



Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Procesamiento del Lenguaje Natural
Trabajo Práctico No. 2
17/12/2024

Docentes: Juan Pablo Manson, Alan Geary
Giuliano Crenna (C-7438/1)

Contenidos

1	Introducción	2
2	Ejercicio 1	3
2.1	Resumen	3
2.2	Desarrollo	3
2.2.1	Fuentes de datos	3
2.2.2	Preparación de Datos	3
2.2.3	Creación de la Base de Datos Vectorial	4
2.2.4	Recuperación y Expansión de Información	5
2.3	Construcción del RAG	5
2.4	Resultados del RAG	6
2.5	Construcción del Modelo de Clasificación	7
2.5.1	Resultados de la Búsqueda con RandomForest	7
2.6	Conclusiones	7
3	Ejercicio 2	8
3.1	Resumen	8
3.2	Desarrollo	8
3.2.1	Configuración del Modelo de Lenguaje (LLM)	8
3.2.2	Inicio del Servidor LLM	8
3.2.3	Selección de Herramientas	8
3.2.4	Creación del Agente ReAct	9
3.2.5	Prueba del Agente	9
3.2.6	Robustez y Manejo de Errores	10
3.2.7	Resultados	10
4	Enlaces	11

1 Introducción

El presente informe constituye el Trabajo Práctico Final de la materia TUIA NLP 2024, centrado en el diseño y desarrollo de un chatbot experto basado en la técnica de *Retrieval Augmented Generation* (RAG) aplicado al juego de mesa *Las ruinas perdidas de Arnak*. Este trabajo tiene como objetivo integrar conocimientos adquiridos a lo largo de la cursada, implementando soluciones avanzadas de procesamiento de lenguaje natural, recuperación de información y generación de texto.

El proyecto se divide en dos ejercicios principales. En el primero, se implementa un sistema capaz de interactuar con los usuarios, respondiendo preguntas en español e inglés mediante consultas dinámicas a diversas fuentes de datos, como documentos textuales, tablas y bases de datos de grafos. En el segundo, se extiende esta funcionalidad al incorporar un agente basado en el modelo ReAct, que utiliza herramientas específicas para buscar información en dichas fuentes, maximizando la eficacia y precisión de las respuestas.

La temática propuesta por la cátedra, *Las ruinas perdidas de Arnak*, es un Eurogame destacado, lo que añade un desafío adicional al tratamiento de los datos disponibles y a la generación de respuestas contextualmente relevantes. Este informe detalla las decisiones de diseño, implementación y justificación de cada componente desarrollado, presentando además un análisis crítico de los resultados obtenidos y propuestas de mejora.

2 Ejercicio 1

2.1 Resumen

El código presentado aborda un problema de procesamiento y análisis de información relacionada con el juego de mesa "Las Ruinas Perdidas de Arnak". Este problema incluye la descarga y preprocesamiento de datos en diferentes formatos (texto, audio, PDF y CSV), la construcción de un modelo de clasificación basado en Random Forest, y la creación de un sistema de recuperación de información empleando bases de datos vectoriales y modelos de lenguaje.

2.2 Desarrollo

2.2.1 Fuentes de datos

Se tomaron las siguientes fuentes de datos para la posterior creación de las bases de datos. Estas fuentes proporcionan información valiosa y variada sobre el juego *Las Ruinas Perdidas de Arnak*, y fueron seleccionadas por su relevancia, accesibilidad y capacidad para ser estructuradas en bases de datos adecuadas para el sistema.

