg	0.17	0.21	0.17	24

El modelo presenta un desempeño limitado debido, posiblemente, al conjunto de datos reducido y desequilibrado.

Precisión promedio (weighted avg): 17%.

F1-score promedio (weighted avg): 17%.

2) rf_predict(prompt)

Función para realizar predicciones con el modelo entrenado. Recibe un prompt como entrada, lo vectoriza con el modelo TF-IDF, y genera la categoría predicha.

Algunos ejemplos:

```
prompt = "Is the game too expensive?"
print("Prediction:", rf_predict(prompt))
```

```
Prediction: micellaneous_info
```

```
prompt = "What is the average playtime for Lost Ruins of Arnak?"
print("Prediction:", rf_predict(prompt))
```

```
Prediction: stats
```

```
prompt = "What are the pros and cons of Lost Ruins of Arnak?"
print("Prediction:", rf_predict(prompt))
```

```
Prediction: review
```

```
prompt = "What are the most common complaints about Lost Ruins of Arnak?"
print("Prediction:", rf_predict(prompt))
```

```
Prediction: review
```

En estos ejemplos, observamos que el modelo no clasifica correctamente la primera pregunta, "Is the game too expensive?", ya que debería pertenecer a la categoría review pero fue asignada incorrectamente a miscellaneous_info. Sin embargo, las demás preguntas parecen estar clasificadas correctamente, asignándole la categoría esperada, como stats y review. Esto sugiere que el modelo tiene cierto nivel de precisión para algunas categorías, pero podría beneficiarse de un ajuste más fino o de un conjunto de datos más amplio para mejorar su capacidad de generalización, especialmente en casos más ambiguos como el primero.

Antes de realizar ajustes en este modelo, decidí explorar un enfoque comparativo probando un segundo modelo. El objetivo era evaluar el desempeño relativo de ambos clasificadores y seleccionar el más prometedor para optimizarlo. Por esta razón, no se realizaron modificaciones ni ajustes en este modelo inicial, ya que su propósito principal era servir como punto de referencia en el análisis comparativo.

HuggingFace LLM

Utilizando la API de Hugging Face se implementa un clasificador utilizando las preguntas ya definidas. El clasificador llama al modelo Qwen/Qwen2.5-72B-Instruct para obtener una respuesta, que se espera sea una de las cuatro categorías.

llm_classifier(prompt)

Este método utiliza un modelo de lenguaje de Hugging Face para clasificar preguntas relacionadas con el juego "Lost Ruins of Arnak" en las categorías. El modelo procesa cada pregunta y devuelve una categoría basada en el análisis de contexto.

Procemidiento:

- Configuración del Modelo:

Se utiliza el modelo Qwen/Qwen2.5-72B-Instruct a través de la API de Hugging Face.

Se proporciona un token de acceso para autenticar las solicitudes.

- Prompt Engineering:

Inicialmente, se utilizó un prompt básico que instruía al modelo a clasificar preguntas. Este prompt logró un accuracy de 67.5%.

Posteriormente, el prompt fue reforzado, proporcionando descripciones más detalladas de cada categoría y enfatizando que la respuesta debía ser únicamente el nombre de una categoría. Esto mejoró el desempeño a un accuracy de 75.83%.

- Clasificación de Preguntas:

Cada pregunta del conjunto de datos se pasa al modelo mediante el prompt reforzado.

Si la predicción no coincide con una de las categorías predefinidas, se aplica un mecanismo de clasificación aleatoria basado en la longitud del texto para evitar respuestas inválidas.

Resultado

Accuracy inicial: 67.5% (con el prompt básico).

Accuracy final: 75.83% (con el prompt reforzado).

Este modelo superó significativamente al clasificador de Random Forest, mostrando un mejor desempeño en la clasificación de preguntas en todas las categorías. Algunos ejemplos:

```
print(llm_classifier("Who designed Lost Ruins of Arnak?"))
micellaneous_info

print(llm_classifier("What are the most common complaints about Lost Ruins of Arnak?"))
review

print(llm_classifier("What are the pros and cons of Lost Ruins of Arnak?"))
review

print(llm_classifier("What is the average playtime for Lost Ruins of Arnak?"))
stats

print(llm_classifier("Is the game too expensive?"))
review
```

A partir de estos ejemplos podemos concluir que el modelo LLM clasifica correctamente todas las preguntas en estos ejemplos, asignándolas a las categorías adecuadas como micellaneous_info, review, y stats. Esto demuestra un claro entendimiento del contexto, superando las limitaciones del modelo Random Forest. El uso del prompt reforzado contribuye significativamente a este desempeño, mejorando la precisión en categorías como review, donde el modelo previo fallaba.

