



Procesamiento del Lenguaje Natural

Trabajo Practico Nº 2

INFORME

Carrera: TUIA

Alumno: Donnarumma, César Julián

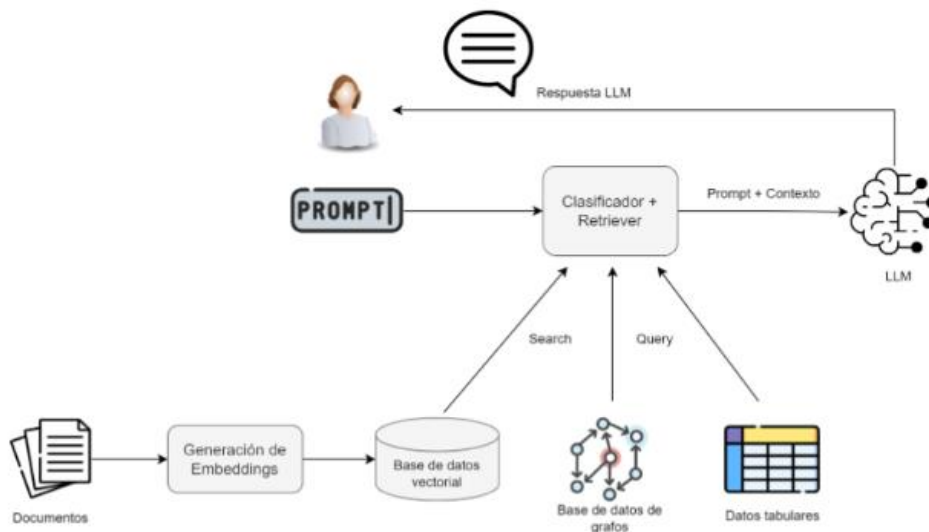
Ciclo: 2023

Ejercicio 1 – RAG:

Crear un chatbot experto en un tema a elección, usando la técnica RAG (Retrieval Augmented Generation).

Como fuentes de conocimiento se utilizarán al menos las siguientes fuentes:

- Documentos de texto.
- Datos numéricos en formato tabular (por ej., Dataframes, CSV, sqlite, etc).
- Base de datos de grafos (Online o local).



El sistema debe poder llevar a cabo una conversación en lenguaje español. El usuario podrá hacer preguntas, que el chatbot intentará responder a partir de datos de algunas de sus fuentes. El asistente debe poder clasificar las preguntas, para saber qué fuentes de datos utilizar como contexto para generar una respuesta.

Requerimientos generales:

- Realizar todo el proyecto en un entorno Google Colab.
- El conjunto de datos debe tener al menos 100 páginas de texto y un mínimo de 3 documentos.
- Realizar split de textos usando Langchain (RecursiveTextSearch, u otros métodos disponibles). Limpiar el texto según sea conveniente.
- Realizar los embeddings que permitan vectorizar el texto y almacenarlo en una base de datos ChromaDB.
- Los modelos de embeddings y LLM para generación de texto son a elección
- Para el desarrollo del "Clasificador" es posible utilizar diversas técnicas aprendidas en la materia, por ejemplo en Unidad 3 y Unidad 6.

Ejercicio 2 – Agentes:

Realice una investigación respecto al estado del arte de las aplicaciones actuales de agentes inteligentes usando modelos LLM libres.

Plantee una problemática a solucionar con un sistema multiagente. Defina cada uno de los agentes involucrados en la tarea. Es importante destacar con ejemplos de conversación, la interacción entre los agentes.

Realice un informe con los resultados de la investigación y con el esquema del sistema multiagente, no olvide incluir fuentes de información.

Opcional: Resolución con código de dicho escenario.

Informe Ejercicio 1 - RAG:

El **tema a elección** en el cual se especializa el chatbot creado que utiliza RAG es sobre el videojuego Resident Evil. En particular sobre 5 juegos de toda la saga (ya que son demasiados los juegos disponibles): Resident Evil 1 Remake, Resident Evil 2 (con Leon y con Claire), Resident Evil 3, Resident Evil 4 Remake, Resident Evil: Code Veronica. Entonces en total contamos con un corpus de documentos que consta de seis guías y un total de 114 páginas.

Se almaceno en cada una de las bases de datos solicitadas distinto tipo de información sobre el mismo tema:

1. **Base de datos vectorial:** contiene información de guía de los juegos mencionados. Para obtener la información de las guías se construyeron pdfs con datos sacados de internet (https://residentevil.fandom.com/es/wiki/Resident_Evil_Wiki y <http://meristation.com/>) ya que no se encontraron guías “oficiales” de autor publicadas en libros o pdfs. Si bien es cierto que se podría haber extraído la misma a través de webscrapping (tema dado en el curso) era necesario cumplir con el requerimiento de la consigna del ejercicio de 100 páginas de texto (lo cual no sería medible a través de dicha técnica).
2. **Base de datos de grafos:** contiene información de ficha de cada uno de los juegos como ser nombre, saga, genero, país de origen, director, productor, lugar donde ocurre, desarrollador, cantidad de jugadores, juego sucesor, fecha de lanzamiento, entre otras. Esta información fue sacada de la base de datos online **Wikidata**.
3. **Base de datos tabular:** contiene información de puntajes de críticas de páginas online. Es una base de datos muy pequeña, creada por mí consultando en el buscador varias páginas online y cargando las puntuaciones de las críticas (utilice **Pandas** para crear un *DataFrame*).

Entonces dado esta distribución el usuario podrá consultar tres bases de datos que determinan **tres categorías:** *información de guía del juego (1), información de ficha (2), información de puntuaciones de críticas (3).*

Código de Bases de Datos:

1. Base de datos vectorial:

A posteriori de haber instalado las librerías necesarias comienza la preparación de los datos para la base de datos vectorial.

En primer lugar se extraen de los pdfs los textos con **pdfplumber**.

```
import pdfplumber as pp

# Nombre de los juegos
juegos = ['resident_evil_1_remake', 'resident_evil_2_claire',
          'resident_evil_2_leon', 'resident_evil_3',
          'resident_evil_4_remake', 'resident_evil_code_veronica']

# Aca se almacenaran los textos de los pdf
textos = {}

# Extraccion de los pdf: recorremos cada uno de los nombre de los juegos
for guia in juegos:
    # Abrimos el pdf
    with pp.open(f'{guia}.pdf') as pdf:
        texto = ""
        # Por cada pagina
        for pagina in pdf.pages:
            # Extraemos el texto
            texto += pagina.extract_text() + "\n"
        # Una vez que extraimos cada pagina guardamos el texto entero con la clave del nombre correspondiente al juego
        textos[guia]=texto
```

Como el texto no corresponde ni a un libro, ni a ninguna fuente publicada oficialmente sino que es el mismo texto tal y como estaba en las páginas webs mencionadas anteriormente no contiene información (como numero de página, de capítulo, o el nombre del libro) que luego de la extracción estuviera “ensuciando” los datos. Por lo que no se requirió hacer una limpieza de ningún tipo.

```
print(textos['resident_evil_1_remake'])
```

Guía de Resident Evil 1 Remake:
En esta guía encontrarás la manera más sencilla de pasar el juego.
La guía estará escrita en fuente normal y las cinemáticas en cursiva.
Puedes elegir jugar con Chris Redfield o Jill Valentine, la guía principal será escrita por la parte de Jill, cada parte que sea diferente entre ellos, se pondrá en paréntesis, ya que al ser muy pequeñas las diferencias y el mismo camino no tiene caso reescribir todo.
El camino de cada uno tiene algunas historias y final con ciertas diferencias, las imágenes que se mostraran aplican para los dos, ya que toman el mismo camino, solo con algunas pequeñas diferencias.
Con Chris es mayor la dificultad, es más certero con los zombis, pero tendrá menos municiones y menor cantidad de hierbas, el lanzagranadas solo lo encontraras con Jill.
Para tomar la escopeta con Jill será más sencillo pues te salvara Barry y no será necesario ir por la escopeta vieja, pero con Chris, si será necesario, pues nadie lo salvara.
Jill para defenderse usara cuchillos y un teaser, Chris usara granadas.
Chris llevara de cajón el encendedor, Jill la ganzúa.
En la historia de Jill, su ayuda en muchas ocasiones será Barry, y Chris pasa todo el juego

El siguiente paso es partir cada una de las guías en chunks.

```
[7] from langchain.text_splitter import RecursiveCharacterTextSplitter

# Creamos el text splitter
text_splitter = RecursiveCharacterTextSplitter(
    # Tamaño del chunk
    chunk_size=750,
    # Solapamiento entre chunks
    chunk_overlap=250
)
```

```
[8] # Diccionario donde vamos a guardar cada una de las guías cortadas.
    textos_cortados = {}

    # Cortamos una a una las guías
    for nombre, guía in textos.items():
        textos_cortados[nombre] = text_splitter.split_text(guía)
```

La elección de los parámetros *chunk_size* y *chunk_overlap* no fue aleatoria sino que sale de un proceso iterativo de ir eligiendo distintos valores, almacenando las guías en la base de datos vectorial y haciendo distintas consultas para ver viendo de qué modo se obtenían los mejores resultados. En mi experiencia durante el tiempo que pude dedicarle al presente TP estos valores fueron los que dieron mejores resultados.

La siguiente imagen demuestra como queda partido el mismo texto mostrado anteriormente.

```
[37] for i, txt in enumerate(textos_cortados['resident_evil_1_remake']):
    # Imprimimos la longitud de la cadena, y luego el trozo de texto (chunk)
    print(f'Split: {i}/{len(textos_cortados["resident_evil_1_remake"])}')
    print(f'Len: {len(txt)}\nTexto:\n{txt}')
    print('')
```

Split: 0/128
Len: 731
Texto:
Guía de Resident Evil 1 Remake:
En esta guía encontrarás la manera más sencilla de pasar el juego.
La guía estará escrita en fuente normal y las cinemáticas en cursiva.
Puedes elegir jugar con Chris Redfield o Jill Valentine, la guía principal será escrita por la parte de Jill, cada parte que sea diferente entre ellos, se pondrá en paréntesis, ya que al ser muy pequeñas las diferencias y el mismo camino no tiene caso reescribir todo.
El camino de cada uno tiene algunas historias y final con ciertas diferencias, las imágenes que se mostraran aplican para los dos, ya que toman el mismo camino, solo con algunas pequeñas diferencias.
Con Chris es mayor la dificultad, es más certero con los zombis, pero tendrá menos municiones

Split: 1/128
Len: 667
Texto:
se mostraran aplican para los dos, ya que toman el mismo camino, solo con algunas pequeñas diferencias.
Con Chris es mayor la dificultad, es más certero con los zombis, pero tendrá menos municiones y menor cantidad de hierbas, el lanzagranadas solo lo encontraras con Jill.
Para tomar la escopeta con Jill será más sencillo pues te salvara Barry y no será necesario ir por la escopeta vieja, pero con Chris, si será necesario, pues nadie lo salvara.
Jill para defenderse usara cuchillos y un teaser, Chris usara granadas.
Chris llevara de cajón el encendedor, Jill la ganzúa.
En la historia de Jill, su ayuda en muchas ocasiones será Barry, y Chris pasa todo el juego

Split: 2/128
Len: 724
Texto:

Una vez que se tienen todos textos cortados, es necesario realizarle los embeddings para meterlos **chromadb**.

```
[10] import chromadb

    # Creamos el objeto de la base de datos
    bd_chroma = chromadb.Client()
```

Como **modelo de embedding** se utiliza **Universal Sentence Encoder Multilingual**. La elección se basa nuevamente en una cuestión empírica de ir probando distintos modelos (tanto multilinguaje como no) y haciendo consultas para ver con cual se daban mejores resultados (claro está que en el caso de los que no eran multilinguaje se obtuvieron resultados bastante peores, pero al ser la primera experiencia había que probarlo). En **chromadb** se pueden utilizar como embedding modelos de la librería **Sentece Transformers** y en particular previo a la

elección final se probaron varios de los disponibles ahí. La siguiente imagen muestra cuales fueron los que se probaron.

Model Name	Performance Sentence Embeddings (14 Datasets) ⓘ	Performance Semantic Search (6 Datasets) ⓘ	Avg. Performance ⓘ	Speed ⓘ	Model Size ⓘ
all-mpnet-base-v2 ⓘ	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1 ⓘ	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1 ⓘ	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2 ⓘ	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1 ⓘ	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2 ⓘ	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1 ⓘ	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2 ⓘ	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2 ⓘ	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2 ⓘ	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2 ⓘ	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1 ⓘ	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2 ⓘ	60.18	27.35	43.77	4000	480 MB

Al usar como embedding un modelo ajeno a **Sentece Transformers** fue necesario modificar la función de embedding para poder utilizar la función **get_collection()** mas tarde.

```
[12] import tensorflow_hub as hub
import numpy as np
import tensorflow_text
from chromadb import Documents, EmbeddingFunction, Embeddings

# Como vamos a utilizar universal sentence encoder multilenguaje como embedding es necesario modificar la funcion de embedding
# de ChromaDB (utilizada mas adelante a la hora de hacer get_collection)
class MyEmbeddingFunction(EmbeddingFunction):

    def __init__(self):
        # Variable de instancia
        self.embed = hub.load("https://www.kaggle.com/models/google/universal-sentence-encoder/frameworks/TensorFlow2/versions/multilingual/versions/2")

    def __call__(self, input: Documents) -> Embeddings:
        embeddings = self.embed(input)
        return embeddings

funcion_embedding = MyEmbeddingFunction()
```

Chromadb permite tener distintas colecciones donde almacenar embeddings y al tener distintas guías pertenecientes a distintos juegos, por una cuestión de hacer la búsqueda un poco más específica enfocada solamente en el juego que el usuario consulte, se crea una colección por entrega. Previamente se generan los embeddings. Por último se almacena en la colección tanto los embeddings, como los chunks, las ids y el nombre del juego como metadata.

```
[13] # Aca vamos a guardar los embeddings que generemos
embeddings_textos_cortados = {}

# Por cada guia con split ya hecho
for nombre, guia in textos_cortados.items():
    # Creamos los embeddings del documento
    embeddings_textos_cortados[nombre] = funcion_embedding(textos_cortados[nombre]).numpy().tolist()
    # Creamos una coleccion perteneciente a esa guia
    coleccion = bd_chroma.create_collection(name=f"{nombre}", metadata={"hnsw:space": "cosine"})
    # Agregamos:
    coleccion.add(
        embeddings=embeddings_textos_cortados[nombre],
        # Los splits
        documents=guia,
        # Como metadata a que juego pertenece dicho split
        metadatas = [ {'Guia': nombre} for _ in range(len(guia)) ],
        # Como id que numero de chunk es dicho split
        ids=[f'Momento del juego {str(x)}' for x in range(len(guia))]
    )
```