1. **Reseñas de páginas web:** Las reseñas de diversas páginas web especializadas en juegos de mesa fueron utilizadas como fuente primaria de información para comprender la recepción general del juego, las mecánicas, las reglas y las impresiones de los jugadores. Estas reseñas permiten obtener una visión amplia de los aspectos más destacados del juego, como su complejidad, duración y la experiencia que ofrece a los jugadores. Las reseñas fueron extraídas de sitios web especializados en juegos de mesa y blogs de expertos. Este tipo de información es útil para contextualizar el juego y proporcionar datos sobre su popularidad y críticas.
2. **Manual en PDF:** El manual oficial del juego (en Español e Inglés), disponible en formato PDF, fue otro recurso fundamental utilizado. Este manual contiene las reglas completas del juego, instrucciones sobre cómo jugar, los componentes del juego y detalles sobre las mecánicas y estrategias que los jugadores pueden seguir. La información obtenida del manual fue esencial para comprender las reglas y la lógica interna del juego, lo que permite proporcionar respuestas precisas y fundamentadas a las preguntas sobre cómo jugar y cómo se desarrollan las partidas.
3. **Transcripciones de videos de YouTube:** Los videos en YouTube de reseñas, tutoriales y partidas de *Las Ruinas Perdidas de Arnak* fueron transcritos para extraer información adicional sobre las mecánicas del juego, estrategias recomendadas y experiencias de los jugadores. Estas transcripciones incluyen entrevistas con diseñadores de juegos, análisis de expertos y comentarios de jugadores durante las partidas. Al transcribir y organizar esta información, se logró capturar detalles que no están siempre claramente descritos en otros recursos, y se añadió una perspectiva práctica y vivencial del juego.

2.2.2 Preparación de Datos

1. Se descargan los archivos necesarios desde fuentes externas utilizando comandos `wget`, incluyendo reseñas, reglas del juego en PDF, audio y un archivo CSV de preguntas y

respuestas.

2. Se implementan funciones específicas para procesar diferentes tipos de archivos:

- Extracción de texto desde páginas web y archivos de texto plano.
- Extracción de texto desde documentos PDF usando `PdfReader`.
- Procesamiento de archivos de audio en formato M4A, dividiendo en fragmentos y utilizando `speech_recognition` para la transcripción.
- Lectura y transformación de datos CSV, preparándolos para su inserción en una base de datos vectorial.

3. Los textos procesados se dividen en fragmentos manejables usando el `RecursiveCharacterTextSplitter`, lo que permite una mejor manipulación y almacenamiento.

2.2.3 Creación de la Base de Datos Vectorial

La base de datos vectorial fue diseñada para almacenar y recuperar información de manera eficiente a partir de grandes volúmenes de datos no estructurados. Para este propósito, se utilizó la librería `chromadb`, una herramienta de bases de datos vectoriales persistentes. Esta base de datos permite almacenar representaciones de alta dimensión de los datos, facilitando la búsqueda semántica y mejorando el rendimiento en consultas relacionadas con la información contenida en los textos.

Los datos procesados, como las preguntas y respuestas relacionadas con el juego *Las Ruinas Perdidas de Arnak*, fueron transformados en vectores de características mediante embeddings. Estos embeddings representan el contenido textual de una manera que captura su significado semántico, permitiendo comparar y buscar contenido relevante basado en el contexto de las consultas del usuario.

Para la generación de estos embeddings, se empleó el modelo `all-MiniLM-L6-v2` de la librería `SentenceTransformer`, el cual es un modelo preentrenado optimizado para generar representaciones vectoriales de textos en un espacio de alta dimensión. Este modelo fue elegido debido a su eficiencia y capacidad para producir representaciones semánticas de alta calidad, lo que es fundamental para la búsqueda rápida y precisa dentro de grandes conjuntos de datos textuales.

Una vez que los datos son procesados y los embeddings generados, se almacenan en la base de datos `chromadb`. Esta base de datos vectorial permite realizar consultas rápidas mediante técnicas de búsqueda por proximidad, utilizando métricas de similitud como la distancia del coseno. De esta manera, cuando el agente recibe una consulta, puede buscar de manera eficiente las respuestas más relevantes basándose en la proximidad semántica de las representaciones vectoriales.

El uso de `chromadb` ofrece varias ventajas, como la capacidad de manejar grandes volúmenes de datos y la persistencia de los embeddings a lo largo del tiempo, lo que garantiza que los datos no se pierdan y puedan ser utilizados para futuras consultas o actualizaciones. La estructura de la base de datos también facilita la integración con otras fuentes de datos y herramientas, permitiendo que la base de datos vectorial sea un componente clave en el sistema de respuesta automática.