Retriever

El retriever es una pieza clave que actúa como intermediario para seleccionar y devolver el contexto relevante según el tipo de pregunta realizada por el usuario.

El retriever es utilizado para buscar y recuperar información relevante desde diferentes fuentes de datos (base vectorial, tablas o grafos) con base en el análisis y clasificación de la consulta del usuario.

1) *Clasificación de la consulta:*

Usa el modelo LLM para determinar la categoría (review, gameplay, stats, micellaneous_info) de la consulta realizada.

2) *Recuperación desde distintas fuentes:*

Dependiendo de la clasificación:

review o gameplay: Busca información relevante en una base de datos vectorial mediante embeddings, optimizando los resultados con un modelo Cross-Encoder para reordenar los documentos más relevantes (reranker).

stats: Recupera datos tabulares o estadísticos de un DataFrame.

micellaneous_info: Recupera información estructurada desde un grafo (relaciones entre premios, diseñadores, etc.).

Si la clasificación no es clara, combina información de todas las fuentes.

3) Re-rankeo de resultados:

Cuando la consulta recae en la base vectorial, utiliza un modelo Cross-Encoder (como ms-marco-MiniLM) para reordenar los documentos recuperados, asignando mayor relevancia a los más ajustados al prompt.

4) Entrega del contexto:

Devuelve el contenido más relevante (texto, datos o relaciones) para que sea usado como contexto en la generación de respuestas.

Prompt

El diseño del prompt es crítico para guiar al modelo a responder de manera precisa y contextualizada. Al estructurarlo con instrucciones específicas y un contexto relevante:

- Se mejora la capacidad del modelo para comprender y responder preguntas relacionadas con "Lost Ruins of Arnak".
- Se minimizan respuestas genéricas o irrelevantes al enfocar al modelo en un dominio específico.
- Se optimiza el uso del contexto recuperado, asegurando que las respuestas estén alineadas con las necesidades del usuario.

1) prompt_prep(context, user_input):

Esta función prepara el prompt que será enviado al modelo de lenguaje (LLM). Combina el contexto disponible (información recuperada por el retriever) con la consulta del usuario para construir una instrucción clara y específica.

El prompt detalla el rol del modelo como un asistente especializado en el juego de mesa "Lost Ruins of Arnak".

Se incluyen el contexto y la pregunta del usuario para guiar al modelo en la generación de una respuesta enfocada y precisa.

Establece expectativas para evitar respuestas extensas o irrelevantes.

2) translate_to_english(content):

Traduce el texto ingresado al inglés, asegurando que las consultas o datos en español sean compatibles con el modelo.

3) translate_to_spanish(content):

Traduce las respuestas generadas en inglés al español para que sean comprensibles para el usuario final.

4) generate_answer(context, question):

Interactúa con el modelo de Hugging Face para generar una respuesta basada en el prompt construido.

Llama a prompt_prep para construir el mensaje completo con el contexto y la pregunta.

Envía el prompt al modelo Qwen/Qwen2.5-72B-Instruct mediante la API de Hugging Face. Y luego devuelve la respuesta generada, asegurándose de que sea clara, relevante y específica para la consulta.

Manejo de errores: Si ocurre un problema durante la generación, devuelve un mensaje genérico de error.

En esta parte del trabajo elegí el modelo Qwen/Qwen2.5-72B-Instruct por su capacidad avanzada para seguir instrucciones complejas, su potencia con 72 mil millones de parámetros, y su optimización para tareas conversacionales. Es ideal para manejar preguntas específicas sobre el juego, adaptándose rápidamente al dominio, ofreciendo precisión y soporte multilingüe. Esto asegura respuestas claras, relevantes y ajustadas al contexto recuperado.

