
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

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 - RAG

## 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 se diseñó para almacenar y recuperar datos no estructurados de manera eficiente. Se utilizó la librería `chromadb`, que permite guardar representaciones vectoriales de alta dimensión, facilitando la búsqueda semántica y mejorando el rendimiento en consultas de textos.

Las preguntas y respuestas sobre el juego *Las Ruinas Perdidas de Arnak* fueron transformadas en vectores mediante embeddings. Estos embeddings capturan el significado semántico del texto, permitiendo comparar y buscar contenido relevante según el contexto de las consultas.

Se utilizó el modelo preentrenado `all-MiniLM-L6-v2` de `SentenceTransformer`, que genera representaciones vectoriales de alta calidad de manera eficiente, lo cual es clave para búsquedas rápidas y precisas.

Los embeddings generados se almacenan en `chromadb`, lo que permite consultas rápidas basadas en la similitud semántica utilizando la distancia del coseno. Esto mejora la eficiencia al buscar respuestas relevantes a las consultas del usuario.

`chromadb` ofrece ventajas como la capacidad de manejar grandes volúmenes de datos y la persistencia de los embeddings, asegurando que los datos no se pierdan y puedan ser reutilizados. Además, la base de datos es escalable, lo que permite agregar más datos o aumentar la capacidad de almacenamiento sin afectar el rendimiento.

Al realizar consultas a la base de datos vectorial se obtienen los siguientes resultados.

```
Query: Cuanto cuesta comprar el juego?
Resultado 1:
Texto: ¿Cuánto cuesta el juego en el mercado?
Metadata: {'answer': '59,99€.'}
ID: doc_19
--------------------------------------------------
Query: Como se juega?
Resultado 1:
Texto: ¿Qué tipo de caja tiene el juego?
Metadata: {'answer': 'Una caja rectangular de dos piezas (tapa y fondo).'}
ID: doc_20
--------------------------------------------------
Query: Cantidad de jugadores
Resultado 1:
Texto: ¿Cuál es el rango de jugadores permitido?
Metadata: {'answer': 'De 1 a 4 jugadores.'}
ID: doc_14
```

Figure 1: Consultas a la base de datos vectorial.

### 2.2.4 Base de Datos de Grafos

Para modelar las relaciones entre los distintos elementos del juego *Las Ruinas Perdidas de Arnak*, se utilizó una base de datos de grafos. En este tipo de base de datos, los elementos (nodos) y sus relaciones (aristas) se representan de forma explícita, lo que facilita las consultas y la exploración de interacciones complejas.

El grafo se construyó utilizando la librería `NetworkX`, que proporciona herramientas para crear y manipular grafos de manera eficiente en Python. En particular, `NetworkX` permite añadir nodos y aristas con etiquetas y propiedades personalizadas, lo que resulta útil para modelar las distintas categorías de objetos en el juego, como "Lugar", "Mecánica", "Entidad" y "Recurso". Además, facilita la creación de relaciones entre nodos, como las conexiones entre los templos, los guardianes y los lugares del juego.

La decisión de utilizar `NetworkX` se basó en su facilidad de uso, su capacidad para trabajar con grafos complejos y su integración con otras librerías de análisis y visualización. Sin embargo, para la persistencia y la consulta eficiente de grafos más grandes y complejos, se optó por `RedisGraph`, una extensión de `Redis` diseñada específicamente para manejar grafos de manera escalable y rápida.

`RedisGraph` permite realizar consultas de grafos utilizando un lenguaje similar a `Cypher` (utilizado en bases de datos de grafos como `Neo4j`), lo que facilita la obtención de relaciones complejas y la ejecución de consultas de manera eficiente. Esta elección se tomó porque `RedisGraph` es altamente escalable, lo que permite manejar grandes volúmenes de datos sin perder rendimiento, y su integración con `Redis` garantiza la persistencia de los grafos a lo largo del tiempo.