Además, la base de datos vectorial fue diseñada para ser escalable, lo que significa que puede expandirse fácilmente a medida que se incorporen más datos o se requiera mayor

capacidad de almacenamiento. Esto asegura que el sistema pueda adaptarse a futuras actualizaciones o ampliaciones del conjunto de datos sin comprometer su rendimiento o eficiencia.

2.2.4 Recuperación y Expansión de Información

1. Se implementa un sistema de recuperación de información que permite realizar consultas a la base vectorial.
2. Las consultas se enriquecen con información relevante recuperada de la base vectorial para generar prompts más completos.
3. La generación de respuestas se realiza mediante la API de OpenAI, incorporando el contexto ampliado.

2.3 Construcción del RAG

El desarrollo del agente RAG se compone de los siguientes pasos clave:

- **Inicialización del Agente:** Se define una clase `Agente` que incluye los elementos necesarios: un modelo de embeddings (`SentenceTransformer`), una base de datos vectorial (`ChromaDB`), y un archivo CSV para consultas tabulares.
- **Consulta Tabular:** Mediante la función `consultar_csv()`, el sistema busca coincidencias exactas o parciales en el archivo CSV. Si encuentra una coincidencia para la pregunta, devuelve la respuesta asociada.
- **Búsqueda Semántica:** Si no hay respuesta en el CSV, se utiliza la función `realizar_búsqueda_semantica()`, que emplea embeddings generados por el modelo para buscar respuestas relevantes en la base de datos vectorial.
- **Predicción de Respuesta:** La función `predecir_respuesta()` combina ambas fuentes de información: primero consulta el CSV, luego recurre a la base vectorial si es necesario. Además, actualiza el contexto del agente con la respuesta seleccionada.
- **Interacción con el Usuario:** El método `chat()` permite una interacción continua con el usuario. Las respuestas generadas se basan en las consultas realizadas, utilizando el contexto acumulado para enriquecer las interacciones.

El modelo se apoya en un flujo híbrido, asegurando que las respuestas sean precisas y contextualmente relevantes, aprovechando tanto datos estructurados como no estructurados.

2.4 Resultados del RAG

Pregunta	Respuesta
¿Cuáles son las mecánicas principales de Las Ruinas Perdidas de Arnak?	Las mecánicas principales de Las Ruinas Perdidas de Arnak son la gestión de recursos, la exploración de la isla, la mejora de tu mazo de cartas, la colocación de trabajadores (peones de arqueólogos), la resolución de desafíos, la utilización estratégica de tus acciones y la obtención de puntos de victoria. ¡Disfruta descubriendo las ruinas perdidas de Arnak!
¿Cómo se juega el modo de exploración en Las Ruinas Perdidas de Arnak?	En el modo de exploración de Las Ruinas Perdidas de Arnak, los jugadores pueden enviar a sus arqueólogos a explorar nuevas ubicaciones en la isla de Arnak. Para explorar, los jugadores deben pagar la cantidad indicada en recursos y colocar un peón de arqueólogo en la ubicación deseada. Al explorar, los jugadores pueden descubrir nuevos lugares, obtener beneficios adicionales y descubrir secretos ocultos en las distintas ubicaciones de la isla.
¿Cuántos jugadores pueden participar en Las Ruinas Perdidas de Arnak?	El juego Las Ruinas Perdidas de Arnak está diseñado para ser jugado por 1 a 4 jugadores. ¡Así que reúne a tus amigos o disfruta de una partida en solitario explorando las ruinas de Arnak!
¿Cuál es la duración aproximada de una partida?	La duración aproximada de una partida de Las Ruinas Perdidas de Arnak suele ser de 60 a 120 minutos, dependiendo de la cantidad de jugadores y el ritmo de juego. ¡Prepárate para sumergirte en la exploración y la búsqueda de reliquias en la isla de Arnak!