Ejercicio 2

El agente desarrollado utiliza el concepto ReAct (Reasoning + Acting) para responder preguntas relacionadas con el juego de mesa. Este agente combina herramientas específicas para buscar información y un modelo de lenguaje grande (LLM) para interpretar las consultas y generar respuestas precisas.

Uso de Herramientas:

- retrieve_vectors: Busca información sobre reglas y mecánicas del juego en documentos vectoriales.
- retrieve_dataframe: Proporciona estadísticas del juego desde un DataFrame tabular.
- retrieve_graph: Recupera información estructurada desde una base de datos de grafos, como diseñadores o juegos relacionados.

Librería Llama-Index

Se utiliza Llama-Index para integrar el modelo ReActAgent y sus herramientas, asegurando la capacidad de razonar, elegir la herramienta más adecuada y actuar en función de la consulta del usuario.

Prompt para el Agente

Se construyó un prompt detallado que instruye al agente sobre:

- Su rol como experto en "Lost Ruins of Arnak".
- El formato que debe seguir al procesar una consulta:
- Pensamiento (Thought): Razona sobre la consulta.
- Acción (Action): Selecciona la herramienta adecuada.
- Observación (Observation): Usa los resultados de la herramienta.
- Respuesta Final (Final Answer): Responde de manera clara y precisa.
- Ejemplos concretos de cómo manejar diferentes tipos de preguntas (reglas, diseñadores, etc.).

Implementación de ReActAgent:

- El agente fue configurado con las herramientas y el modelo llama3.2:latest.
- Se estableció un flujo de trabajo lógico para cada consulta:
    - Interpretar la pregunta del usuario.
    - Determinar la fuente de datos más relevante.
    - Utilizar la herramienta adecuada para buscar la información.
    - Responder basándose en los resultados obtenidos.

Ejemplo de Flujo del Agente

Consulta: " How many players is the game for?"

- Pensamiento: Determinar que se necesita información sobre recursos iniciales.
- Acción: Usar la herramienta retrieve_vectors con el prompt exacto.
- Observación: Recupera los datos sobre los recursos iniciales.
- Respuesta Final: Proporciona una explicación clara.

Consulta: "Who designed Lost Ruins of Arnak?"

- Pensamiento: Identificar que se requiere información de diseñadores.
- Acción: Usar la herramienta retrieve_graph.
- Observación: Devuelve los nombres de los diseñadores desde la base de datos de grafos.
- Respuesta Final: "The game was designed by Míchal Štach and Elwen."

El agente implementa el enfoque ReAct para combinar razonamiento lógico con acciones específicas mediante herramientas. Esto garantiza respuestas precisas y bien fundamentadas, adaptándose a la naturaleza variada de las consultas sobre "Lost Ruins of Arnak". Además, el uso de prompts bien diseñados mejora la interacción del agente con el usuario, maximizando su utilidad.

Algunos ejemplos:

Pregunta	Respuesta
How many players can play Lost Ruins of Arnak?	The Lost Ruins of Arnak is designed for 1-4 players.
When was Lost Ruins of Arnak released?	Lost Ruins of Arnak was first published in 2020 by Czech Games Edition (CGE) in an English version.
Is Lost Ruins of Arnak good for beginners?	The Lost Ruins of Arnak is not necessarily a beginner-friendly game, as it has complex rules and mechanics that may be overwhelming for new players. However, the game does provide some guidance and explanations throughout the rulebook, which can help new players learn the game more easily. It's recommended that beginners start with the tutorial or introductory scenarios to get familiar with the game mechanics before diving into the main campaign.
Does Lost Ruins of Arnak involve dice?	The Lost Ruins of Arnak does not involve dice directly in its main mechanics. Instead, it uses worker placement, deck building, and resource management to progress through research tracks and defeat creatures for victory points.
How many dice are used in Lost Ruins of Arnak?	I couldn't find any information about dice usage in Lost Ruins of Arnak. The game mechanics seem to focus on worker placement, deck building, and exploration, but I couldn't find any mention of dice.

Aspectos Positivos:

El agente genera respuestas completas, precisas y específicas para cada pregunta. Esto demuestra que el agente comprende bien el contexto del juego Lost Ruins of Arnak.