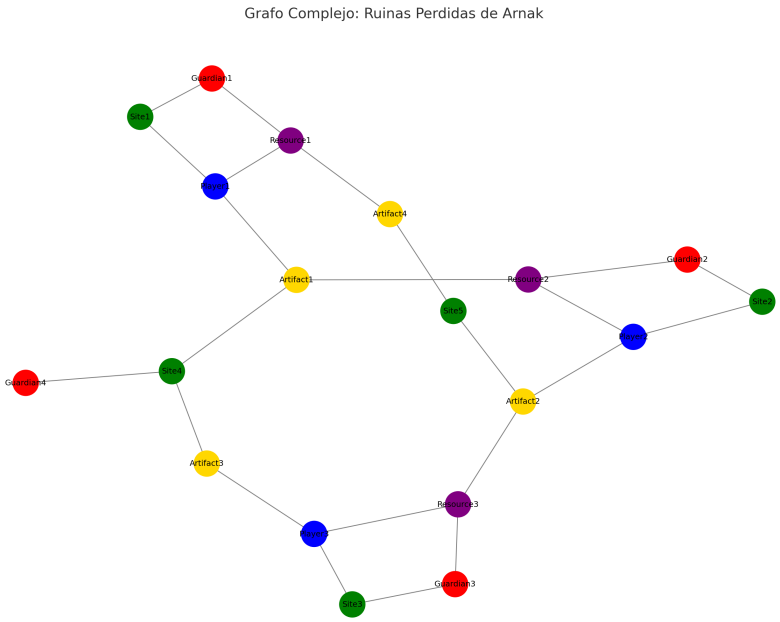


Figure 2: Grafo

### 2.2.5 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

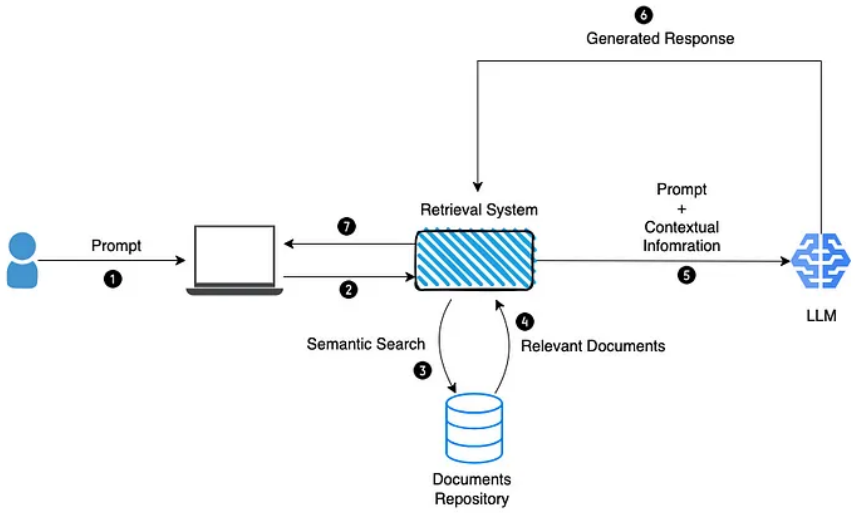


Figure 3: Funcionamiento de un RAG.

## 2.4 Generación de Respuestas con OpenAI

- **Función** `augment_with_retrieved_data`**:** Esta función tiene como objetivo enriquecer el contexto de la conversación, integrando información relevante recuperada de la base de datos vectorial (ChromaDB). A continuación, se describe su funcionamiento:

  1. La función recibe como parámetro `chat_prompt`, que es el contexto de la conversación hasta el momento, y lo complementa con información adicional obtenida de la base de datos vectorial.
  2. Se realiza una consulta a la base de datos vectorial mediante la función `query_chromadb`, que toma como parámetros la colección, la consulta (`query`), el modelo de embeddings y un parámetro `top_k` que determina la cantidad de resultados a recuperar.
  3. Los resultados de la consulta (`retrieval_results`) contienen documentos relevantes relacionados con la consulta. En este caso, se extraen los primeros `top_k` documentos de los resultados y se concatenan en una sola cadena de texto.

4. Se agrega al `chat_prompt` un mensaje con el rol de "system", que incluye la información relevante recuperada como parte del contexto.
5. Finalmente, la función devuelve el `chat_prompt` actualizado, ahora enriquecido con los datos recuperados.

- **Función** `generate_text_with_openai`**:** Una vez que el contexto ha sido enriquecido con la información relevante, se utiliza esta función para generar la respuesta utilizando el modelo de OpenAI:

1. La función recibe el mensaje del usuario (`message`) y lo utiliza para actualizar el `context`.
2. Se llama a `augment_with_retrieved_data`, que enriquece el contexto con los datos recuperados de la base de datos vectorial, utilizando el modelo de embeddings y los resultados de la consulta.
3. El mensaje del usuario se añade al contexto bajo el rol "user", manteniendo el historial de la conversación.
4. Se configura y utiliza la API de OpenAI (`OpenAI`) para generar una respuesta utilizando el modelo `gpt-3.5-turbo`. La función `chat.completions.create` envía el `context` al modelo, y este genera una respuesta en función de la información disponible.
5. La respuesta generada por el modelo se añade al contexto para mantener la coherencia de la conversación.
6. Finalmente, se imprime la interacción, mostrando tanto el mensaje del usuario como la respuesta generada por el modelo.