Table 1: Resultados.

2.5 Construcción del Modelo de Clasificación

- Se utiliza un modelo `Random Forest` para clasificar preguntas basándose en sus características extraídas mediante `TfidfVectorizer`.
- El dataset de preguntas y respuestas se transforma, etiquetando las respuestas de forma categórica.
- El conjunto de datos se divide en entrenamiento y prueba. El modelo es entrenado con `RandomForestClassifier` y evaluado usando un reporte de clasificación.

2.5.1 Resultados de la Búsqueda con `RandomForest`

Pregunta	Respuesta
¿Cuántos tipos de tableros existen en el juego base?	Hay 12 ídolos en el tablero principal.
¿Qué tipo de objetos se encuentran en las cartas de artefactos?	Botas, coches, barcos y aviones.
¿Cuántos ídolos pueden ser obtenidos durante el juego?	Czech Games Edition (CGE).
¿Cuántas cartas de equipo se asignan a cada jugador al comenzar?	35 cartas.

Table 2: Resultados.

2.6 Conclusiones

El código logra integrar múltiples fuentes de datos y formatos en un flujo coherente, combinando técnicas de preprocesamiento, recuperación de información y clasificación. La implementación del modelo de `Random Forest` proporciona una herramienta eficiente para predecir respuestas basadas en preguntas de usuarios, mientras que la base vectorial y los modelos de lenguaje enriquecen la experiencia de búsqueda de información.

Sin embargo, se identificaron puntos de mejora como:

- Optimización del procesamiento de grandes volúmenes de datos en la base vectorial.
- Ampliación del conjunto de datos para cubrir más preguntas y respuestas.
- Refinamiento del modelo de clasificación para mejorar la precisión.

3 Ejercicio 2

3.1 Resumen

En este ejercicio, se analiza un agente implementado en Python que combina capacidades de búsqueda semántica y consultas tabulares para responder preguntas. Este agente utiliza un modelo de embeddings, una base de datos vectorial (ChromaDB) y un archivo CSV con preguntas y respuestas predefinidas. A continuación, se describen los pasos principales de la implementación, su desarrollo y las soluciones adoptadas.

3.2 Desarrollo

3.2.1 Configuración del Modelo de Lenguaje (LLM)

El agente utiliza el modelo de lenguaje `Ollama` con la versión `llama3.2:latest`. Esta elección se tomó debido a las capacidades avanzadas del modelo para generar respuestas precisas y coherentes, algo fundamental para responder preguntas de forma efectiva. Las decisiones tomadas al respecto son las siguientes:

- **Tiempo de espera (Timeout):** Se configuró un tiempo de espera de 30 segundos (`request_timeout=30.0`) para evitar que el sistema quede colgado en caso de que no se obtenga una respuesta o si hay algún retraso en la comunicación con el servidor.
- **Temperatura:** La temperatura se estableció en 0.3 (`temperature=0.3`) para generar respuestas más coherentes y menos creativas. Esta temperatura baja asegura que el modelo no se desvíe demasiado del contenido esperado.
- **Contexto:** El tamaño de la ventana de contexto (`context_window=4096`) fue configurado a 4096 palabras para permitir que el modelo maneje un contexto más amplio en consultas más complejas.

3.2.2 Inicio del Servidor LLM

Para ejecutar el servidor en segundo plano y que pueda mantenerse activo incluso si la terminal se cierra, se utilizó el comando `nohup`. Este comando permite que el servidor escuche en el puerto 8000 y reciba solicitudes sin interrupciones.

```
!nohup litellm --model ollama/llama3.2:latest --port 8000 > litellm.log  
2>&1 &
```

El uso de `nohup` garantiza que el modelo permanezca activo sin interferir con otras tareas.

3.2.3 Selección de Herramientas

Se eligieron tres herramientas para proporcionar al agente acceso a información de diferentes fuentes. Las herramientas son las siguientes:

1. **Base de Datos Vectorial (`get_vector`):** Esta herramienta es la principal, ya que permite recuperar información precisa en formato vectorial, ideal para encontrar respuestas exactas a preguntas específicas.