Manejo de Preguntas Variadas:

Las preguntas abarcan diferentes tipos de categorías (stats, gameplay, review), y el agente las maneja correctamente:

"How many players can play Lost Ruins of Arnak?": Clasifica como stats y proporciona información precisa sobre el número de jugadores.

"When was Lost Ruins of Arnak released?": También categorizada como stats, entregando la fecha de publicación y editor.

"Is Lost Ruins of Arnak good for beginners?": Respuesta más elaborada que reconoce la complejidad del juego y brinda recomendaciones útiles, evidenciando un entendimiento contextual.

"Does Lost Ruins of Arnak involve dice?": Respuesta precisa que aborda la mecánica del juego, indicando que no incluye dados directamente. Aunque en otras pruebas sí proporcionó una cantidad de datos.

Conclusiones

En este proyecto se implementaron técnicas avanzadas de NLP, como RAG (Retrieval Augmented Generation) y modelos de lenguaje, para desarrollar un asistente inteligente especializado en el juego Lost Ruins of Arnak. Integrando fuentes de datos vectoriales, gráficas y tabulares, el sistema logró responder preguntas contextuales de manera precisa, demostrando adaptabilidad y soporte bilingüe mediante traducción automática. Este enfoque permitió manejar consultas diversas sobre estadísticas, mecánicas, reseñas y detalles adicionales del juego.

Aunque el sistema mostró un buen rendimiento general, especialmente en preguntas estadísticas y datos contextuales, algunas categorías como reseñas y mecánicas podrían beneficiarse de un conjunto de datos más amplio y balanceado, además de ajustes en los clasificadores y herramientas de recuperación. En conjunto, este trabajo destaca el potencial de integrar herramientas y modelos avanzados en aplicaciones específicas, sentando una base sólida para futuros desarrollos más escalables, precisos y robustos.

Anexo

Preguntas para probar el asistente

En inglés:

"How do I use artifacts in Lost Ruins of Arnak?"
"What is the turn order in the game?"
"How do you win Lost Ruins of Arnak?"
"What are the worker placement rules in the game?"
"Can you explain the deck-building mechanic in Lost Ruins of Arnak?"
"What is the average rating of Lost Ruins of Arnak?"
"How many awards has the game won?"
"What is the game's ranking in the adventure category?"
"How does the game's complexity compare to similar games?"
"How many players rated Lost Ruins of Arnak?"
"Is Lost Ruins of Arnak worth playing?"
"How does Lost Ruins of Arnak compare to Dune: Imperium?"
"What do players like most about Lost Ruins of Arnak?"
"What are the main criticisms of the game?"
"Is Lost Ruins of Arnak good for beginners?"
"Who designed Lost Ruins of Arnak?"
"What other games did Czech Games Edition publish?"
"What are the alternate names for Lost Ruins of Arnak?"
"Who illustrated the game?"
"When was Lost Ruins of Arnak released?"

En español:

"¿Cómo se usan los artefactos en Lost Ruins of Arnak?"
"¿Cuál es el orden de turno en el juego?"
"¿Cómo se gana en Lost Ruins of Arnak?"
"¿Cuáles son las reglas de colocación de trabajadores en el juego?"
"¿Puedes explicar la mecánica de construcción de mazos en Lost Ruins of Arnak?"
"¿Cuál es la calificación promedio de Lost Ruins of Arnak?"
"¿Cuántos premios ha ganado el juego?"
"¿En qué posición está el juego en la categoría de aventura?"
"¿Cómo se compara la complejidad del juego con juegos similares?"
"¿Cuántos jugadores han calificado Lost Ruins of Arnak?"
"¿Vale la pena jugar a Lost Ruins of Arnak?"
"¿Cómo se compara Lost Ruins of Arnak con Dune: Imperium?"
"¿Qué es lo que más les gusta a los jugadores de Lost Ruins of Arnak?"
"¿Cuáles son las principales críticas del juego?"

"¿Es Lost Ruins of Arnak bueno para principiantes?”
"¿Quién diseñó Lost Ruins of Arnak?"
"¿Qué otros juegos publicó Czech Games Edition?"
"¿Cuáles son los nombres alternativos de Lost Ruins of Arnak?"
"¿Quién ilustró el juego?"
"¿Cuándo se lanzó Lost Ruins of Arnak?"