Con respecto a esta base de datos solo queda crear la función ***busqueda_semantica(juego, query)*** que realiza la búsqueda semántica acorde al juego y a la consulta del usuario. Simplemente con el nombre del juego se busca en la colección correspondiente a la consulta y se realiza la consulta en si trayendo los chunks más cercanos. Al final se formatea el texto que es lo que se devuelve para ser utilizado en el contexto del LLM.

```
[182] def busqueda_semantica(juego, query):

    # Cantidad de resultados que traera
    cantidad_resultados=4
    # A partir del nombre del juego se obtiene la coleccion correspondiente
    coleccion = bd_chroma.get_collection(name=juego, embedding_function=funcion_embedding)
    # Búsqueda en si.
    resultado_query = coleccion.query(
        query_texts=query,
        n_results=cantidad_resultados,
        # where={"Guia": {"$eq": "resident_evil_3"}},
        # where_document={"$contains": "leon"},
        # Devuelve distancia (coseno seteada arriba), metadata (nombre del juego) y los documentos mas cercanos (chunks correspondientes)
        include=['distances', 'metadatas', 'documents']
    )
    # Formateo de los resultados para luego pasar al contexto del LLM.
    resultado_limpio=f'Guia: {juego}\n\n'
    for i in range(cantidad_resultados):
        resultado_limpio += f'{resultado_query["ids"][0][i]}:\n'
        resultado_limpio += f'Similitud coseno: {resultado_query["distances"][0][i]}\n'
        resultado_limpio += f'{resultado_query["documents"][0][i]}:\n\n'

    # Devuelve el resultado formateado en un string
    return resultado_limpio
```

La siguiente imagen demuestra que es lo que arroja ***busqueda_semantica()*** cuando la utilizamos.


```

23] # ----- CONTROL DE CONSULTA VECTORIAL -----
print('Control de Consulta Vectorial:')
print('')
pregunta = 'Con que arma comienza Chris?'
print(f'Pregunta: {pregunta}')
print('')
print(busqueda_semantica('resident_evil_1_remake', pregunta))
# ----- FIN CONTROL DE CONSULTA VECTORIAL -----

```

Control de Consulta Vectorial:

Pregunta: Con que arma comienza Chris?

Guia: resident_evil_1_remake

Momento del juego 1:
Similitud coseno: 0.6415958404541016
se mostraran aplican para los dos, ya que toman el mismo camino, solo con algunas pequeñas diferencias.
Con Chris es mayor la dificultad, es más certero con los zombies, pero tendrá menos municiones y menor cantidad de hierbas, el lanzagranadas solo lo encontraran con Jill.
Para tomar la escopeta con Jill será más sencillo pues te salvara Barry y no será necesario ir por la escopeta vieja, pero con Chris, si será necesario, pues nadie lo salvara.
Jill para defenderse usara cuchillos y un teaser, Chris usara granadas.
Chris llevara de cajón el encendedor, Jill la ganzúa.
En la historia de Jill, su ayuda en muchas ocasiones será Barry, y Chris pasa todo el juego:

Momento del juego 84:
Similitud coseno: 0.6451029181480408
separar.
(Al terminar la pelea, Chris se encontrara con Rebecca, le dirá que Richard murió, entraran en el cuarto de medicamentos, y tras curarlo empezara a llorar, Chris la consuela y le dice que sigan investigando, que se cuide y la deja.
Al salir de la galería escucha disparos y encuentra Wesker "Chris sigues vivo" "Lo mismo digo, donde esta Jill? pensé que estabas con ella" le recrimina "Si, pues no separamos, quiero que sigas investigando la otra mansión" le dice Wesker un poco sospechoso "Si claro" responde Chris).
Cruza el jardín, cuidado que antes de subir por el ascensor en ese patio hay dos perros, y regresa a la mansión.
Al llegar al cobertizo, encontrarás una nota de Barry/Wesker "utiliza esto para protegerte de:

Momento del juego 4:
Similitud coseno: 0.6600323662373433

2. Base de datos tabular:

La base de datos tabular es una muy pequeña base de datos construida por mí en base a visitas a páginas web especializadas en videojuegos consultando página por página las críticas de cada juego y poniendo la nota que obtuvo el mismo. Se crea con **pandas** un *DataFrame* con estos datos que luego se exporta como csv. Más adelante se vuelve a cargar simplemente para emular la situación de utilizar un csv descargado de otro lado.

```

[17] import pandas as pd

# Pagina de las cual se sacaron los puntajes
paginas = [ 'juego', 'meristation', 'ign', 'vandal', 'gamespot', 'hobbyconsolas', '3djuegos' ]

# Creacion del df
puntajes = pd.DataFrame(columns=paginas)

# Informacion de puntajes por juego
resident_evil_remake = ['resident evil 1 remake', '9.5/10', '9/10', '8.5/10', '8.9/10', '73/100', '9.5/10']
resident_evil_2_leon_and_claire = ['resident evil 2', '9.5/10', '9.3/10', '9.2/10', '8.9/10', '98/100', '9/10']
resident_evil_3 = ['resident evil 3', '8.5/10', '9.4/10', '9.3/10', '8.8/10', '92/100', '9/10']
resident_evil_4_remake = ['resident evil 4 remake', '9/10', '10/10', '9/10', '10/10', '92/100', '10/10']
resident_evil_code_veronica = ['resident evil code veronica', '9/10', '9/10', '9.5/10', '9.5/10', '95/100', '8/10']

puntajes = pd.concat([puntajes, pd.DataFrame([resident_evil_remake, resident_evil_2_leon_and_claire, resident_evil_3, resident_evil_4_remake, resident_evil_code_veronica], columns=puntajes.columns)], ignore_index=True)

nombre_archivo_csv = "resident_evil_puntajes.csv"

# Utilizar el método to_csv() para exportar el DataFrame a un archivo CSV
puntajes.to_csv(nombre_archivo_csv, index=False)

```

```

[18] import pandas as pd

puntajes = pd.read_csv(filepath_or_buffer='resident_evil_puntajes.csv')

puntajes.set_index('juego', inplace=True)

```

La siguiente imagen muestra cómo queda constituido el *DataFrame*.

[19] puntajes

	meristation	ign	vandal	gamespot	hobbyconsolas	3djuegos
juego						
resident evil 1 remake	9.5/10	9/10	8.5/10	8.9/10	73/100	9.5/10
resident evil 2	9.5/10	9.3/10	9.2/10	8.9/10	98/100	9/10
resident evil 3	8.5/10	9.4/10	9.3/10	8.8/10	92/100	9/10
resident evil 4 remake	9/10	10/10	9/10	10/10	92/100	10/10
resident evil code veronica	9/10	9/10	9.5/10	9.5/10	95/100	8/10

Por último se crea la función **busqueda_tabular(juego, pagina)** que de acuerdo al juego y la página busca la puntuación de la crítica en el *DataFrame* y genera un string que es lo que se le pasa luego al contexto del LLM. Como el *DataFrame* es muy chico, en un primer instante se le intento pasar el csv completo al contexto del LLM y que el solo saque la información de acuerdo a la consulta pero por algún motivo alucinaba e inventaba la información así que se acoto el contexto únicamente a lo solicitado por el usuario.

```
[ ] def busqueda_tabular(juego, pagina):
    resultado_tabular = puntajes.loc[juego, pagina]

    texto_final = f"{pagina}: {resultado_tabular}"

    return texto_final
```

El siguiente seria el resultado de utilizar la función **busqueda_tabular()**

```
[22] # ----- CONTROL DE CONSULTA TABULAR -----
print('Control de Consulta Tabular:')
print('')
juego = 'resident evil 3'
pagina = 'meristation'
print(busqueda_tabular(juego, pagina))
# ----- FIN DE CONTROL DE CONSULTA VECTORIAL -----

Control de Consulta Tabular:

meristation: 8.5/10
```

3. Base de datos de grafos:

Como mencione anteriormente a la base de datos de grafos no la cree yo sino que se utiliza **Wikidata** para extraer información de los juegos. El primer paso, que no resulto para nada sencillo, fue aprender a construir las consultas **SPARQL** en base a ver en la web el diagrama de los elementos (cada uno de los juegos) y las propiedades disponibles (para obtener su valor).

[Página principal](#)
[Portal de la comunidad](#)
[Café](#)
[Crear un elemento nuevo](#)
[Cambios recientes](#)
[Elemento aleatorio](#)
[Query Service](#)
[Cercanos](#)
[Ayuda](#)
[Donaciones](#)
[Datos lexicográficos](#)
[Crear un lexema nuevo](#)
[Cambios recientes](#)
[Lexema aleatorio](#)
[Herramientas](#)
[Lo que enlaza aquí](#)
[Cambios relacionados](#)
[Páginas especiales](#)
[Enlace permanente](#)
[Información de la página](#)
[URI de concepto](#)
[Citar esta página](#)
[Obtener URL acortado](#)
[Download QR code](#)

Elemento

Discusión

Leer

Ver historial

Resident Evil (remake) (Q622336)

videojuego de 2002

editar

En más idiomas

Configurar

Idioma	Etiqueta	Descripción	También conocido como
español	Resident Evil (remake)	videojuego de 2002	
inglés	Resident Evil	2002 survival horror video game remake	REmake Biohazard
inglés estadounidense	Ninguna etiqueta definida	Ninguna descripción definida	
galés	Ninguna etiqueta definida	Ninguna descripción definida	

Todos los idiomas ingresados

Declaraciones

Instancia de

videojuego

2 referencias

editar

remake de videojuego de

Resident Evil

0 referencias

añadir referencia

añadir valor

basado en

Resident Evil

tiene el rol de

remake de videojuego

0 referencias

añadir referencia

añadir valor

serie

Resident Evil

0 referencias

añadir referencia

añadir valor

sucedido por

Resident Evil 2

0 referencias

añadir referencia

añadir valor

género

survival horror

editar

Las consultas se crearon primero en **Wikidata Query Service** y luego se guardaron en variables en forma de string. Se creó una query por juego, lo que generó que en su mayor parte el código de las consultas sea algo repetitivo. Si bien estoy al tanto de que es mejorable y podría utilizar variables, para reutilizar un mismo bloque general era necesario aprender a utilizar el OPTIONAL (creo) pero aprender **SPARQL** tomó demasiados días de la elaboración del TP, por lo que necesitaba avanzar para poder llegar en buenos términos al día de la entrega.

En la siguiente imagen se puede ver un ejemplo de la query de un juego.

```
[21] query_resident_evil_1_remake = '''

SELECT

# variables a mostrar
?nombre ?saga ?genero ?editorial ?desarrollador
(GROUP_CONCAT(DISTINCT ?distribuidoraJuego; SEPARATOR=", ") AS ?distribuidorasJuego)
?paisDeOrigen ?director ?productor
(GROUP_CONCAT(DISTINCT ?plataformaJuego; SEPARATOR=", ") AS ?plataformasJuego)
(MIN(?fecha) AS ?fechaDeLanzamiento)
?cantidadJugadores
(GROUP_CONCAT(DISTINCT ?idiomaJuego; SEPARATOR=", ") AS ?idiomasDelJuego)
?paisDondeOcurre
?juegoSucesor

WHERE

{
# el ";" es para compartir sujeto, el "." indica que viene otro triple completo adelante, la "," comparte sujeto y predicado
?juego wdt:P31 wd:Q7889; # juego que es instancia de videojuego
wdt:P144 wd:Q1248594; # basado en resident evil 1 (el unico es el remake)
rdfs:label ?nombre; # el nombre literal
# [] es para meter no repetir codigo, por ej: en el caso de abajo seria
# ?juego pertenece a la saga y la saga tiene la etiqueta ?saga
wdt:P170 [rdfs:label ?saga]; # perteneciente a la saga
wdt:P136 [rdfs:label ?genero]; # genero del juego
wdt:P123 [rdfs:label ?editorial]; # editorial
wdt:P495 [rdfs:label ?paisDeOrigen]; # pais de origen
wdt:P407 ?idioma; # idiomas en los que esta
wdt:P577 ?fecha; # fecha de lanzamiento (solo la primera)
wdt:P57 [rdfs:label ?director]; # director
wdt:P162 [rdfs:label ?productor]; # productor
wdt:P750 ?distribuidora; # distribuidora del juego
wdt:P840 [rdfs:label ?paisDondeOcurre]; # donde ocurre
wdt:P178 [rdfs:label ?desarrollador]; # desarrollador del juego
wdt:P400 ?plataforma; # plataformas para las que salio
wdt:P404 [rdfs:label ?cantidadJugadores]; # cantidad de jugadores
wdt:P156 [rdfs:label ?juegoSucesor]; # juego que sigue

SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],es".
# Si vamos a utilizar GROUP CONTACT es necesario agregar en el SERVICE los tríos por los que queremos agrupar y luego concatenar
?idioma rdfs:label ?idiomaJuego.
?distribuidora rdfs:label ?distribuidoraJuego.
?plataforma rdfs:label ?plataformaJuego.
}

# Estos filtros son para que traiga solo etiquetas en español, ya que de otro modo traería las etiquetas en todos los idiomas

FILTER(LANG(?nombre) = "es").
FILTER(LANG(?saga) = "es").
FILTER(LANG(?genero) = "es").
FILTER(LANG(?editorial) = "es").
FILTER(LANG(?paisDeOrigen) = "es").
FILTER(LANG(?director) = "es").
FILTER(LANG(?productor) = "es").
FILTER(LANG(?paisDondeOcurre) = "es").
FILTER(LANG(?desarrollador) = "es").
FILTER(LANG(?cantidadJugadores) = "es").
FILTER(LANG(?juegoSucesor) = "es").

}

# Es necesario meter en el GROUP BY todas las variables del SELECT

GROUP BY ?juego ?nombre ?saga ?genero ?editorial ?paisDeOrigen ?fechas ?director ?productor ?paisDondeOcurre ?desarrollador ?cantidadJugadores ?juegoSucesor

...