2. **Base de Datos CSV (`get_csv_data`):** Se añadió como una fuente secundaria para obtener información almacenada en un archivo CSV. Es útil para obtener respuestas a partir de datos tabulados.
3. **Wikipedia (`wikipedia`):** Esta herramienta sirve como fuente de respaldo para obtener información general sobre el juego que no está cubierta por las bases de datos vectoriales o CSV.

La elección de estas herramientas responde a la necesidad de combinar información estructurada (bases de datos) con información más general (Wikipedia), lo que da al agente un conjunto de herramientas diverso para procesar diferentes tipos de consultas.

3.2.4 Creación del Agente ReAct

El agente **ReAct** fue configurado para interactuar con las herramientas disponibles y seguir un proceso lógico al manejar las consultas. La configuración es la siguiente:

- **Análisis de la consulta:** El agente primero analiza la consulta para identificar el tipo de información que se necesita.
- **Selección de herramientas:** El agente selecciona la herramienta más adecuada para procesar la consulta, con preferencia por la herramienta `get_vector` como la principal.
- **Combinación de resultados:** El agente combina los resultados obtenidos de las herramientas en una respuesta clara y concisa.

Se configuraron también reglas específicas para garantizar que el agente maneje las consultas de manera coherente y eficiente:

- **No modificar la consulta:** Cada consulta se trata de manera independiente, sin influencias previas de consultas anteriores.
- **Priorizar las herramientas:** El agente debe elegir la herramienta más relevante para cada consulta, según el contexto de la pregunta.
- **Usar todas las herramientas:** El agente debe aprovechar todas las herramientas disponibles para garantizar una respuesta completa.
- **Nunca usar información externa:** El agente solo debe basarse en la información proveniente de las herramientas definidas, evitando respuestas que no estén respaldadas por ellas.

3.2.5 Prueba del Agente

Para comprobar la efectividad del agente, se definieron varias preguntas relacionadas con el juego *Las Ruinas Perdidas de Arnak*. Estas preguntas cubren aspectos como las mecánicas del juego, el objetivo principal y el contenido del juego. A través de estas pruebas, se verifica que el agente pueda acceder correctamente a las bases de datos y generar respuestas precisas.

3.2.6 Robustez y Manejo de Errores

El agente cuenta con un sistema de manejo de excepciones que asegura su robustez frente a posibles errores. Si ocurre algún problema durante el procesamiento de las consultas, el sistema captura la excepción y continúa ejecutándose sin interrupciones.

3.2.7 Resultados

Pregunta	Respuesta
¿Qué es el objetivo principal del juego Las Ruinas Perdidas de Arnak?	El objetivo principal del juego Las Ruinas Perdidas de Arnak es explorar y desbloquear áreas del tablero para obtener la mayor cantidad de puntos de victoria, mientras se enfrenta a desafíos y obstáculos durante las rondas.
¿Cuáles son las mecánicas principales de Las Ruinas Perdidas de Arnak?	<p>Las mecánicas principales del juego Las Ruinas Perdidas de Arnak incluyen:</p> <ul style="list-style-type: none"> • Exploración: el jugador explora áreas desconocidas del tablero para obtener objetos antiguos y tomos. • Coleccionismo: el jugador recopila objetos antiguos y tomos para obtener puntos de victoria. • Desbloqueo: el jugador desbloquea áreas del tablero para acceder a nuevos objetos y tomes. • Gestión de recursos: el jugador debe gestionar sus recursos (como energía y movimientos) para explorar y coleccionar objetos. <p>El juego combina estas mecánicas con un sistema de cartas que agrega una capa de estrategia y sorpresa al juego.</p>
How many player can play in Las Ruinas Perdidas de Arnak?	The game Las Ruinas Perdidas de Arnak is designed for 1-2 players.

Table 3: Resultados.