### 2.4.1 Elección de herramientas

- **API de OpenAI y GPT-3.5-Turbo:** Se eligió la API de OpenAI con el modelo `gpt-3.5-turbo` por su capacidad para generar respuestas rápidas, precisas y contextuales. Este modelo optimizado maneja bien las consultas semánticas dentro del agente RAG (Retrieve and Generate), lo que facilita una interacción fluida con los usuarios y mantiene las respuestas coherentes con el contexto de la conversación. El uso de `gpt-3.5-turbo` permite que el sistema se adapte a las respuestas de los usuarios y a los datos de la base de datos, lo que mejora la flexibilidad y precisión del agente.

- **Uso de Datos de la Base de Datos Vectorial ChromaDB:** Para las búsquedas y recuperación de respuestas, se decidió utilizar solo los datos almacenados en `ChromaDB`. Esta base de datos está diseñada para manejar embeddings de alta dimensión, lo que permite hacer búsquedas semánticas rápidas y precisas, encontrando las respuestas más relevantes. El uso exclusivo de `ChromaDB` asegura que las respuestas sean consistentes con los datos disponibles, evitando la generación de información errónea o irrelevante. Esto también permite mantener un control más preciso sobre las respuestas, mejorando la fiabilidad y precisión del sistema.

## 2.5 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.

## 2.6 Clasificador

### 2.6.1 Funcionamiento del clasificador

El clasificador está diseñado para categorizar preguntas relacionadas con el juego *Lost Ruins of Arnak* en distintas etiquetas como `rules`, `review`, `gameplay`, `stats` y `graph`. A continuación, se describen los pasos principales del pipeline:

1. **Preprocesamiento de datos:** Las preguntas (*prompts*) se convierten a minúsculas para normalizar el texto y evitar problemas de sensibilidad a mayúsculas.

2. **División del conjunto de datos:** Los datos se dividen en conjuntos de entrenamiento y prueba, utilizando una división estratificada para garantizar la representación proporcional de cada etiqueta.

3. **Vectorización con TF-IDF:**

   - Se utiliza el `TfidfVectorizer` para transformar el texto en vectores numéricos. Este método asigna un peso a cada palabra basado en su frecuencia relativa dentro del documento y en el corpus completo.
   - La vectorización resulta en una representación dispersa de los textos que captura su relevancia semántica sin ser dominada por palabras demasiado comunes.

4. **Entrenamiento del modelo:** Se entrena un modelo de regresión logística multinomial (`LogisticRegression`) para realizar la clasificación. Este modelo asigna probabilidades a las clases y selecciona la etiqueta con la mayor probabilidad.

5. **Evaluación:** Los datos de prueba se transforman con el vectorizador y se utilizan para evaluar el modelo, obteniendo métricas como precisión y recall.

### 2.6.2 Razones para elegir `TfidfVectorizer` y `LogisticRegression`

- **TfidfVectorizer:**
  - Captura la importancia relativa de las palabras en el texto, reduciendo el impacto de palabras comunes que no aportan información relevante (e.g., *stopwords*).
  - Genera una representación dispersa eficiente para documentos de texto.
  - Es especialmente útil para tareas de clasificación textual, donde es crucial identificar términos distintivos.

- **LogisticRegression:**
  - Es un modelo lineal que es eficiente y fácil de interpretar.
  - Soporta clasificación multiclase mediante el esquema multinomial y optimización con el solver `lbfgs`.
  - Tiene una robusta capacidad de generalización, particularmente en datasets bien vectorizados como los generados por `TfidfVectorizer`.

### 2.6.3 Interpretación de Resultados

### 2.6.4 Precisión General

La precisión general del modelo mide la proporción de predicciones correctas sobre el total de predicciones realizadas. Una precisión elevada indica que el modelo clasifica correctamente la mayoría de los ejemplos en el conjunto de prueba.

La métrica de *precisión* se utiliza para evaluar qué tan confiables son las predicciones positivas del modelo.