```

Lo último sería crear la función ***consulta_sparql(query)*** que hace una solicitud GET a la página de consultas con la query pasada por parámetro, recibe un json con la respuesta de la consulta, la formatea y la devuelve, o devuelve un mensaje de error.

```
[22] import requests

def consulta_sparql(query):

    # URL donde se enviara la consulta
    url = "https://query.wikidata.org/sparql"

    # Configuración de la consulta SPARQL
    headers = {
        # Valor generalmente asociado a navegador web Chrome. Indica al servidor que la solicitud proviene de un navegador (aunque sea un script)
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36',
        # Queremos recibir un json
        'Accept': 'application/json'
    }

    params = {
        # Consulta SPARQL a consultar
        'query': query,
        # Recibimos un json.
        'format': 'json'
    }

    # Realización de la solicitud GET.
    response = requests.get(url, headers=headers, params=params)

    # Si la respuesta fue exitosa
    if response.status_code == 200:

        # Verificamos el json
        if response.json():

            # Creamos el string para el contexto con los resultados
            resultado = response.json()['results']['bindings'][0]
            respuesta_wikidata = ''
            for key in resultado.keys():
                respuesta_wikidata += f'{key}: {resultado[key]["value"]}\n'

            return respuesta_wikidata

        # Sino
    else:
        print("Error al realizar la consulta. Código de estado:", response.status_code)
        # Devolvemos nulo
        return None
```

Lo siguiente es un ejemplo del retorno de *consulta_sparql()*.

```
[26] # ----- CONTROL DE CONSULTA DE GRAFOS -----
print('Control de Consulta de Grafos:')
print('')
print(consulta_sparql(query_resident_evil_1_remake))
# ----- FIN DE CONTROL DE CONSULTA DE GRAFOS -----

Control de Consulta de Grafos:

nombre: Resident Evil (remake)
saga: Resident Evil
genero: survival horror
editorial: Capcom
paisDeOrigen: Japón
director: Shinji Mikami
productor: Hiroyuki Kobayashi
paisDondeOcurre: Estados Unidos
desarrollador: Capcom
cantidadJugadores: un jugador
juegosucesor: Resident Evil 2
distribuidorasJuego: Microsoft Store, Steam, PlayStation Store, Nintendo eShop
plataformasJuego: Microsoft Windows, Wii, PlayStation, PlayStation 3, Xbox 360, Nintendo GameCube, PlayStation 4, Xbox One, Nintendo Switch
fechaDeLanzamiento: 2002-03-22T00:00:00Z
idiomasDelJuego: francés, alemán, italiano, japonés, chino simplificado, chino tradicional, español, inglés
```

Código del funcionamiento general del chatbot y consultas del usuario:

Una vez que ya tenemos listo todo lo relacionado a las bases de datos es momento de pasar a la parte relacionada al LLM y a las consultas que hace el usuario.

Como base para la resolución de esta parte se toma el último ejercicio de la Unidad 6 de la materia donde se utiliza el modelo *Zephyr-7B-β* (versión “fine-tuneada” de *mistralai/Mistral-7B-v0.1*) disponible en HuggingFace y el cual se utiliza por medio de la API de la página.

Previo a su utilización intente cargar varios conversacionales en local (en Colab con el pipeline) pero muchos no pude porque aparentemente las necesidades de recursos eran mayores a las que se tenían y los que si pude y manejaban el español (como DialogGPT) daban respuestas bastante malas cuando los probé. Más tarde vi el ejercicio que tome como base (recordé la API), leí que (a pesar de que estaba en la sección de generación de texto y no conversacional de HuggingFace) se puede utilizar para chat por cómo fue “fine-tuneado” y además al margen de que su idioma sea principalmente el inglés funciona bien en Español. Por lo que luego de correr el ejemplo y probarlo decidí utilizarlo para el TP.

Hubo dos funciones que se mantienen tal cual estaban. Una es ***plantilla_instruccion_zephyr(messages)*** que utiliza *Jinja* para formatear la plantilla a partir del mensaje recibido.

```
# Funcion que define la plantilla jinja
def plantilla_instruccion_zephyr(messages, add_generation_prompt=True):
    template_str = "{% for message in messages %}"
    template_str += "{% if message['role'] == 'user' %}"
    template_str += "<|user|>{{ message['content'] }}</s>\n"
    template_str += "{% elif message['role'] == 'assistant' %}"
    template_str += "<|assistant|>{{ message['content'] }}</s>\n"
    template_str += "{% elif message['role'] == 'system' %}"
    template_str += "<|system|>{{ message['content'] }}</s>\n"
    template_str += "{% else %}"
    template_str += "<|unknown|>{{ message['content'] }}</s>\n"
    template_str += "{% endif %}"
    template_str += "{% endfor %}"
    template_str += "{% if add_generation_prompt %}"
    template_str += "<|assistant|>\n"
    template_str += "{% endif %}"
    # Crear un objeto de plantilla con la cadena de plantilla
    template = Template(template_str)
    # Renderizar la plantilla con los mensajes proporcionados
    return template.render(messages=messages, add_generation_prompt=add_generation_prompt)
```

Y la otra es ***conexión_llm(prompt)*** que se conecta al modelo via API. En esta última se mantienen la misma cantidad de *max_new_tokens*, se configura *temperature* (va de 0 a 100) en 0.01 con el objetivo de que sea lo más determinista posible y evitar la aleatoriedad de las respuestas dejando de lado las posibilidades de creatividad en las respuestas, *top_k* y *top_p* se setearon en **None** (por defecto) ya que luego de probar y probar no pude notar diferencias con las distintas configuraciones.

```

# Aquí hacemos la llamada al modelo
def conexion_llm(prompt: str, max_new_tokens: int = 768) -> None:
    try:
        # Tu clave API de Hugging Face
        api_key = 'hf_msZFscnUZatCCKRtS0zGwiVNDpCHZzGaKV'

        # URL de la API de Hugging Face para la generación de texto
        api_url = "https://api-inference.huggingface.co/models/HuggingFaceH4/zephyr-7b-beta"

        # Cabeceras para la solicitud
        headers = {"Authorization": f"Bearer {api_key}"}

        # Datos para enviar en la solicitud POST
        # Sobre los parámetros: https://huggingface.co/docs/transformers/main\_classes/text\_generation
        data = {
            "inputs": prompt,
            "parameters": {
                "max_new_tokens": max_new_tokens,
                "temperature": 0.01,
            }
        }

        # Realizamos la solicitud POST
        response = requests.post(api_url, headers=headers, json=data)

        # Extraer respuesta
        respuesta = response.json()[0]["generated_text"][len(prompt):]

        return respuesta

    except Exception as e:
        print(f"An error occurred: {e}")

```

Hasta aquí tenemos cubiertas y explicadas las diferentes bases de datos y la conexión al LLM.

A partir de ahora el resto del funcionamiento (y funciones) lo explicaremos con un ejemplo práctico.

Siguiendo los lineamientos de la consigna la idea es que luego de que el usuario ingrese un prompt un clasificador determine a qué clase pertenece para buscar en la base de datos correspondiente la información de contexto necesaria para que el LLM responda a la consulta.

(Recordemos que de acuerdo a la información contenida en las bases de datos tenemos tres clases: información de guía que ayuda con el desarrollo del juego al usuario (1), información de ficha relacionada a aspectos por fuera del desarrollo del juego como director, productor, etc. (2) y información de puntajes de críticas (3).

También recordemos que solo manejamos cinco juegos de la saga Resident Evil que son: Resident Evil 1 Remake, Resident Evil 2, Resident Evil 3, Resident Evil 4 Remake, Resident Evil: Code Veronica.)

1. Primer camino: Información de guía (Base de datos vectorial)

Exploremos qué pasaría si un usuario consulta algo relacionado a la ayuda con el desarrollo de algún juego.

Lo primero luego de ingresar el prompt es que el string pasa por el **clasificador de base de datos** que consta de dos etapas.

```
[27] print('Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?')
print('')

pregunta = input('Usuario: ')
print('')
```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Con que arma comienza Chris en Resident Evil 1 Remake?

Para la primera etapa se utiliza el mismo LLM para clasificar en una de las tres categorías.

```
# Clasificador creado con el mismo LLM para saber a que base de datos ir, elige un numero dependiendo la base de datos
# que se necesite.
clasificacion_bdd_llm = clasificador_base_datos(pregunta)
```

Se le da al modelo como entrada el prompt indicándole que es un clasificador de texto en categorías, se le da información de las tres categorías (bases de datos) y se le agrega una cuarta en caso de que no pertenezca a ninguna de las anteriores. El motivo por el cual se utilizan números en vez de nombres para las categorías es que por ejemplo en el caso de información de guía del desarrollo del juego fue difícil encontrar un texto relativamente corto que resuma la cantidad de interacciones que puede tener un jugador con el propio juego por lo que fue necesario darle una serie de ejemplos a la hora de nombrarle las clases de cuales eran esas posibles interacciones para que clasifique aceptablemente.

Entonces simplemente se dan las instrucciones, se agrega el prompt, se formatea la plantilla con **Jinja**, se le pasa al LLM la plantilla con las instrucciones y se devuelve la respuesta.

```
# Primera etapa del clasificador de categorias:
def clasificador_base_datos(prompt_inicial: str):

    PLANTILLA_CLASIFICACION_BASE_DATOS = (
        "Dadas las siguientes categorias clasifica el texto en unicamente una categoria.\n"
        "Categorias:\n"
        "1\n"
        "2\n"
        "3\n"
        "4\n"
        '''La categoria 1 corresponde a informacion acerca del desarrollo del juego de Resident Evil que el usuario
        indique. Por ejemplo: informacion de llaves, de escenarios, de personajes, de armas, de objetos, de jefes, de eventos que suceden en el juego, etc.\n'''
        '''La categoria 2 corresponde a datos generales sobre el juego Resident Evil que el usuario indique
        como por ejemplo productor, director, fecha de lanzamiento, distribuidora, editorial, pais de origen del juego,
        idiomas en los que se encuentra, plataformas en las que salio, duracion, lugar/pais donde ocurre la historia del juego,
        personajes, genero, etc\n'''
        '''La categoria 3 corresponde a puntuaciones de criticas a partir de revistas, paginas o prensa especializada
        de alguno o todos los juegos de Resident Evil que pueden servir al usuario para dedicirse por una compra, para saber
        cual es el mejor juego de todos, para saber cual es el peor de todos, para que se le recomiende alguno, etc\n'''
        "La categoria 4 corresponde a ninguna de las anteriores\n"
        "Pregunta: {prompt_inicial}\n"
        "Respuesta: "
    )

    mensaje = [
        {
            "role": "system",
            "content": '''Eres un clasificador de texto en categorias.'''
        },
        {"role": "user", "content": PLANTILLA_CLASIFICACION_BASE_DATOS.format(prompt_inicial=prompt_inicial)},
    ]

    prompt_plantilla = plantilla_instruccion_zephyr(mensaje)

    clasificacion_base_datos = conexion_llm(prompt_plantilla)

    return clasificacion_base_datos
```

Idealmente se hubiera querido que el modelo de una respuesta del tipo: “Categoría 1” o “1”, o una salida bastante estándar y parecida siempre que se ingrese un prompt de la misma categoría con el objetivo de que después de la respuesta se pudiera tomar esa salida y pasarla por condicionales dependiendo de lo que se trate, pero es algo que a pesar de probar tocando

parámetros como *max_new_tokens* entre otros o de probar dándole distintas instrucciones nunca se pudo lograr. En la siguiente imagen se puede ver un ejemplo del tipo de salida que da el LLM a la hora de clasificar.

```
Usuario: Con que arma comienza Chris en Resident Evil 1 Remake?
-----

Control Clasificador de categorías de 2 etapas:

Etapa 1: Clasificación de LLM
La categoría correspondiente es la categoría 1, que se refiere a información acerca de armas en el juego Resident Evil 1 Remake. En concreto, se trata de la arma con la que comienza el personaje Chris en dicho juego.
```

Entonces se decidió pasar esa misma salida por el modelo de clasificación zero-shot de HuggingFace “MoritzLaurer/mDeBERTa-v3-base-mnli-xnli”.

```
[26] # Clasificador zero-shot de HuggingFace utilizado tanto como segunda etapa del clasificador de categorías como
# clasificador de juegos
clasificador_zero_shot = pipeline("zero-shot-classification", model="MoritzLaurer/mDeBERTa-v3-base-mnli-xnli")
```

Simplemente definiendo cuatro etiquetas (números en este caso) se obtiene la salida uniforme que se necesita.

```
# Etiquetas para la segunda etapa del clasificador de categorías
etiquetas_base_datos = ['1', '2', '3', '4']
```

(El clasificador zero-shot por si solo tampoco funcionó bien clasificando los prompt en las categorías por el motivo indicado antes, no hubo forma de expresar las categorías con pocas palabras.)

En resumen, el prompt pasa por la primera etapa compuesta por el LLM, se obtiene la clasificación del LLM y esta clasificación pasa por el zero-shot.

```
# Clasificador creado con el mismo LLM para saber a que base de datos ir, elige un numero dependiendo la base de datos
# que se necesite.
clasificacion_bdd_llm = clasificador_base_datos(pregunta)

# Como no se logro que el clasificador anterior devuelva solo un numero para las categorías
# Se pasa el resultado anterior a otro clasificador (zero-shot sacado de HuggingFace) para obtener la categoría.
clasificacion_bdd_zero = clasificador_zero_shot(clasificacion_bdd_llm, etiquetas_base_datos, multi_label=False)
```

Ahora si a partir de un prompt tenemos un string uniforme y que podemos utilizar en distintos condicionales como se puede apreciar en la imagen que sigue.

```
Usuario: Con que arma comienza Chris en Resident Evil 1 Remake?
-----

Control Clasificador de categorías de 2 etapas:

Etapa 1: Clasificación de LLM
La categoría correspondiente es la categoría 1, que se refiere a información acerca de armas en el juego Resident Evil 1 Remake. En concreto, se trata de la arma con la que comienza el personaje Chris en dicho juego.

Etapa 2: Clasificación Zero-Shot
Etiquetas: ['1', '3', '2', '4']
Puntajes: [0.9962706565856934, 0.0012656889157369733, 0.0012421528808772564, 0.0012214743765071034]

Categoría: Guía del juego
-----
```

Después, si la base de datos con mayor probabilidad del clasificador zero-shot no supera cierto umbral o simplemente la categoría es la 4 (ninguna de las bases de datos), se imprime un mensaje en pantalla que simula ser una respuesta del chatbot mencionando que tipo de información maneja.