4 Enlaces

- Repositorio de datos: Repositorio de TUIA-NLP en GitHub. Este repositorio contiene el código fuente y los datos utilizados en el proyecto, incluyendo scripts de procesamiento de datos, modelos de machine learning y bases de datos relevantes para el análisis de datos en procesamiento de lenguaje natural (NLP). [16-12-2024]
- Documentación de ChromaDB: ChromaDB - Documentación oficial. ChromaDB es una base de datos vectorial eficiente, utilizada para la creación de aplicaciones de búsqueda y recomendación, donde la información es procesada en vectores. La documentación proporciona detalles sobre cómo configurar y utilizar ChromaDB para almacenar y consultar grandes cantidades de datos de manera eficiente. [16-12-2024]
- Documentación de SentenceTransformers: SentenceTransformers - Documentación oficial. SentenceTransformers es una biblioteca de Python para generar representaciones vectoriales (embeddings) de oraciones y párrafos, que se utiliza ampliamente en tareas de NLP como la búsqueda semántica y la similitud de texto. Esta documentación explica cómo integrar el modelo ‘all-MiniLM-L6-v2’ para obtener embeddings eficientes en tareas de NLP. [16-12-2024]
- Documentación de OpenAI: OpenAI - Documentación oficial. OpenAI ofrece acceso a modelos de lenguaje avanzados como GPT-3, que pueden ser utilizados para generar texto, comprender consultas y realizar tareas relacionadas con el procesamiento de lenguaje natural. Esta documentación cubre cómo interactuar con los modelos de OpenAI, sus API y cómo implementar soluciones basadas en inteligencia artificial. [16-12-2024]
- Documentación de Hugging Face: Hugging Face - Plataforma de NLP. Hugging Face es una plataforma líder en modelos de aprendizaje automático, especialmente en el campo de procesamiento de lenguaje natural. En este sitio encontrarás una amplia variedad de modelos preentrenados, herramientas y tutoriales para facilitar el desarrollo de aplicaciones de NLP. [16-12-2024]
- Documentación de Ollama: Ollama - Biblioteca de modelos de lenguaje. Ollama proporciona una plataforma para ejecutar modelos avanzados de lenguaje como Llama, permitiendo crear agentes de conversación eficaces y herramientas de NLP para diversos casos de uso. La documentación de Ollama incluye guías de instalación y ejemplos de integración. [16-12-2024]
- Documentación de TensorFlow: TensorFlow - Documentación oficial. TensorFlow es una biblioteca de código abierto para el desarrollo de modelos de machine learning, incluyendo redes neuronales profundas. Este recurso es fundamental para el desarrollo de aplicaciones que involucren grandes volúmenes de datos y modelos complejos en NLP y otras áreas. [16-12-2024]
- Documentación de PyPDF2: PyPDF2 - Python Package Index. PyPDF2 es una biblioteca de Python para la manipulación de archivos PDF, permitiendo fusionar, dividir y extraer información de documentos PDF. Este paquete es útil para proyectos que requieren el procesamiento de archivos PDF, como la extracción de texto para análisis de datos o la creación de reportes automatizados. [16-12-2024]

- Documentación de Deep-Translator: Deep-Translator - Repositorio GitHub. Deep-Translator es una biblioteca de Python que permite traducir textos entre varios idiomas utilizando diversos motores de traducción, como Google Translate, Yandex, entre otros. Es útil para integrar traducción automática en proyectos de NLP. [16-12-2024]
- Documentación de Rank-BM25: Rank-BM25 - Python Package Index. Rank-BM25 es un algoritmo de recuperación de información basado en el modelo de probabilidad de relevancia, utilizado para realizar búsquedas de texto eficaces. Es ampliamente usado en motores de búsqueda y sistemas de recomendación. [16-12-2024]
- Documentación de LlamaIndex: LlamaIndex - Documentación oficial. LlamaIndex es una herramienta diseñada para integrar bases de datos y sistemas de indexación de texto con modelos de lenguaje, facilitando la creación de agentes inteligentes que interactúan con datos no estructurados. [16-12-2024]