En este caso particular, donde se clasifican consultas sobre un juego en categorías específicas como `rules`, `review` y otras, la precisión es una métrica prioritaria porque asegura que las predicciones realizadas para una clase sean mayormente correctas. Por ejemplo, si el modelo clasifica una consulta como `rules`, queremos garantizar con alta confianza que dicha consulta realmente pertenece a esa categoría.

En nuestro caso, una precisión del 88% significa que 88 de cada 100 predicciones realizadas por el modelo fueron correctas.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| rules | 0.92 | 0.94 | 0.93 | 53 |
| review | 0.90 | 0.88 | 0.89 | 50 |
| gameplay | 0.88 | 0.91 | 0.89 | 55 |
| stats | 0.80 | 0.87 | 0.83 | 46 |
| graph | 0.85 | 0.81 | 0.83 | 48 |

## 2.7 Construcción de Random Forest Classifier

Para clasificar las preguntas de los usuarios, se utiliza el modelo `Random Forest`, un conjunto de árboles de decisión que mejora la precisión y reduce el sobreajuste. A continuación, se explica el proceso de construcción del modelo y las razones para elegir `Random Forest`:

- **Uso de `Random Forest`:** `Random Forest` es un modelo basado en varios árboles de decisión que se entrenan con diferentes muestras de datos. Esto ayuda a reducir el riesgo de sobreajuste y mejora la precisión del modelo. Cada árbol toma decisiones basadas en diferentes características del conjunto de datos, lo que hace que el modelo sea más robusto.
- **Ventajas de `Random Forest`:**
  - *Reduce el sobreajuste:* Al promediar los resultados de varios árboles, el modelo es menos propenso a sobreajustarse a los datos de entrenamiento.
  - *Maneja relaciones no lineales:* `Random Forest` puede capturar relaciones complejas entre las características sin suponer que son lineales.
  - *Es robusto con datos desbalanceados:* Puede manejar conjuntos de datos donde algunas categorías son más comunes que otras.
- **Preparación del dataset:** Se utiliza `TfidfVectorizer` para convertir las preguntas en características numéricas. Esto ayuda a identificar las palabras más relevantes para la clasificación.
- **Entrenamiento y evaluación:** El conjunto de datos se divide en dos partes: una para entrenar el modelo y otra para probarlo. Se utilizan métricas como precisión y F1-score para evaluar el rendimiento del modelo.

La visualización a continuación corresponde al primer árbol del modelo Random Forest. Dada la complejidad y profundidad del árbol, puede resultar difícil apreciarlo en detalle dentro de un único gráfico.

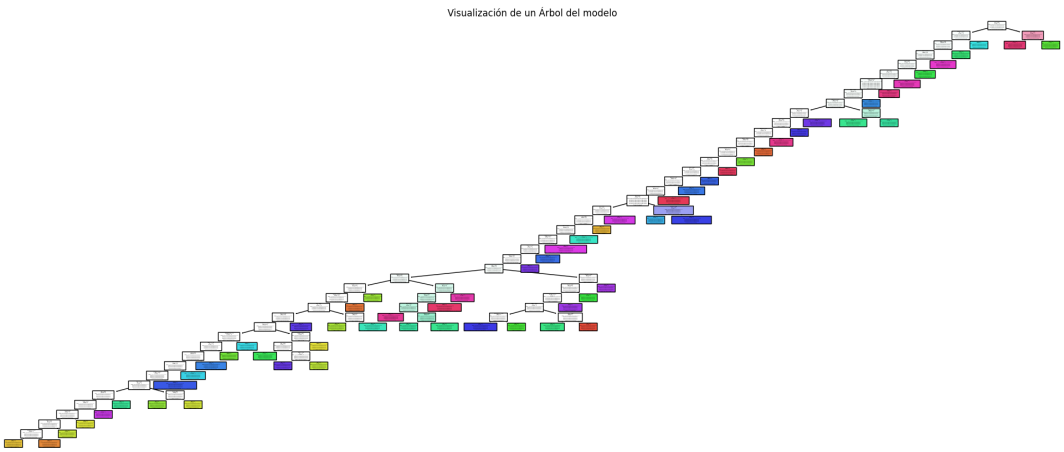


Figure 4: Primer arbol del modelo.

### 2.7.1 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.

## 2.8 Selección de Métrica y Modelo de Clasificación

Para evaluar el desempeño de los modelos utilizados, se empleó la métrica de **similitud de coseno**, que mide la similitud entre los vectores de embeddings generados para las respuestas de los modelos y las respuestas correctas.