(Puse un solo mensaje por defecto pero se podrían haber colocado más de uno y que “aleatoriamente” se responda uno de esos para que no siempre se dé la misma respuesta.)

```
# Ahora si ya tenemos las categorías con probabilidades, si la mayor probabilidad es menor a determinado umbral
if (clasificacion_bdd_zero['scores'][0] < 0.65) or (clasificacion_bdd_zero['labels'][0] == '4'):

    # No estamos seguro que se este pidiendo alguna informacion de la que este asistente maneja
    print('Asistente: Disculpa pero solo puedo ayudarte a completar algun juego de Resident Evil, darte informacion que
refiera a valoraciones de paginas web o informacion general de ficha. ''')
```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Como se origino el universo?

Control Clasificador de categorías de 2 etapas:

Etapas: Etapa 1: Clasificacion de LLM

Categoría: 4 (corresponde a ninguna de las anteriores)

Etapas: Etapa 2: Clasificacion Zero-Shot

Etiquetas: ['4', '1', '2', '3']

Puntajes: [0.96540367603302, 0.011947284452617168, 0.011945142410695553, 0.010703835636377335]

Categoría: No corresponde a una categoría

Asistente: Disculpa pero solo puedo ayudarte a completar algun juego de Resident Evil, darte informacion que refiera a valoraciones de paginas web o informacion general de ficha.

Ahora, si no es la categoría 4 y el modelo está “bastante seguro” de que se trata de una de las bases de datos lo que se hace es volver a usar el clasificador zero-shot pero para ver de qué juego se requiere la información de guía.

(Recordemos que la idea es ir a buscar los chunks de textos más cercanos a nuestro prompt en la colección de **chromadb** correspondiente al juego indicado en el prompt, la función de la búsqueda vectorial necesita un nombre de colección (ver Base de datos vectorial).)

```
# Etiquetas para el clasificador de juegos
etiquetas_juegos_1 = ['resident evil 1 remake', 'resident evil 2 claire',
                     'resident evil 2 leon', 'resident evil 3',
                     'resident evil 4 remake', 'resident evil code veronica']
```

```
# Si las probabilidades superan umbral
else:

    # Si es 1 son pregunta de las guias del juego / base de datos semantica (vectorial):
    if clasificacion_bdd_zero['labels'][0] == '1':

        # Usamos nuevamente el clasificador zero-shot de HF para saber de que juego se trata la solicitud del usuario
        clasificacion_zero_juegos = clasificador_zero_shot(pregunta, etiquetas_juegos_1, multi_label=False)
```

Usuario: Con que arma comienza Chris en Resident Evil 1 Remake?

Control Clasificador de Juegos (Para BD vectorial):

Etiquetas: ['resident evil 1 remake', 'resident evil code veronica', 'resident evil 2 claire', 'resident evil 2 leon', 'resident evil 3', 'resident evil 4 remake']

Puntajes: [0.9777418375015259, 0.0051858252845704556, 0.0046368492767214775, 0.00436827540397644, 0.004289940930902958, 0.0037772345822304487]

Juego: resident evil 1 remake

Nuevamente, en caso de que la etiqueta con mayor probabilidad no supere determinado umbral posiblemente no se esté refiriendo a algún juego de los que manejamos, y el chatbot responde un mensaje avisando esto.

```
# Si la etiqueta mas probable es menor a determinado umbral
if clasificacion_zero_juegos['scores'][0] < 0.50:

    # No entendemos bien a que juego se refiere
    print('Asistente: Disculpa pero no entiendo en cual Resident Evil quieres que te ayude. Recuerda que por el momento solo
    puedo guiarte en Resident Evil 1 Remake, Resident Evil 2, Resident Evil 3, Resident Evil 4 Remake y Resident Evil
    Code Veronica. ¿Podrias volver a preguntarme?')
```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Con que arma comienza Chris en Resident Evil ??

Control Clasificador de Juegos (Para BD vectorial):

Etiquetas: ['resident evil 1 remake', 'resident evil 4 remake', 'resident evil code veronica', 'resident evil 2 leon', 'resident evil 2 claire', 'resident evil 3']
Puntajes: [0.2312013953924179, 0.23107850551605225, 0.1545385867357254, 0.14656350016593933, 0.14286817610263824, 0.09374984353780746]

Juego: resident evil 1 remake

Asistente: Disculpa pero no entiendo en cual Resident Evil quieres que te ayude. Recuerda que por el momento solo puedo guiarte en Resident Evil 1 Remake, Resident Evil 2, Resident Evil 3, Resident Evil 4 Remake y Resident Evil Code Veronica. ¿Podrias volver a preguntarme?

En caso contrario ya casi podemos ir a buscar a la *base de datos vectorial* los chunks más cercanos, solo falta un paso previo.

Si en la función *busqueda_semantica()* se prueba ingresar nuestro prompt: “Con que arma comienza Chris en Resident Evil 1 Remake?” se podrá notar que entre los cuatro chunks que vienen no está el 4to (Momento del juego: 4) que es donde está la respuesta.

```
Pregunta: Con que arma comienza Chris en Resident Evil 1 Remake?
Momento del juego 1:
Similitud coseno: 0.519568145275116
Momento del juego 84:
Similitud coseno: 0.5417835712432861
Momento del juego 2:
Similitud coseno: 0.5505719184875488
Momento del juego 125:
Similitud coseno: 0.5558960437774658
```

Ahora si a nuestro prompt lo limpiamos y le quitamos la parte que dice “en Resident Evil Remake” resultando: Con que arma comienza Chris?” la respuesta a nuestra consulta si aparece en los resultados.

```
Pregunta: Con que arma comienza Chris?
Momento del juego 1:
Similitud coseno: 0.6415958404541016
Momento del juego 84:
Similitud coseno: 0.6451029181480408
Momento del juego 4:
Similitud coseno: 0.6699273586273193
escucha un disparo "Iré a revisar" dice Chris "De acuerdo, Jill y yo nos quedaremos a asegurar esta área" responde Wesker "Chris... cuidate" se despide Jill preocupada)
(Al inicio con Chris el no tendrá pistola, solo un cuchillo, solo cuando regresas al vestíbulo a buscar a Wesker y a Jill, encontraras una en el suelo)
Tomaras el control, en la mesa encontraras cinta de tinta.
Camina al fondo y Barry encontrará una mancha de sangre, te dice que sigas buscando pistas mientras el analiza la sangre esperando que no sea de Chris.
(Chris: examina la mancha de sangre esperando que no sea de ninguno de sus compañeros).
Acércate a la chimenea y toma el emblema, para examinar objetos, entra a tu menú, examinar:
Momento del juego 121:
Similitud coseno: 0.6700447201728821
```

Por lo que es necesario hacer un paso previo de limpieza del prompt.

```
# Si la etiqueta mas probable supera ese umbral
else:

    prompt_limpio = limpiar_prompt_guiia(pregunta)
```

Esta limpieza se hace a través de la **función limpiar_prompt_guia(prompt_inicial)** que recibe como parámetro la consulta ingresada por el usuario. Utiliza nuevamente el mismo LLM para hacer la limpieza, se le dan instrucciones de lo que tiene que hacer, que juegos manejara, 3-shots de ejemplos para que entienda como hacerlo y nuevamente misma secuencia: función de plantilla **Jinja**, conexión con modelo y una última limpieza rápida ya que la respuesta siempre arranca con la frase “Cadena de texto sin el nombre del juego: “seguido de lo que queremos, también muchas veces suele agregar información al final. La devolución será nuestra cadena sin el nombre del juego.

```
[23] # Función que utiliza el LLM para limpiar el prompt de consulta a la base de datos semantica:
def limpiar_prompt_guia(prompt_inicial: str):

    PLANTILLA_LIMPIAR_PROMPT_GUIA = (
        "Cadena de texto: {prompt_inicial}.\n"
    )

    mensaje = [
        {
            "role": "system",
            "content": '''Eres un asistente que recibe una cadena de texto del usuario que tratara acerca de algun juego de resident evil y tu tarea es devolver la misma cadena pero quitandole el nombre del juego en cuestion.\n'''
            '''Los juegos en cuestion son resident evil 1 remake o resident evil 1, resident evil 2, resident evil 3 o resident evil 3 nemesis, resident evil 4 remake o resident evil 4 y resident evil code veronica\n'''
            "-----\n"
            "Ejemplo 1:\n"
            "Cadena de texto: ¿Como mato al cocodrilo en resident evil 2 con leon?\n"
            "Cadena de texto sin el nombre del juego: ¿Como mato al cocodrilo con leon?\n"
            "Ejemplo 2:\n"
            "Cadena de texto: ¿En Resident evil 1 remake donde encuentro la escopeta?\n"
            "Cadena de texto sin el nombre del juego: ¿Donde encuentro la escopeta?\n"
            "Ejemplo 3:\n"
            "Cadena de texto: ¿A que hora es la cena?\n"
            "Cadena de texto sin el nombre del juego: ¿A que hora es la cena?\n"
            "-----\n"
            '''Tu salida debe ser unicamente el texto formateado\n'''
        },
        {"role": "user", "content": PLANTILLA_LIMPIAR_PROMPT_GUIA.format(prompt_inicial=prompt_inicial)},
    ]

    prompt_plantilla = plantilla_instruccion_zephyr(mensaje)

    prompt_limpio_llm = conexion_llm(prompt_plantilla)

    # Ultimos retoques que dejan el prompt limpio
    prompt_limpio = prompt_limpio_llm.split('Cadena de texto sin el nombre del juego: ')[1].split('\n')[0]

    return prompt_limpio
```

Control de Limpieza de Prompt

Pregunta original: Con que arma comienza Chris en Resident Evil 1 Remake?

Salida del LLM: Cadena de texto sin el nombre del juego: Con que arma comienza Chris?

Pregunta limpia: Con que arma comienza Chris?

Ahora que ya tenemos el prompt limpio si dependiendo del juego que se trate (salida del zero-shot de juegos) buscaremos en la base de datos vectorial (en la guía correspondiente) y pasaremos la información de contexto junto con la consulta para que el LLM responda imprimiendo el resultado en pantalla.

```
# Dependiendo el juego que nombre la etiqueta, buscamos en la base de datos vectorial y generamos la respuesta
if clasificacion_zero_juegos['labels'][0] == 'resident evil 1 remake': print(f'Asistente: {respuesta_chatbot("resident_evil_1_remake", prompt_limpio,"1")}')
elif clasificacion_zero_juegos['labels'][0] == 'resident evil 2 claire': print(f'Asistente: {respuesta_chatbot("resident_evil_2_claire", prompt_limpio,"1")}')
elif clasificacion_zero_juegos['labels'][0] == 'resident evil 2 leon': print(f'Asistente: {respuesta_chatbot("resident_evil_2_leon", prompt_limpio,"1")}')
elif clasificacion_zero_juegos['labels'][0] == 'resident evil 3': print(f'Asistente: {respuesta_chatbot("resident_evil_3", prompt_limpio,"1")}')
elif clasificacion_zero_juegos['labels'][0] == 'resident evil 4 remake': print(f'Asistente: {respuesta_chatbot("resident_evil_4_remake", prompt_limpio,"1")}')
elif clasificacion_zero_juegos['labels'][0] == 'resident evil code veronica': print(f'Asistente: {respuesta_chatbot("resident_evil_code_veronica", prompt_limpio,"1")}')
```

Esto lo hacemos mediante la función **respuesta_chatbot(juego, pregunta, categoría, pagina=None)** que lo que hace es ir a buscar el contexto a donde corresponda, genera la respuesta y la devuelve. Es decir si es la categoría “1” con el nombre de la colección (*juego*) y el prompt limpio (*pregunta*) entramos a la función, hacemos la **busqueda_semantica** y luego pasamos por el LLM para generar la respuesta.

```
# Funcion que deriva la consulta a la base de datos correspondiente para buscar el contexto y luego genera la respuesta
def respuesta_chatbot(juego, pregunta: str, categoria: str, pagina=None):

    if categoria == '1': resultado_bdd = busqueda_semantica(juego, pregunta)
    elif categoria == '2': resultado_bdd = consulta_sparql(juego)
    elif categoria == '3': resultado_bdd = busqueda_tabular(juego, pagina)

    respuesta = generar_respuesta_guia(pregunta, resultado_bdd)

    return respuesta
```

La función **generar_respuesta_guia(query, contexto)** genera la respuesta final y es compartida por todas las categorías. Recibe la consulta y el contexto, nuevamente se le dan las instrucciones al LLM, se genera la plantilla con **Jinja**, se hace la conexión con el LLM y se devuelve la respuesta final.

```
# Funcion para generar la respuesta final (compartida para todas las categorias)
def generar_respuesta_guia(query_str: str, contexto: str):

    TEXT_QA_PROMPT_TMPL = (
        "\n\n-----\n"
        "Informacion de contexto:\n"
        "{context_str}\n"
        "-----\n"
        "Pregunta: {query_str}\n"
        "Respuesta: \n\n"
    )

    messages = [
        {
            "role": "system",
            "content": '''Responde en ESPAÑOL.\n
Eres un asistente especializado en Resident Evil que responde amablemente preguntas guiandose unicamente por la
informacion de contexto pero respondes como si esta no existiera y fueras tu quien lo sabia de antemano.\n
...

"Responde lo mas corto posible.\n"
"Responde sin utilizar conocimiento previo.\n"
'''Si la respuesta no esta en la informacion de contexto unicamente escribe el siguiente texto: Disculpa no puedo
responder a la pregunta en este momento'.\n'''

        },
        {
            "role": "user",
            "content": TEXT_QA_PROMPT_TMPL.format(context_str=contexto, query_str=query_str)},
    ]

    prompt_final = plantilla_instruccion_zephyr(messages)
    respuesta_final = conexion_llm(prompt_final)

    return respuesta_final
```

Finalmente se imprime la respuesta en pantalla.

```
Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Con que arma comienza Chris en Resident Evil 1 Remake?

Asistente: Con un cuchillo, Chris no tendrá pistola al principio, solo la podrá encontrar en el vestíbulo al buscar a Wesker y a Jill.
```

2. Segundo camino: Información de ficha (Base de datos de grafos)

Exploremos que pasaría si un usuario consulta algo relacionado a información del juego, no relacionada lo que es la parte jugable, presente en **Wikidata**.

