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

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.

Ejercicio 1

Se debe implementar un chatbot especializado en el juego de mesa utilizando la técnica Retrieval Augmented Generation (RAG). El sistema integrará tres fuentes principales de conocimiento:

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

El chatbot debe ser 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 un modelo entrenado, comparando sus resultados.

Base de datos

En lo que respecta a la base de datos la información se obtuvo principalmente de los links proporcionados por la cátedra. Uno de ellos[1] cuenta con una sección de 'files', donde se encuentran varios archivos pdf que fueron analizados para ver cuáles podían ser incluidos a la base de datos de documentos de texto.

Una decisión importante en esta parte del desarrollo fue en qué idioma generamos la base de datos. Decidí generar la base de datos en inglés debido a la mayor compatibilidad y rendimiento de las herramientas y modelos de procesamiento de lenguaje natural en este idioma, así como la abundante disponibilidad de recursos y datos en inglés que facilitan la integración.

Se realizó Web Scraping utilizando Selenium para obtener datos de la misma página web mencionada. Y BeautifulSoup del segundo link proporcionado. En el último caso la información si estaba en español por lo que fue necesario traducirla.

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 aregar a la base de datos tabular.

Luego de generar la base de datos de documentos, el conjunto de textos se dividio 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.

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

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.

Clasificador

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

Se crean preguntas (con 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.
- similar_games: Preguntas sobre juegos similares a "Lost Ruins of Arnak".

Estas preguntas cubren diversos aspectos del juego y sirven como ejemplos etiquetados.

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

Las columnas del DataFrame son:

- prompt: Contiene las preguntas simuladas.
- type: Contiene la categoría correspondiente.
- Visualización:

Random Forest

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

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

```
Prediction: review
```

HuggingFace LLM

Utilizando la API de Hugging Face se implementa un clasificador de preguntas relacionadas con el juego de mesa en una de las siguientes categorías: gameplay, stats, review, o similar_games. El clasificador llama al modelo Qwen/Qwen2.5-72B-Instruct para obtener una respuesta, que se espera sea una de las cuatro categorías.

```
[ ] print(llm_classifier("What is the average raiting for this game?"))
```

```
review
```

```
[ ] print(llm_classifier("How many dice do I need to play the game?"))
```

```
gameplay
```

```
[ ] print(llm_classifier("Do you know any similar games?"))
```

```
similar_games
```

Flujo del asistente

- Selección de idioma

El usuario elige entre inglés (1) o español (2). Según la elección, el asistente procesa la consulta en el idioma seleccionado.

- Pregunta del usuario

En inglés: El usuario ingresa su pregunta directamente.

En español: La pregunta es traducida al inglés usando la función translate_to_english.

- Recuperación del contexto

Llama a la función retriever para obtener información relevante desde diferentes fuentes (documentos, datos tabulares o gráficas) basada en la consulta del usuario.

- Generación de respuesta:

Usa un modelo de lenguaje preentrenado (Qwen/Qwen2.5-72B-Instruct) alojado en Hugging Face para generar una respuesta basada en el contexto recuperado y el prompt construido con prompt_prep.

- Traducción (si aplica):

Si el usuario eligió español, la respuesta generada en inglés es traducida al español usando la función translate_to_spanish.

- Respuesta al usuario:

En inglés: La respuesta generada se presenta directamente.

En español: La respuesta traducida se presenta al usuario.

Ejercicio 2

En este ejercicio se extenderá el trabajo del primer ejercicio mediante la creación de un agente basado en el enfoque ReAct, implementado con la librería Llama-Index. Este agente integra herramientas para buscar información en las bases de datos de documentos, grafos y datos tabulares.

Explicación del flujo

- Entrada del usuario

El usuario ingresa su pregunta o consulta relacionada con el juego.

- Configuración del modelo LLM

Se inicializa un modelo de lenguaje preentrenado (Llama v3.2) con configuraciones específicas:
Tiempo de espera (request_timeout) de 100 segundos.
Temperatura baja (temperature=0.1) para respuestas más precisas y menos creativas.
Una ventana de contexto amplia (context_window=4000).

- Herramientas del asistente:

Se configuran herramientas específicas que el agente puede usar para responder preguntas como retrieve_vectors para información sobre reglas y mecánicas del juego.

- Agente ReAct:

El agente utiliza el enfoque ReAct (Reasoning + Acting) para resolver las consultas.
El flujo es:
    Pensamiento (Thought): Determina qué información se necesita.
    Acción (Action): Selecciona y utiliza la herramienta adecuada.

Observación (Observation): Procesa los resultados de la herramienta.
Respuesta final (Final Answer): Genera una respuesta basada en los datos obtenidos.

- Formato del sistema:

El sistema incluye un prompt detallado que define el comportamiento esperado del agente.
Ejemplos de interacciones guían el formato de las respuestas (pensamiento, acción, observación, respuesta final).

- Flujo de consulta y respuesta:

Si el usuario ingresa una consulta vacía, el sistema lo solicita nuevamente.
Si ocurre un error, se captura y devuelve un mensaje de error.

Ejemplo obtenido:

```
Lost Ruins of Arnak - GAME EXPERT ASSISTANT!
What do you need help with? How many dice do I need to play the game?
WARNING:llama_index.core.agent.react.formatter:ReActChatFormatter.fro
m_context is deprecated, please use `from_defaults` instead.
> Running step 0cecc569-73b4-4ede-bb42-2483230a1da8. Step input: How
many dice do I need to play the game?
Thought: The current language of the user is: English. I need to use
a tool to help me answer the question.
Action: retrieve_vectors
Action Input: {'properties': AttributedDict([('prompt',
AttributedDict([('title', 'Prompt for Lost Ruins of Arnak')]))])}
Observation: Error: retrieve_vectors() got an unexpected keyword
argument 'properties'
> Running step 2184a990-be78-42b8-a0cb-389bcdd176f0. Step input: None
Thought: The tool retrieve_vectors does not support the 'properties'
keyword argument. I need to find another way to use this tool.
Action: retrieve_vectors
Action Input: {'prompt': 'How many dice do I need to play the game?'}
Observation: Decks are small, and randomness in the game is heavily
mitigated by the wealth of tactical decisions offered on the game
board. With a variety of worker actions, artifacts, and equipment
cards, the set-up for each game will be unique, encouraging players to
explore new strategies to meet the challenge.
> Running step cc967ae6-c5e5-463c-ae77-7c750ee5d162. Step input: None
Thought: The tool retrieve_vectors did not provide any useful
information about the number of dice needed to play the game. I need to
rely on my general knowledge of the Lost Ruins of Arnak game.
Answer: In the Lost Ruins of Arnak, players typically use a single
six-sided die to determine which actions they can take during their
turn.
```

```
  Response: In the Lost Ruins of Arnak, players typically use a single
six-sided die to determine which actions they can take during their
turn.
```

Conclusiones

En este proyecto se pudieron poner en práctica técnicas avanzadas de NLP, como RAG y modelos de lenguaje, para crear un asistente inteligente que combina bases vectoriales, gráficas y tabulares. Logró responder preguntas contextuales de manera eficiente y adaptable, con soporte bilingüe y traducción automática. Aunque el sistema mostró un buen rendimiento en ciertas categorías, áreas como reseñas y mecánicas podrían mejorarse con datos más balanceados y ajustes adicionales. En conjunto, este trabajo demuestra el potencial de integrar herramientas y modelos modernos en aplicaciones especializadas, sentando las bases para futuros desarrollos escalables y robustos.