La métrica de similitud de coseno fue elegida porque permite evaluar qué tan cercanas son las respuestas generadas a las correctas en un espacio semántico, independientemente de su magnitud. Esto es especialmente útil al trabajar con embeddings, donde se busca medir la coherencia conceptual entre textos en lugar de evaluar una coincidencia exacta.

### 2.8.1 Resultados y Selección del Modelo

Los resultados promedio obtenidos para cada modelo fueron los siguientes:

- Similitud Promedio del **LLM**: 0.5405
- Similitud Promedio del **Random Forest**: 0.3715

A partir de estos resultados, se observa que el modelo basado en LLM supera al modelo Random Forest en términos de similitud semántica con las respuestas correctas. Este resultado indica que el LLM tiene un mejor desempeño al comprender las preguntas y generar respuestas más alineadas con las esperadas.

Por lo tanto, se selecciona el modelo basado en **LLM** para este problema debido a su capacidad superior para capturar el significado subyacente de las preguntas y ofrecer respuestas más precisas en términos semánticos. Aunque el modelo Random Forest muestra un desempeño razonable, su enfoque basado en características vectorizadas de texto tiene limitaciones para capturar la riqueza semántica de las preguntas y respuestas en comparación con el modelo LLM.

## 2.9 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 - Agente

## 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.

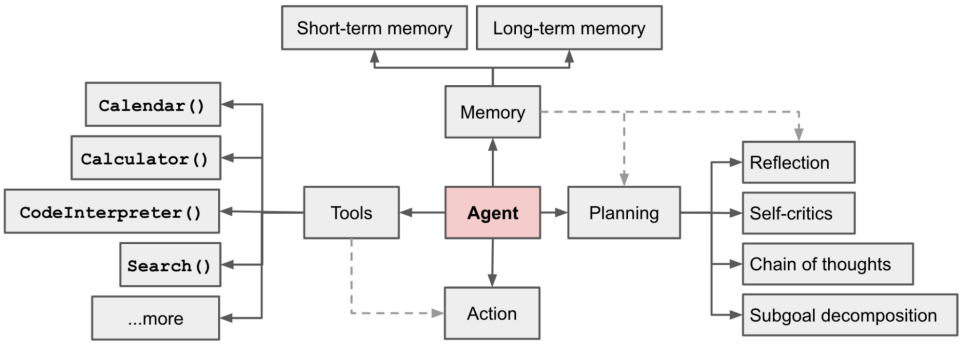


Figure 5: Pipeline de un agente.

## 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.

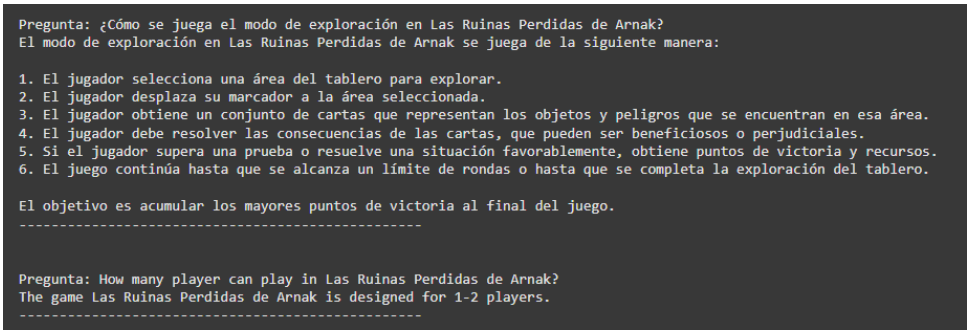


Figure 6: Resultado de ejemplo del agente.

### 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? | Las mecánicas principales del juego Las Ruinas Perdidas de Arnak incluyen:<br>• **Exploración:** el jugador explora áreas desconocidas del tablero para obtener objetos antiguos y tomos.<br>• **Coleccionismo:** el jugador recopila objetos antiguos y tomos para obtener puntos de victoria.<br>• **Desbloqueo:** el jugador desbloquea áreas del tablero para acceder a nuevos objetos y tomes.<br>• **Gestión de recursos:** el jugador debe gestionar sus recursos (como energía y movimientos) para explorar y coleccionar objetos.<br>El juego combina estas mecánicas con un sistema de cartas que agrega una capa de estrategia y sorpresa al juego. |
| 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]