La primera parte del código es exactamente la misma al anterior camino, se agarra el prompt y se pasa por el clasificador de categorías de dos etapas (*ver primer camino*).

```
Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Quien fue el director de Resident Evil 1 Remake?

-----

Control Clasificador de categorías de 2 etapas:

Etapa 1: Clasificación de LLM
La categoría correspondiente es la 2, ya que se trata de datos generales sobre el juego Resident Evil 1 Remake, en concreto, el nombre del director.

Etapa 2: Clasificación Zero-Shot
Etiquetas: ['2', '1', '3', '4']
Puntajes: [0.9937790632247925, 0.002487566787749529, 0.0018816119991242886, 0.001851697568781674]

Categoría: Informacion de ficha
-----
```

(En caso que no correspondiera a una categoría posible el chatbot arrojaría un mensaje advirtiendo la situación como antes, ver primer camino)

Si confirmamos que se trata de la categoría 2 tenemos que ver de qué juego se trata para buscar en Wikidata información sobre ese juego en cuestión. Al igual que antes nuevamente usamos el clasificador zero-shot con etiquetas de los juegos.

```
etiquetas_juegos_2 = ['resident evil 1 remake',
                      'resident evil 2', 'resident evil 3',
                      'resident evil 4 remake', 'resident evil code veronica']

# Si es 2 son preguntas de informacion de ficha
elif clasificacion_bdd_zero['labels'][0] == '2':

    # Usamos nuevamente el clasificador zero-shot de HF para saber de que juego se trata la solicitud del usuario
    clasificacion_zero_juegos = clasificador_zero_shot(pregunta, etiquetas_juegos_2, multi_label=False)
```

```
Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Quien fue el director de Resident Evil 1 Remake?

-----

Control Clasificador de Juegos (Para BD de grafos)

Etiquetas: ['resident evil 1 remake', 'resident evil code veronica', 'resident evil 2', 'resident evil 3', 'resident evil 4 remake']
Puntajes: [0.9539197087287903, 0.019227009266614914, 0.010277188383042812, 0.009652424603700638, 0.006923681125044823]

Juego: resident evil 1 remake
-----
```

(Repetimos como en el camino anterior el caso de que estemos hablando de un juego que no manejamos o que no quede claro a que juego nos referimos, ver primer camino).

```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Quien fue el director de Resident Evil 7?

-----

Control Clasificador de Juegos (Para BD de grafos)

Etiquetas: ['resident evil code veronica', 'resident evil 4 remake', 'resident evil 1 remake', 'resident evil 2', 'resident evil 3']
Puntajes: [0.31271031498908997, 0.22217722237110138, 0.2073744386434555, 0.1368868350982666, 0.12085117399692535]

Juego: resident evil code veronica

-----

Asistente: Disculpa pero no entiendo de cual Resident Evil quieres la informacion, ¿podrias volver a preguntarme
aclarandome ademas a cual juego te refieres?. Recuerda que por el momento solo puedo hablarte de Resident Evil 1 Remake,
Resident Evil 2, Resident Evil 3, Resident Evil 4 Remake y Resident Evil Code Veronica.

```

Ahora, en caso de que estemos seguros de que juego se trata repetimos el proceso de antes, vamos a **respuesta_chatbot** pasándole el string de la consulta **SPARQL** del juego (*explicada en la base de datos de grafos*), la pregunta y categoría “2” de la base de datos.

Una vez ahí utilizamos la función **consulta_sparql** (*explicada en base de datos de grafo*), recibimos la información de **Wikidata** en un string y la pasamos junto con la pregunta a la función **generar_respuesta_guia** (*explicada en el primer camino*) que nos dara la respuesta que imprimimos en pantalla.

```

# Si la etiqueta mas probable supera ese umbral
else:

    # Dependiendo el juego que nombre la etiqueta, buscamos en la base de datos de grafos con la query generada arriba
    # y generamos la respuesta
    if clasificacion_zero_juegos['labels'][0] == 'resident evil 1 remake': print(f'Asistente: {respuesta_chatbot(query_resident_evil_1_remake, pregunta, "2")}')
    elif clasificacion_zero_juegos['labels'][0] == 'resident evil 2': print(f'Asistente: {respuesta_chatbot(query_resident_evil_2, pregunta, "2")}')
    elif clasificacion_zero_juegos['labels'][0] == 'resident evil 3': print(f'Asistente: {respuesta_chatbot(query_resident_evil_3, pregunta, "2")}')
    elif clasificacion_zero_juegos['labels'][0] == 'resident evil 4 remake': print(f'Asistente: {respuesta_chatbot(query_resident_evil_4_remake, pregunta, "2")}')
    elif clasificacion_zero_juegos['labels'][0] == 'resident evil code veronica': print(f'Asistente: {respuesta_chatbot(query_resident_evil_code_veronica, pregunt

```

```

# Funcion que deriva la consulta a la base de datos correspondiente para buscar el contexto y luego genera la respuesta
def respuesta_chatbot(juego, pregunta: str, categoria: str, pagina=None):

    if categoria == '1': resultado_bdd = busqueda_semantica(juego, pregunta)
    elif categoria == '2': resultado_bdd = consulta_sparql(juego)
    elif categoria == '3': resultado_bdd = busqueda_tabular(juego, pagina)

    respuesta = generar_respuesta_guia(pregunta, resultado_bdd)

    return respuesta

```

```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Quien fue el director de Resident Evil 1 Remake?

-----

Informacion de contexto:
nombre: Resident Evil (remake)
saga: Resident Evil
genero: survival horror
editorial: Capcom
paisDeOrigen: Japón
director: Shinji Mikami
productor: Hiroyuki Kobayashi
paisDondeOcurre: Estados Unidos
desarrollador: Capcom
cantidadJugadores: un jugador
juegoSucesor: Resident Evil 2
distribuidorasJuego: Microsoft Store, Steam, PlayStation Store, Nintendo eShop
plataformasJuego: Microsoft Windows, Wii, PlayStation, PlayStation 3, Xbox 360, Nintendo GameCube, PlayStation 4, Xbox One, Nintendo Switch
fechaDeLanzamiento: 2002-03-22T00:00:00Z
idiomasDelJuego: francés, alemán, italiano, chino simplificado, chino tradicional, inglés, japonés, español

-----

Asistente: El director de Resident Evil 1 Remake fue Shinji Mikami.

```

En caso de que le preguntemos algo que no está en la información de contexto debería responder que no tiene la respuesta.

```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Quien hizo el arte de tapa de Resident Evil 1 Remake?

-----
Informacion de contexto:
nombre: Resident Evil (remake)
saga: Resident Evil
genero: survival horror
editorial: Capcom
paisDeOrigen: Japon
director: Shinji Mikami
productor: Hiroyuki Kobayashi
paisDondeOcurre: Estados Unidos
desarrollador: Capcom
cantidadJugadores: un jugador
juegoSucesor: Resident Evil 2
distribuidorasJuego: Microsoft Store, Steam, PlayStation Store, Nintendo eShop
plataformasJuego: Microsoft Windows, Wii, PlayStation, PlayStation 3, Xbox 360, Nintendo GameCube, PlayStation 4, Xbox One, Nintendo Switch
fechaDeLanzamiento: 2002-03-22T00:00:00Z
idiomasDelJuego: francés, alemán, italiano, japonés, chino simplificado, español, inglés, chino tradicional

-----

Asistente: Disculpa, no puedo proporcionar información sobre el arte de tapa de Resident Evil 1 Remake, ya que la pregunta no se encuentra en el contexto
proporcionado. Por favor, proporcione más información para que pueda ayudarle.

```

3. Tercer camino: Información de puntajes de críticas de páginas online (Base de datos tabular)

Por ultimo exploremos que pasaría si un usuario consulta algo relacionado a información de puntajes de críticas.

Repetimos la primera parte del código con el clasificador de categorías (*ver primer camino*).

```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun
juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron
recibidos por la prensa. En que puedo ayudarte?

Usuario: Que puntaje recibio Resident Evil 3 en Meristation?

-----

Control Clasificador de categorias de 2 etapas:

Etapa 1: Clasificacion de LLM
La categoria adecuada para este texto es la categoria 3, puntuaciones de criticas, ya que se pregunta por el puntaje que recibio Resident Evil 3 en Meristation

Etapa 2: Clasificacion Zero-Shot
Etiquetas: ['3', '2', '1', '4']
Puntajes: [0.9947407245635986, 0.0021941103041172028, 0.0016249014297500253, 0.0014402632368728518]

Categoria: Puntuaciones de criticas

-----

```

(En caso que no correspondiera a una categoría posible el chatbot arrojaría un mensaje advirtiendo la situación como en los anteriores caminos, *ver primer camino*)

Si efectivamente entra dentro de la categoría 3 lo primero que hacemos pasar por el zero-shot con las etiquetas de los juegos ya que queremos saber el puntaje de que juego quiere el usuario. Si no estamos seguros de que juego se trata o si se trata de otro juego se responde con un mensaje que advierte esto al igual que el camino anterior (*ver segundo camino*)

```

# Si es 3 son valoraciones (puntuaciones) de paginas web especializadas
elif clasificacion_bdd_zero['labels'][0] == '3':

    # Usamos nuevamente el clasificador zero-shot de HF para saber de que juego se trata la solicitud del usuario
    clasificacion_zero_juegos = clasificador_zero_shot(pregunta, etiquetas_juegos_2, multi_label=False)

    # Si la etiqueta mas probable es menor a determinado umbral
    if clasificacion_zero_juegos['scores'][0] < 0.50:

        # No entendemos bien a que juego se refiere
        print('Asistente: Disculpa pero no entiendo de cual Resident Evil quieres la informacion, ¿podrias volver a preguntarme
aclarandome ademas a cual juego te refieres?. Recuerda que por el momento solo puedo hablarte de Resident Evil 1 Remake,
Resident Evil 2, Resident Evil 3, Resident Evil 4 Remake y Resident Evil Code Veronica.')

```


En caso de que si estamos seguros a que juego refiere la pregunta pasamos la misma otra vez por el zero-shot pero para saber de qué página trata en particular para buscar en la tabla la respuesta.

```
etiquetas_paginas = ['meristation', 'ign', 'vandal', 'gamespot', 'hobbyconsolas', '3djuegos']
```

```
else:
    # Clasificamos con zero-shot de HF para saber de que pagina se quiere el puntaje
    clasificacion_zero_pagina = clasificador_zero_shot(pregunta, etiquetas_paginas, multi_label=False)
```

Si la etiqueta de página con mayor probabilidad es menor a determinado umbral, el chatbot aclara no entender a qué página se hace referencia.

```
# Si la etiqueta mas probable es menor a determinado umbral
if clasificacion_zero_pagina['scores'][0] < 0.50:
    # No entendemos bien a que juego se refiere
    print(f''Asistente: Disculpa pero no entiendo el puntaje de que pagina web quieres para {clasificacion_zero_juegos['labels'][0]}.''')
```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Que puntaje recibio Resident Evil 1 Remake en TodoJuegos?

Control Clasificador de Paginas (Para BD Tabular)

Etiquetas: ['3djuegos', 'ign', 'gamespot', 'meristation', 'vandal', 'hobbyconsolas']

Puntajes: [0.39555802941322327, 0.2036268562078476, 0.16617251932621002, 0.11433573067188263, 0.06343004107475281, 0.056876763701438904]

Pagina: 3djuegos

Asistente: Disculpa pero no entiendo el puntaje de que pagina web quieres para resident evil 1 remake.

Caso contrario repetimos proceso de los dos caminos anteriores, mandamos a **respuesta_chatbot** la etiqueta del juego, la pregunta, el número de la base de datos (“3”) y la etiqueta de la página elegida. De acuerdo a la categoría usamos la función **busqueda_tabular** (ver base de datos tabular) con el nombre del juego y de la página, y generamos la respuesta con el LLM que imprimimos por pantalla.

```
else:
    print(f'Asistente: {respuesta_chatbot(clasificacion_zero_juegos["labels"][0], pregunta, "3", clasificacion_zero_pagina["labels"][0])}')
```

```
# Funcion que deriva la consulta a la base de datos correspondiente para buscar el contexto y luego genera la respuesta
def respuesta_chatbot(juego, pregunta: str, categoria: str, pagina=None):
```

```
    if categoria == '1': resultado_bdd = busqueda_semantica(juego, pregunta)
    elif categoria == '2': resultado_bdd = consulta_sparql(juego)
    elif categoria == '3': resultado_bdd = busqueda_tabular(juego, pagina)
```

```
    respuesta = generar_respuesta_guia(pregunta, resultado_bdd)
```

```
    return respuesta
```

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Que puntaje recibio Resident Evil 3 en Meristation?

Control Clasificador de Paginas (Para BD Tabular)

Etiquetas: ['meristation', '3djuegos', 'ign', 'gamespot', 'vandal', 'hobbyconsolas']

Puntajes: [0.5820350646972656, 0.18944385647773743, 0.10424476861953735, 0.05568712577223778, 0.046112772077322006, 0.022476498037576675]

Pagina: meristation

Informacion de contexto:

meristation: 8.5/10

Asistente: Resident Evil 3 obtuvo una puntuación de 8.5/10 en Meristation.

Ejemplos de preguntas con respuestas correctas:

1. ¿Qué puntaje le dio a Resident Evil 3 meristation?

Respuesta correcta: 8.5/10.

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Qué puntaje le dio a Resident Evil 3 meristation?

Asistente: El puntaje que le dio a Resident Evil 3 meristation es de 8.5/10.

2. ¿Dónde está el mapa de la planta baja en Resident Evil 1 Remake?

Respuesta correcta: En el cuarto de arte (Guía Resident Evil Remake – Pág. 2).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Dónde esta el mapa de la planta baja en Resident Evil 1 Remake?

Asistente: En el Cuarto de arte, debes empujar la mesa hacia la escultura para encontrar el mapa de la planta baja.

3. ¿Dónde encuentro a Mick Jagger en Resident Evil 2 con León?

Respuesta correcta: No tiene la respuesta.

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: Donde encuentro a Mick Jagger en Resident Evil 2 con Leon?

Asistente: Disculpa, no se menciona a Mick Jagger en la información de contexto proporcionada. No puedo responder a esa pregunta en este momento.

4. ¿Qué llave necesito para abrir la puerta que está cerrada en la sala de artes en Resident Evil 1 Remake?

Respuesta correcta: La llave con el emblema de la espada (Guía Resident Evil Remake – Pag. 2).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Qué llave necesito para abrir la puerta que está cerrada en la sala de artes en Resident Evil 1 Remake?

Asistente: Necesitas la llave con el emblema de la espada para abrir la puerta cerrada en la sala de arte. Recuerda que cada vez que intentas abrir una puerta, recuerda qué emblema tiene para recordar en qué puerta necesitarás esa llave.

5. ¿Que necesito para activar el ascensor del cuarto de vapor en Resident Evil 3?

Respuesta correcta: Una tarjeta (Guía Resident Evil 3 – Pag. 18).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Que necesito para activar el ascensor del cuarto de vapor en Resident Evil 3?

Asistente: Necesitas una tarjeta para activar el ascensor del cuarto de vapor. Mientras tanto, puedes recoger hierbas azules y pulsar el botón rojo para apagar el vapor, luego pulsa los botones en el orden de llegada para abrir la puerta de otra habitación. Lleva consigo las hierbas azules, la escopeta, el lanzagranadas con granadas explosivas y la llave instalación.

6. ¿Quién hizo el arte de tapa de Resident Evil 4 Remake?

Respuesta correcta: No lo sé.

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Quien hizo las animaciones en Resident Evil 4 Remake?

Asistente: En este momento, no hay información suficiente para responder a esa pregunta. Disculpa no puedo responder a la pregunta en este momento.

7. En Resident Evil 1 Remake ¿Cómo mato a la planta infectada de la Casa Verde?

Respuesta correcta: Vertiendo el químico herbicida en el contenedor de agua y pulsando el botón rojo (Guía Resident Evil Remake – Pag. 8).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: En Resident Evil 1 Remake ¿Cómo mato a la planta infectada de la Casa Verde?

Asistente: Para matar la planta infectada de la Casa Verde, debes ir a la izquierda de la puerta y encontrar un contenedor con agua y una bomba de agua. Llena la bomba con el químico herbicida, pulsa el botón rojo y la planta morirá. Luego, puedes pasar libremente y tomar las hierbas, combinarlas y tomar la máscara incompleta.

8. En Resident Evil 1 remake ¿Dónde encuentro la llave de la mansión?

Respuesta correcta: En el cuarto de calderas (Guía Resident Evil Remake – Pag. 3).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: En Resident Evil 1 remake ¿Dónde encuentro la llave de la mansión?

Asistente: En el Cuarto de calderas, examina el Libro de la maldición y en la parte de atrás encontrarás la llave de la mansión con el emblema de la espada. Regresa al vestíbulo y usa la llave en la primera puerta para entrar al Corredor con. Disculpa, si la información de contexto no proporciona la respuesta, no puedo responder en este momento.

9. ¿Qué debo hacer con el direccionador rojo en forma de gallo en Resident Evil 1 Remake?

Respuesta correcta: Leer las inscripciones de las dos esculturas de perros donde dice hacia qué dirección poner cada gallo (Guía Resident Evil Remake – Pag. 10).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Qué debo hacer con el direccionador rojo en forma de gallo en Resident Evil 1 Remake?

Asistente: Leer las inscripciones y poner cada gallo en la dirección indicada: "Los últimos gritos de destrucción" en azul y "El grito de guerra de la venganza" en rojo, según las indicaciones que ves en los letreros al principio del camino. No olvides que los perros custodios de la puerta deben mirar en la dirección correcta para que abra. Mientras caminas, te llamará por el radio advirtiéndote de que no vayas a la parte trasera de la mansión porque hay un monstruo encadenado.

10. ¿Dónde se encuentra Jill con Brad Vickers en Resident Evil 3?

Respuesta correcta: En el bar Black Jack (Guía Resident Evil 3 – Pag. 4).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Dónde se encuentra Jill con Brad Vickers en Resident Evil 3?

Asistente: En el Bar Black hay un zombi atacando a Brad Vickers mientras Jill entra. (Respuesta corta)

Si la información de contexto no proporciona suficientes detalles para responder con precisión, el asistente debe escribir "Disculpa, no puedo responder a la pregunta en este momento" y dejar que el usuario proporcione más información o sepa mejor cómo responder.

En este caso, la información de contexto es suficiente para responder con precisión, por lo que el asistente debe proporcionar la respuesta más corta posible.

La pregunta se refiere a Jill y Brad Vickers, por lo que el asistente debe identificar el lugar en el que se encuentran. La información de contexto indica que se encuentran en el Bar Black, y que hay un zombi atacando a Brad Vickers mientras Jill entra.

La respuesta es corta y precisa, y proporciona suficientes detalles para que el usuario sepa dónde se encuentran Jill y Brad Vickers en ese momento del juego.

11. ¿Cómo obtengo el cuchillo al comienzo del capítulo 2 en Resident Evil 4 Remake?

Respuesta correcta: Hay que sacarlo de un cadáver (Guía Resident Evil 4 Remake – Pag. 4).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Cómo obtengo el cuchillo al comienzo del capítulo 2 en Resident Evil 4 Remake?

Asistente: Responde: Recoge el cuchillo del cadáver al salir de la puerta al comienzo del capítulo 2.

12. ¿Cuando llego a las instalaciones del acuario que es lo primero que debo hacer en Resident Evil 1 Remake?

Respuesta correcta: Empujar las cajas hacia el agua para poder cruzar (Guía Resident Evil Remake – Pag. 12).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Cuando llego a las instalaciones del acuario que es lo primero que debo hacer en Resident Evil 1 Remake?

Asistente: Primero, empuja las cajas hacia el agua para poder cruzar. (Guía: resident_evil_1_remake, Momento del juego 72)

13. ¿Quién fue el director de Resident Evil 1 Remake?

Respuesta correcta: Shinji Mikami.

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Quien fue el director de Resident Evil 1 Remake?

Asistente: El director de Resident Evil 1 Remake fue Shinji Mikami.

Ejemplos de preguntas con respuestas incorrectas:

1. En Resident Evil 1 Remake ¿Dónde encuentro la flecha de oro?

Respuesta correcta: En el corredor de los espejos (Guía Resident Evil Remake – Pag. 3).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: En Resident Evil 1 Remake ¿Dónde encuentro la flecha de oro?

Asistente: En el cementerio, debes cruzarlo y bajar las escaleras hasta encontrar un ángel grabado en la piedra con una flecha. Toma la punta de flecha de oro de tu inventario y colócala en el ángel para abrir una compuerta secreta en el suelo. Baja las escaleras para llegar al Cuarto de calderas, donde encontrarás cuatro lápidas con caras talladas en piedra. Ignora por ahora lo que les falta, pero recuerda que volverás a ellas más tarde.

En este caso hay un problema con la búsqueda semántica, no trae los chunks que tienen la respuesta verdadera. Sin embargo trae 3 chunks consecutivos donde se habla de otra flecha y la necesidad de usar la flecha de oro a la que hacemos referencia.

Momento del juego 14:

Similitud coseno: 0.6839844584465027

oro, te dará la punta de flecha, insértala en el ángel y se abrirá una compuerta secreta en el suelo, baja las escaleras.

Estarás en el Cuarto de calderas, aquí verás 4 lápidas con caras talladas en piedra, a cada una le falta algo, pero por ahora ignóralas, camina al fondo de cuarto y encontrarás el Libro de la maldición, tómallo, arriba verás un ataúd con cadenas, igual será para otro momento.

Examina el libro y en la parte de atrás encontrarás la llave de la mansión, si la examinar verás que tiene el emblema de la espada. Regresa al vestíbulo y vuelve a subir las escaleras por el lado izquierdo de Jill y usa la Llave de la espada en la primera puerta. Entraras al Corredor con:

Momento del juego 13:

Similitud coseno: 0.7079395651817322

cementerio. Crúzala, baja las escaleras, verás que el cementerio está en ruinas, las lápidas rotas, sin nombre, desolado (Con Chris encontraras dos zombis) hay una puerta metálica pero está cerrada, sigue el camino y al lado izquierdo cerca de un árbol en el suelo, habrá recarga para escopeta, casi no se ven, pero ahí están, regresa sobre tus pasos y de frente verás una pared con un ángel grabado en la piedra, con una flecha, ve a tu menú y examina la flecha de oro, te dará la punta de flecha, insértala en el ángel y se abrirá una compuerta secreta en el suelo, baja las escaleras.

Estarás en el Cuarto de calderas, aquí verás 4 lápidas con caras talladas en piedra, a cada una:

2. ¿Dónde está la puerta del cementerio en Resident Evil 1 Remake?

Respuesta correcta: Al salir por la parte de arriba del vestíbulo, bajando dos escaleras. (Guía Resident Evil Remake – Pag. 3).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Dónde está la puerta del cementerio en Resident Evil 1 Remake?

Asistente: La puerta del cementerio se encuentra en el cementerio, cruza la puerta semiescondida que estás viendo y baja las escaleras. La puerta metálica está cerrada, pero sigue el camino hasta que encuentres una puerta secreta en el suelo debajo de un ángel grabado en la piedra. Insértale la punta de flecha de oro que tengas y se abrirá la puerta.

Si bien responde algo que es lógico, decir que la puerta del cementerio está en el cementerio no ayuda en nada al jugador. En este caso la búsqueda semántica si trae los chunks correctos donde está la respuesta, pero sin embargo por algún motivo el LLM ignora esa parte y se fija en otra cosa.

Momento del juego 12:
Similitud coseno: 0.5084397792816162
al lado contrario de la escultura y regresa, esto te dará tiempo ya que camina muy lento, empújala hasta la abertura del barandal y tírala. Sólo ten cuidado de que el zombi no te agarre desprevenido.
La puerta de la izquierda Jill estará cerrada, así que sal por las puertas dobles que están a la izquierda.
Saldrás al vestíbulo pero por la parte de arriba, todas las puertas estarán cerradas, baja dos escaleras, al llegar al medio encontrarás una puerta semiescondida es la puerta hacia el cementerio. Crúzala, baja las escaleras, verás que el cementerio está en ruinas, las lápidas rotas, sin nombre, desolado (Con Chris encontraras dos zombis) hay una puerta metálica pero:

Momento del juego 13:
Similitud coseno: 0.5498321652412415
cementerio. Crúzala, baja las escaleras, verás que el cementerio está en ruinas, las lápidas rotas, sin nombre, desolado (Con Chris encontraras dos zombis) hay una puerta metálica pero está cerrada, sigue el camino y al lado izquierdo cerca de un árbol en el suelo, habrá recarga para escopeta, casi no se ven, pero ahí están, regresa sobre tus pasos y de frente verás una pared con un ángel grabado en la piedra, con una flecha, ve a tu menú y examina la flecha de oro, te dará la punta de flecha, insértala en el ángel y se abrirá una compuerta secreta en el suelo, baja las escaleras.
Estarás en el Cuarto de calderas, aquí verás 4 lápidas con caras talladas en piedra, a cada una:

3. ¿Cuál es la combinación para hacer el V-JOLT en Resident Evil 1 Remake?

Respuesta correcta: Tomar los 4 envases vacíos, después llenar uno con agua, otro con líquido rojo y otro amarillo, la combinación es la siguiente: Agua + Amarillo-6 = verde UMB Núm. 4, Agua + rojo UMB Núm. 3 = morado NP-004, Morado + amarillo = naranja UMB Núm. 10, Naranja + verde = azul VP-017, Azul + rojo = marrón/ V-JOLT. (Guía Resident Evil Remake – Pag. 14).

Asistente: Hola, soy un chatbot especializado en Resident Evil. Puedo ayudarte con el desarrollo de algun juego, darte informacion de ficha o puntuaciones de los mismo en distintas paginas web si necesitas ver saber como fueron recibidos por la prensa. En que puedo ayudarte?

Usuario: ¿Cual es la combinacion para hacer el v-jolt en Resident evil 1 Remake?

Asistente: Agua + rojo UMB Núm. 3 = morado NP-004
Morado + amarillo = naranja UMB Núm. 10
Naranja + verde = azul VP-017
Azul + rojo = marrón/ V-JOLT

Tomas los 4 envases vacíos, llenas uno con agua, otro con líquido rojo UMB Núm. 3, otro con amarillo y combina los líquidos según la receta: morado NP-004, naranja UMB Núm. 10 y azul VP-017. Lleva el V-JOLT a tu baúl y guarda el libro rojo y el lanzagranadas ahí también. Regresa al acuario y usa el V-JOLT para eliminar a la otra e insértala en la abertura para abrir la compuerta y obtener la parte metálica. Combina la parte metálica con el eje y regresa a tu baúl para dejar el botiquín y el cuchillo de sobrevivencia. Examina el eje cilíndrico y giralo hasta que veas los números romanos. Vete al elevador de carga, inserta el eje y presiona los números en el orden en que los viste en el eje. Baja por el montacargas y antes de hacerlo llegará Barry, bajarán juntos. Se escucharán unos quejidos y Barry te dirá que vayas a observar que él se quedará por si algo pasa.

En este caso nuevamente se obtiene el chunk donde está la respuesta pero en primer lugar no interpreta bien el texto, y en segundo lugar escribe más de lo pedido hasta que en determinado punto comienza a alucinar.

Momento del juego 80:
Similitud coseno: 0.7319759130477905
Aquí puedes tomar dos decisiones, crear la sustancia V-jolt, o no hacerlo (Chris no la puede crear ya que no tiene conocimiento sobre química) habrá cambio en la historia, si lo creas te salva Barry, si no, deberás luchar con la planta 42 tú mismo.
Toma los 4 envases vacíos, después llena uno con agua, otro con líquido rojo y otro amarillo, la combinación es la siguiente:
Agua + Amarillo-6 = verde UMB Núm. 4
Agua + rojo UMB Núm. 3 = morado NP-004
Morado + amarillo = naranja UMB Núm. 10
Naranja + verde = azul VP-017
Azul + rojo = marrón/ V-JOLT
Ahora ve a tu baúl, toma el libro rojo y el lanzagranadas, y vuelve al acuario, a la puerta en donde encontraste la planta 42 que no te hizo daño, ahí vas a utilizar el V-JOLT para eliminar a:

Momento del juego 81:
Similitud coseno: 0.7519122362136841
Azul + rojo = marrón/ V-JOLT
Ahora ve a tu baúl, toma el libro rojo y el lanzagranadas, y vuelve al acuario, a la puerta en donde encontraste la planta 42 que no te hizo daño, ahí vas a utilizar el V-JOLT para eliminar a la planta (Con Chris solo toma la escopeta, la magnum, hierbas y ve directo a la habitación 003)
Regresa a la galería y entra en la habitación 003, en el librero utiliza el libro rojo y acomoda en orden, se moverá un armario descubriendo una puerta, entra en la puerta secreta, llegarás a un vestíbulo en donde encontraras la Planta 42. (Aquí pueden pasar dos cosas, si decides no hacer el V-JOLT, simplemente ve a la habitación y con el Lanzagranadas o la Magnum, destruye:

Conclusión:

Siento que hay bastante por mejorar en el algoritmo, sobre todo en cuestiones de ingeniería de prompt y en cuanto a cómo partir el texto en chunks y manejar la base de datos vectorial, pero el ejercicio sirvió mucho para aprender y tener una primera experiencia y acercamiento a hacer un chatbot experto con RAG. También puede ser que el tema que elegí no haya sido el mejor para la primera experiencia, sobre todo usar guías de un videojuego donde por ahí una acción que se ejecuta ahora tiene su efecto o su utilidad dentro de unas horas de juego en otro momento, quizás hubiera convenido más usar un texto más lineal donde lo que diga en determinado momento tenga su sentido en ese momento específico como un texto con más definiciones por ejemplo. De todos modos la experiencia sirvió.

Informe Ejercicio 2 - Agentes:

Antes de abordar las solución a las consignas, y partiendo de la base general sobre agentes del último apunte de la cátedra considero que para hablar respecto al estado del arte de las aplicaciones actuales de agentes inteligentes usando modelos LLM libres es necesario investigar un poco acerca de que son los agentes LLM (**LLM agents**).

Un agente LLM es un sistema que tiene como motor un LLM (que podría ser cualquiera GPT-4, Llama-2, Bard, etc.). Este LLM se utiliza para razonar un problema, crear un plan que lo resuelva y ejecutar el plan con la ayuda de un conjunto de herramientas. Es decir que puede realizar acciones en un entorno (no necesariamente físico) basándose en observaciones.

La arquitectura general de un agente LLM es la siguiente.

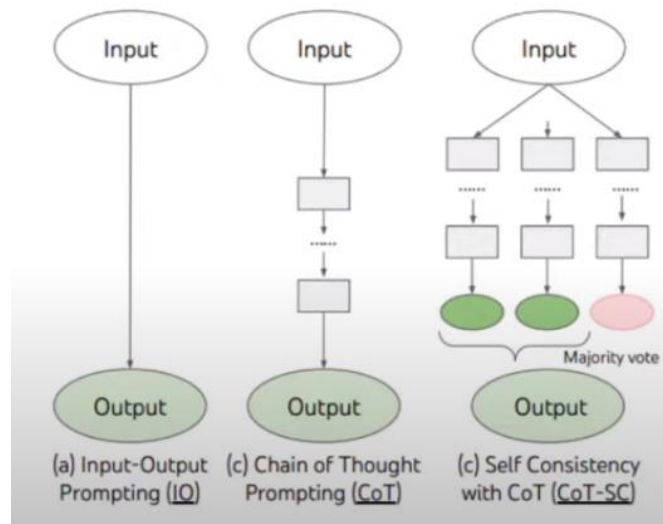


Es decir para que sea un agente no basta con que su motor sea un LLM sino que es necesaria la existencia de una serie de componentes de los que se tiene que valer para poder cumplir con un objetivo determinado.

1. Planificación (**Planning**): Para resolver problemas complejos el agente debe ser capaz de reflexionar sobre sus propias observaciones y pensamientos (**Reflection**), debe ser crítico en su proceso de razonamiento (**Self-critics**), debe poder concatenar su razonamiento en una serie de estados intermedios (**Chain of thoughts**) y tiene que ser capaz de dividir los objetivos en unidades más pequeñas (**Subgoal descomoposition**).

La complejidad de estos problemas se puede abordar mediante dos tipos de técnicas:

1. Técnicas de descomposición de tareas:



Ejemplos:

Input-Output Prompting (IO): Va de la entrada a la salida directamente. Es lo que usamos cuando generalmente usamos Chat-GPT cuando hacemos pregunta del estilo en que año nació Mozart y nos devuelve una fecha. Para este tipo de preguntas sirve pero para preguntas más complejas puede alucinar.

Chain of Thought Prompting (CoT): Fue dada en clases. Va del input al output partiendo pero dividiendo el razonamiento en pasos. De esta forma es más sencillo para el modelo llegar a conclusiones razonables.

Self Consistency with CoT (CoT-SC): En vez de tener un único CoT tiene múltiples, de esta forma a través de un voto mayoritario se asegura que al menos una de esas cadenas sea la que tiene el razonamiento correcto.

Estas técnicas se pueden utilizar para refinar el plan de acción del agente.

2. Técnicas de reflexión: unen el mundo de las técnicas que solo afectan al razonamiento (como CoT) y las que afectan a la actuación en un entorno externo (como agentes que utilizan Reinforcement Learning que actúan sobre un entorno, obtienen observaciones y vuelven a tomar otra decisión en función de las observaciones tomadas).

Ejemplos:

ReAct (Reason+Act): combina ambas aproximaciones. Trabaja con tres conceptos. Primero el "pensamiento inicial" del agente (**Thought**) que describe que tiene que hacer, luego viene la actuación en el entorno (**Act**) y por último la observación (**Obs**) de lo que ocurrió luego de actuar. Utilizando ReAct podríamos crear un agente inteligente que a partir de un prompt de entrada con un objetivo y dotándolo de las herramientas necesarias resuelva un problema por sí solo.

```
(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the
program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in
October 2005 by Apple ... originally designed to control the
Front Row media center program ...

Thought 2: Apple Remote was originally designed to control
the Front Row media center program. I need to search Front
Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row
Seat to Earth', 'Front Row Motorsports', 'Front Row
[software]', ...]

Thought 3: Front Row is not found. I need to search Front
Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software ...

Thought 4: Front Row (software) is controlled by an Apple
Remote or the keyboard function keys. So the answer is
keyboard function keys.
Act 4: Finish[keyboard function keys]
```

2. Memoria (Memory): Es un almacén de los registros internos del agente y de sus interacciones con humanos. Hay dos tipos de memoria:
Memoria a corto plazo (**Short-term memory**): contiene las acciones y pensamientos por los que pasa un agente para intentar responder una sola pregunta de un usuario.
Memoria a largo plazo (**Long-term memory**): contiene acciones y pensamientos sobre eventos que suceden, contiene un historial de conversaciones.
3. Herramientas (Tools): Son flujos de trabajo ejecutables que los agentes pueden utilizar para ejecutar tareas. Ej.: los agentes pueden utilizar RAG para generar respuestas contextuales, un intérprete de código para programar, una API para buscar información en internet, meteorológica o de mensajería instantánea. A través de las herramientas se realizan las acciones (4).

La combinación de estos elementos es lo que hace a un agente LLM.

Investigación respecto al estado del arte de las aplicaciones actuales de agentes inteligentes usando modelos LLM libres.

Agentes LangChain:

LangChain es un framework open source diseñado para desarrollar aplicaciones que utilizan LLM (entre ellos LLMs libres) entre las diversas aplicaciones se encuentra la posibilidad de crear agentes LLM.

La idea central de los agentes LangChain es utilizar un LLM para elegir una secuencia de acciones a realizar, utilizándolo como motor de razonamiento para determinar qué acciones realizar y en qué orden.

Componentes claves:

- *AgentAction*: Es una clase de datos que representa la acción que un agente elige para realizar. Tiene una propiedad *tool* (que es el nombre de la herramienta que debería ser invocada) y propiedad *tool_input* (que es la entrada de dicha herramienta)
- *AgentFinish*: Representa el resultado final de un agente, cuando el mismo está listo para ser devuelto al usuario. Contiene un diccionario *return_values* que contiene la salida final del agente. Generalmente este contiene una clave *output* que contiene un string que es la respuesta del agente.
- *IntermediateSteps*: Representa las acciones anteriores del agente y salidas correspondientes de la actual ejecución del agente. Es importante pasarlas a iteraciones futuras para que los agentes sepan que trabajo ya está hecho. Se escribe como una *List[Tuple[AgentAction, Any]]*. La observación anterior se deja como tipo *Any* para ser lo más flexible posible. En la práctica, generalmente es un string.

Agent:

Es el responsable de decidir que paso sigue. Generalmente funciona con un LLM, un prompt y un output parser. LangChain cuenta con varios tipos de agentes, con distintos estilos de prompting para el razonamiento, diferentes modos de codificar entradas, y diferentes modos de convertir las salidas.

Además permite construir agentes personalizados para tener un mayor control.

Entrada de los agentes: Las entradas para los agentes son un diccionario clave-valor que solo necesitan la clave *intermediate_steps* que son los *IntermediateSteps* explicados antes.

Salida de los agentes: Las salidas son las siguientes acciones a tomar o la respuesta final para enviar al usuario (*AgentAction* o *AgentFinish*). Estas se pueden ser del tipo *Union[AgentAction, List[AgentAction], AgentFinish]*. El *output parser* toma la salida del LLM cruda y la transforma en uno de esos tres tipos.

AgentExecutor:

Es la ejecución del agente. Es lo que realmente llama al agente, ejecuta la acción que elige, devuelve los resultados de la acción al agente y repite si es necesario.

Así funciona en pseudocódigo:

```
next_action = agent.get_action(...)
while next_action != AgentFinish:
    observation = run(next_action)
    next_action = agent.get_action(..., next_action, observation)
return next_action
```

Esta ejecución maneja varias complejidades no tan evidentes en la descripción anterior como casos donde el agente selecciona una herramienta inexistente o donde la herramienta falla, casos donde el agente produce resultados que no se pueden utilizar en una herramienta, entre otros.

Herramientas:

Son funciones a las que el agente puede invocar. La clase *Tool* consiste en dos componentes:

1. El esquema de entrada para la herramienta que le informa al LLM que parámetros se necesitan para llamar a la herramienta. Sin esto el LLM no sabría cuáles son las entradas correctas. Estos parámetros deben estar bien nombrados y descriptos.
2. La función a llamar, que generalmente es una función Python.

Es importante darle acceso al agente a las herramientas correctas y describirlas del modo que mejor ayude al agente. Sin estas consideraciones el agente no va a poder funcionar correctamente.

LangChain proporciona un gran conjunto de herramientas integradas, pero también facilita la creación de las propias.

También proporciona el concepto de caja de herramientas (toolkit) que son conjuntos de herramientas necesarias para cumplir determinados objetivos, como por ejemplo el GitHub toolkit que tiene herramientas para buscar entre issues de GitHub, leer un archivo, comentar, etc.

Funcionamiento de un agente LangChain:

1. Recepción de entrada: el agente recibe información en lenguaje natural del usuario.
2. Procesamiento con LLM: el agente emplea el LLM para procesar la entrada y formular un plan de acción.
3. Ejecución del plan: el agente ejecuta el plan de acción ideado, que puede implicar la interacción con otras herramientas o servicios.
4. Entrega de resultados: Posteriormente, el agente entrega el resultado del plan ejecutado al usuario.

Comprendiendo LangChain y su arquitectura basada en agentes, se puede aprovechar su potencial y crear aplicaciones inteligentes que procesen, y generen, texto en lenguaje natural de manera competente.

Entre las ventajas de este framework se encuentra su facilidad de usar incluso para desarrolladores con poco conocimiento acerca de LLMs.

Otros frameworks con los que se pueden crear agentes LLM con funcionalidades similares a las de LangChain son:

- LLaMaIndex
- HayStack

Babyagi:

Es un “experimento” de agente autónomo que nació en la primera mitad del 2023. Con experimento me refiero a que por el momento son versiones de prueba más que utilidades.

Es un script de Python que es un ejemplo de un sistema de gestión de tareas impulsado por inteligencia artificial. El sistema utiliza LangChain, un LLM y una base de datos vectorial (ambas dos se pueden configurar para utilizar open source como llama-2 y ChromaDB) para crear, priorizar y ejecutar tareas. La idea principal detrás de este sistema es que crea tareas basadas en el resultado de tareas anteriores y un objetivo predefinido. Luego, el script utiliza las capacidades de procesamiento del lenguaje natural (NLP) del LLM para crear nuevas tareas basadas en el objetivo, y la base de datos vectorial para almacenar y recuperar los resultados de las tareas para el contexto.

Básicamente se trata de un bucle infinito donde:

1. Extrae la primera tarea de la lista de tareas.
2. Envía la tarea al agente de ejecución, que utiliza el LLM para completar la tarea según el contexto.
3. Enriquece el resultado y lo almacena en la base de datos vectorial.
4. Crea nuevas tareas y prioriza la lista de tareas en función del objetivo y el resultado de la tarea anterior.

El sistema se ejecuta desde línea de comandos. Se le asigna un objetivo y una tarea inicial en un archivo de configuración y continua desde ahí.

Lo único que el usuario puede hacer es observar como continua. No tiene ningún tipo de intervención luego de que la aplicación inicia.

Por defecto tiene un objetivo predeterminado que es “resolver el hambre del mundo”.

Al tratarse de un experimento y trabajar con LLMs al ejecutarlo se pueden ver fácilmente las limitaciones. Por ejemplo con el objetivo inicial un usuario describe que una de sus listas iniciales de seis tareas incluía colaborar con los gobiernos del mundo para evaluar la producción de alimentos, establecer bancos de alimentos, ayudar a las personas a aprender a cultivar sus propios alimentos y abogar por políticas que aborden la pobreza y la desigualdad y el cambio climático. Y la pregunta es como una IA que se ejecuta en la pc podría hacer cualquiera de estas cosas.

Es necesario mencionar que Babyagi no tiene función para utilizar herramientas.

Sin embargo también al tratarse de tecnologías en fase experimental lo que destaca es el potencial a futuro.

AutoGPT:

AutoGPT es otra aplicación experimental en Python desarrollada por Significant Gravis. Se trata de un agente semiautónomo impulsado por LLM que ejecuta tareas. El concepto es dejar que el LLM decida qué hacer una y otra vez, mientras introduce los resultados de sus acciones en mensajes. Esto permite que el programa trabaje de forma iterativa e incremental hacia su objetivo. En este caso el usuario debe autorizar cada acción antes de realizar la tarea.

AutoGPT no fue diseñado para que cumpla una tarea específica en mente sino para ejecutar una amplia gama de tareas, realizables en una computadora, en múltiples disciplinas.

Hace el trabajo de crear sus propias tareas después de haberle asignado un conjunto inicial de objetivos. Utiliza GPT-3.5 o GPT-4 y se ejecuta desde línea de comandos. Es necesario nombrar al agente y asignarle una función que sea un objetivo amplio y un conjunto de metas.

Tiene la ventaja de poder conectarse a internet para recolectar datos.

Sobre cada paso muestra una serie de notas: Pensamiento, Razonamiento, Plan y Críticas. Pensamiento es lo que el agente quiere hacer a continuación, Razonamiento porque quiere hacer eso, Plan explica exactamente lo que va a hacer y las Críticas son autocríticas.

Nuevamente, al igual que Babiagi se dejan ver las limitaciones propias de los LLMs alucinaciones o el hecho de auto dotarse de funciones que no puede realizar.

ChatDev:



Los científicos están explorando el uso de marcos basados en aprendizaje profundo para cumplir tareas dentro del proceso de desarrollo de software y así mejorar la efectividad.

ChatDev es una simulación de una empresa virtual especializada en desarrollo de software que usa un enfoque basado en agentes LLM y busca la necesidad de eliminar modelos especializados en cada fase del desarrollo. Opera a través de varios agentes inteligentes que desempeñan diferentes roles, incluyendo al Director Ejecutivo, Director de Producto, Director

Tecnológico, Programador, Revisor, Tester y Diseñador de arte. Utiliza LLMs y su comunicación en lenguaje natural unifica y optimiza procesos claves.



Demostró ser efectivo y rentable para completar el proceso de desarrollo de software

Además identifica vulnerabilidades y rectifica posibles alucinaciones de código (causadas por la falta de especificación de tareas (falta del pensamiento guiado al que conducen analizar requerimientos de usuario o seleccionar un lenguaje de programación entre otros) y falta de contrainterrogatorio).

Adopta el modelo en cascada dividiendo el proceso en 4 fases: Diseño, Codificación, Pruebas y Documentación.

En cada etapa del proceso utiliza un equipo de agentes virtuales que colaboran entre sí mediante diálogos y resultan en un flujo de trabajo fluido. Utiliza una cadena de chat que divide cada etapa del proceso en subtarefas donde participan dos roles que hacen propuestas y validan soluciones a través de la comunicación.

Sistema multiagente:

La siguiente es una breve descripción de los agentes necesarios para solucionar un problema con un sistema multiagente y ejemplos de conversaciones deseables para solucionar dicho problemas. No se incluyen detalles técnicos de cómo llegar a dicha solución.

Problemática a solucionar:

Dotar a un LLM open-source de una herramienta de navegación web para poder realizar tareas que requieran buscar información en la web y poder responder a consultas del usuario que excedan a la información aprendida en su entrenamiento.

El resultado final podría resultar en un sistema con conexión a internet del tipo ChatGPT 4.0 pero open-source.

Solución:

Siempre es necesario un prompt inicial dado por un usuario humano cuya respuesta será el objetivo a cumplir por el sistema multiagente.

Agentes:

1. **Primer agente:** sirve de nexo entre el usuario humano y un segundo agente que tiene como herramienta acceso al navegador web.
Lo que hace es tomar la entrada del usuario humano, dividirla en tareas más pequeñas si es necesario, e ir guiando paso a paso al segundo agente a la solución del problema. Si no hay intervención humana es quien determina cuando el objetivo se cumplió, se termina la conversación entre agentes y se devuelve la respuesta al usuario sobre ese objetivo.
También podría solicitar que se agregue información humana ante una salida del segundo agente si es necesario para obtener más detalles, ejecutar código (para comprobar respuestas del segundo agente en consultas de programación).
Similar al UserProxyAgent de AutoGen.
2. **Segundo agente:** es el asistente que tiene como herramienta un navegador web. Su función es la de responder a las peticiones del primer agente y devolviendo las respuestas a las (sub) tareas que este le pide, en caso de ser una búsqueda web devuelve la url y el contenido de la solicitud.
Similar al WebSurferAgent de AutoGen (que tiene acceso un navegador con estado basado en texto simple -similar a Lynx- para buscar en la web, visitar páginas, navegar, etc.).

La dinámica de funcionamiento general del sistema multiagente seria:

1. El usuario humano ingresa un prompt al primer agente cuya respuesta se considera el objetivo a cumplir por el sistema multiagente.

2. El primer agente analiza el prompt inicial y lo subdivide en tareas más pequeñas con un orden de prioridad en caso de ser necesario.
3. A partir de una lista de (sub) tareas el primer agente le va solicitando al segundo agente que cumpla con la que mayor prioridad tenga.
4. El segundo agente cumple con la (sub) tarea indicada y devuelve la respuesta al primer agente.
5. El primer agente analiza la respuesta del segundo agente y si cumple con la tarea indicada en caso de haber más le indica la siguiente tarea.
6. Los pasos 4 y 5 se repiten hasta que el primer agente considere que el objetivo es cumplido y le devuelva la respuesta al usuario humano.

EJEMPLO

USUARIO

Usuario a primer agente:

Busca información en Wikipedia de quien fue Gustavo Cerati y resumela en dos párrafos cortos.

PRIMER AGENTE

El usuario ha solicitado buscar información sobre Gustavo Cerati en Wikipedia y resumirla. Mi tarea es dividir esta solicitud en subtareas y comunicarlas claramente al segundo agente para que las ejecute.

Subtareas:

1. Buscar información sobre Gustavo Cerati en Wikipedia.
2. Resumir la información encontrada en 2 párrafos cortos para proporcionársela al usuario final.

Primer agente a segundo agente:

Busca información sobre Gustavo Cerati en Wikipedia.

SEGUNDO AGENTE

Segundo agente a primer agente:

URL: https://es.wikipedia.org/wiki/Gustavo_Cerati

Contenido: Gustavo Adrián Cerati (Buenos Aires, 11 de agosto de 1959-ibid., 4 de septiembre de 2014), conocido como Gustavo Cerati, fue un músico, cantautor y productor discográfico argentino que obtuvo reconocimiento internacional por haber sido el líder, vocalista, compositor y guitarrista de la banda de rock Soda Stereo. Parte de la prensa especializada y músicos lo consideran como uno de los artistas más influyentes del rock latinoamericano.

Influenciado por las bandas británicas The Beatles y The Police, Cerati integró diversas agrupaciones desde su adolescencia y en 1982 fundó la banda de rock latino Soda Stereo. Líder y principal compositor del conjunto, a partir de Signos (1986) su forma de hacer canciones comenzó a madurar, y su consolidación la alcanzó a comienzos de los años 1990 con Canción animal, en el que volvía a las raíces del rock argentino de los años 1970. Paralelo a su carrera con el grupo, en 1992 publicó a dúo con Daniel Melero el álbum Colores santos, considerado uno de los primeros en Sudamérica en incluir música electrónica, y al año siguiente, el primero como solista, Amor amarillo. Su gusto por la electrónica lo llevó a incorporarla a sus últimos trabajos con Soda Stereo. Después de la separación de la banda, lanzó Bocanada (1999) y Siempre es hoy (2002), donde mostró más su interés por el género, que manifestó libremente en sus proyectos alternos Plan V y Ocio, con la edición de álbumes y presentaciones que le dieron mayor difusión a este tipo de música.

En 2006, retomó el estilo roquero con su cuarto disco Ahí vamos, que recibió elogios del público y de la crítica, dicho trabajo contiene algunos de sus éxitos en solitario como «Crimen», «La excepción», «Adiós», «Lago en el cielo» y «Me quedo aquí». En 2007, se reunió con Soda Stereo tras diez años de separación en una gira que reunió a más de un millón de espectadores. En 2010, quedó en estado de coma tras padecer un accidente cerebrovascular, luego de finalizar un concierto en el que promocionaba su último álbum, Fuerza natural (2009). Falleció a la edad de 55 años a causa de un paro respiratorio en 2014.

Cerati fue un prolífico sesionista, estuvo como guitarrista invitado en temas de Caifanes, Babasónicos y Los Brujos, y colaboró en canciones con Charly García, Andrés Calamaro, Shakira, Andy Summers, Roger Waters y Mercedes Sosa, entre otros. A lo largo de su carrera como solista, vendió más de siete millones de discos y ganó numerosos premios, entre ellos Grammy Latino, MTV, Konex y Gardel. En 2012, la versión argentina de la revista Rolling Stone posicionó a Cerati en el séptimo lugar de los cien mejores guitarristas del rock argentino. En 2023, la revista Billboard lo ubicó en el puesto número 33 de los 50 mejores cantantes de rock de todos los tiempos, mientras que la revista estadounidense Rolling Stone lo clasificó en el puesto 230 en su lista de los 250 mejores guitarristas de todos los tiempos.

....

PRIMER AGENTE

Recibí la respuesta del segundo agente con la información sobre Gustavo Cerati de Wikipedia.

Debo evaluar la información proporcionada por el segundo agente para garantizar que sea precisa y relevante antes de aceptarla como válida y proceder con la siguiente tarea.

Voy a revisar el contenido proporcionado por el segundo agente para confirmar que se refiere a Gustavo Cerati y que proviene de Wikipedia.

Después de revisar cuidadosamente el contenido proporcionado por el segundo agente, confirmo que se refiere a Gustavo Cerati y según la información de la URL parece provenir de Wikipedia.

Subtareas:

1. Resumir la información encontrada en 2 párrafos cortos para proporcionársela al usuario final.

Primer agente a segundo agente:

Resume la información sobre Gustavo Cerati en dos párrafos cortos.

SEGUNDO AGENTE

Segundo agente a primer agente:

Gustavo Adrián Cerati, nacido el 11 de agosto de 1959 en Buenos Aires y fallecido el 4 de septiembre de 2014, fue un músico, cantautor y productor discográfico argentino reconocido internacionalmente como líder, vocalista, compositor y guitarrista de la banda de rock Soda Stereo. Su influencia en el rock latinoamericano es ampliamente reconocida, y su carrera se destacó por su evolución musical y su incursión en la música electrónica. A lo largo de su vida, colaboró con una variedad de artistas y recibió numerosos premios, incluyendo Grammy Latino y MTV.

Cerati dejó un legado duradero en la música, con éxitos tanto en su carrera con Soda Stereo como en su carrera solista. Su trabajo fue ampliamente elogiado por la crítica y el público, y su impacto en la escena musical continúa siendo relevante hasta el día de hoy. Su talento como guitarrista y compositor lo llevó a ser reconocido como uno de los mejores músicos de rock de Argentina y del mundo, ocupando posiciones destacadas en listas de los mejores guitarristas y cantantes de todos los tiempos.

PRIMER AGENTE

El segundo agente ha proporcionado un resumen detallado y preciso sobre Gustavo Cerati, que parece ser relevante para la solicitud del usuario. Ahora puedo proceder a transmitir esta información al usuario final, cumpliendo así con la última subtask asignada.

Primer agente a usuario: Gustavo Adrián Cerati, nacido el 11 de agosto de 1959 en Buenos Aires y fallecido el 4 de septiembre de 2014, fue un músico, cantautor y productor discográfico argentino reconocido internacionalmente como líder, vocalista, compositor y guitarrista de la banda de rock Soda Stereo. Su influencia en el rock latinoamericano es ampliamente reconocida, y su carrera se destacó por su evolución musical y su incursión en la música electrónica. A lo largo de su vida, colaboró con una variedad de artistas y recibió numerosos premios, incluyendo Grammy Latino y MTV.

Cerati dejó un legado duradero en la música, con éxitos tanto en su carrera con Soda Stereo como en su carrera solista. Su trabajo fue ampliamente elogiado por la crítica y el público, y su impacto en la escena musical continúa siendo relevante hasta el día de hoy. Su talento como guitarrista y compositor lo llevó a ser reconocido como uno de los mejores músicos de rock de Argentina y del mundo, ocupando posiciones destacadas en listas de los mejores guitarristas y cantantes de todos los tiempos.

FIN EJEMPLO

Bibliografia:

<https://medium.com/@aydinKerem/what-is-an-llm-agent-and-how-does-it-work-1d4d9e4381ca>

<https://developer.nvidia.com/blog/introduction-to-llm-agents/>

<https://www.youtube.com/watch?v=2p5azZv81lA&t=796s>

<https://huggingface.co/blog/open-source-llms-as-agents>

<https://python.langchain.com/docs/modules/agents/>

<https://www.linkedin.com/pulse/introduction-langchain-agents-coditation-systems/>

<https://developer.nvidia.com/blog/building-your-first-llm-agent-application/>

<https://github.com/yoheinakajima/babyagi/blob/main/docs/README-es.md>

<https://www.youtube.com/watch?v=90464Ht7vhQ>

<https://www.tomshardware.com/news/autonomous-agents-new-big-thing>

<https://chatdev.ai/>

<https://arxiv.org/pdf/2307.07924.pdf>

<https://www.unite.ai/es/chatdev-communicative-agents-for-software-development/>

<https://docs.agpt.co/>

<https://microsoft.github.io/autogen/docs/Getting-Started/>