TRABAJO PRÁCTICO FINAL

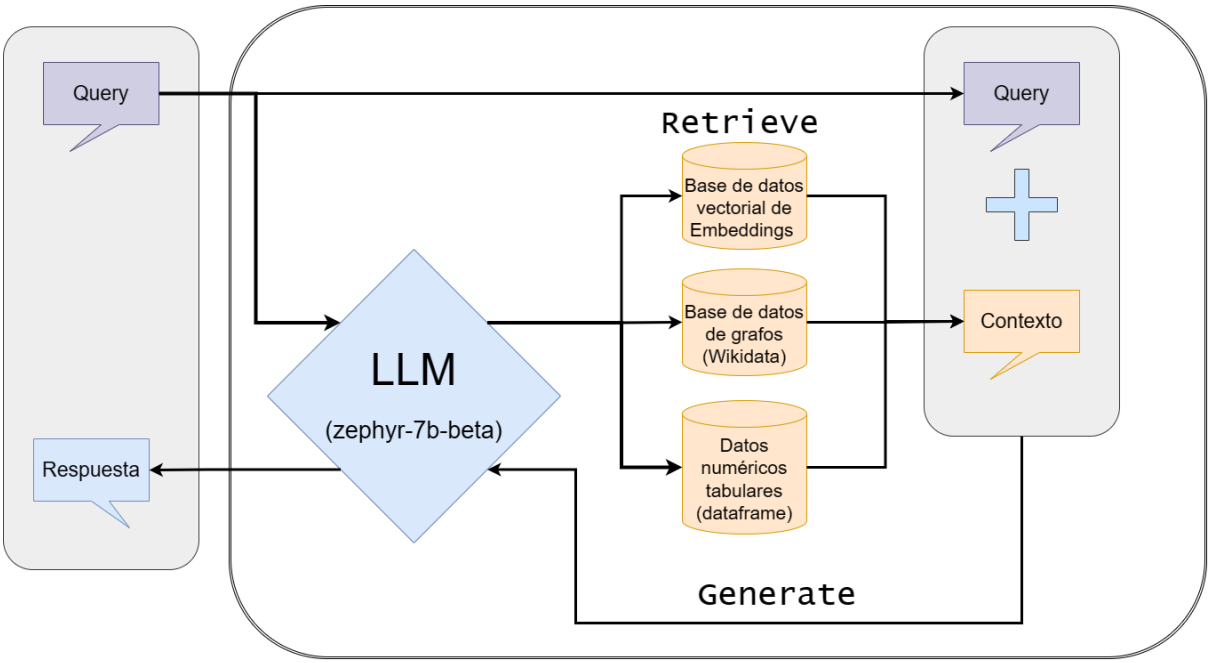
PROCESAMIENTO DEL LENGUAJE NATURAL

Tecnicatura Universitaria en Inteligencia Artificial
FECIA - UNR
18/12/23

Leandro Salvañá

1. Ejercicio 1: Implementación de RAG

El objetivo de este RAG (Retrieval Augmented Generation) es crear un chatbot experto en **Harry Potter**. El esquema general es el siguiente:



Se utilizará un LLM con un prompt con formato Few-Shot para clasificar qué fuente de datos utilizar dado el prompt introducido, luego se usan diversos métodos para traer el contexto adecuado de cada una de estas fuentes y finalmente se alimenta el LLM con el prompt del usuario sumado al contexto y las instrucciones de que no utilice su conocimiento previo sino el contexto brindado. El enunciado indica que el intercambio debe ser en idioma Español. Debido a que los dataset están en inglés y que los modelos en general funcionan mejor en este idioma, se decidió traducir la query al inglés, trabajar en este idioma y luego traducir al español la respuesta.

1.1. Fuentes de datos

Para comenzar la implementación del RAG se prepararon las fuentes de datos para brindar contexto a los prompts. Estas serán una base de datos vectorial ChromaDB llena con embeddings generados a partir de los siete libros de la saga principal de Harry potter, Wikidata que es una base de datos de grafos y datos tabulares ubicados en formato CSV.

1.1.1. Texto

Los libros de Harry Potter fueron descargados de un dataset de Kaggle y el texto se extrajo con la librería PyPDF2.

Se limpiaron los saltos de línea "ficticios" es decir, que existen sólo porque se llegó al extremo de la hoja del PDF y no porque tenga un sentido semántico. Esto es porque el splitter de texto de langchain utiliza los saltos de línea como guía para dividir el texto.

1.1.2. Datos numéricos tabulares

Los datos tabulares se obtuvieron de un dataset de Kaggle y consta de un CSV con información sobre las novelas físicas. Contienen información del nombre del libro, copias vendidas en el reino unido, copias vendidas en todo el mundo, fecha de publicación, mes de publicación, año de publicación, cantidad de páginas, cantidad de palabras, duración del audiolibro y cantidad de premios ganados. Los datos fueron cargados en un dataframe de Pandas y luego se hizo una función para imprimirlo entero en formato de texto con sentido semántico.

Quizás hubiera sido mejor dar como contexto sólo la información solicitada en el prompt. Pero por priorizar el tiempo de realización y dado el reducido tamaño del dataset, se optó por brindar todo el texto cuando el prompt hace referencia a las novelas físicas.

1.1.3. Base de datos de grafos

Se utilizó una base de datos de grafos online, siendo Wikidata la opción elegida. Se decidió que se brindará contexto de esta fuente únicamente cuando se pregunte por un personaje.

Para esto, se definió una función que pide a la base de datos el ID del sujeto del personaje al que el prompt hace referencia, y luego se le piden todos las etiquetas de objetos relativos al sujeto dada una lista de IDs de predicados previamente investigados.

```python
# ID de Albus Dumbledore en Wikidata
sujeto_label_ejemplo = "Albus Dumbledore"

# Obtener y mostrar los valores asociados a la propiedad
propiedades_personaje = obtener_valores_por_sujeto(sujeto_label_ejemplo)

print(propiedades_personaje)
```

```
Albus Dumbledore  :
sex or gender: male
country of citizenship: United Kingdom
name in native language: Albus Percival Wulfric Brian Dumbledore
birth name: Albus Percival Wulfric Brian Dumbledore
given name: Percival
given name: Brian
given name: Albus
given name: Wulfric
date of birth: 1881-01-01T00:00:00Z
place of birth: Mould-on-the-Wold
date of death: 1997-06-30T00:00:00Z
manner of death: homicide
cause of death: Avada Kedavra
killed by: Severus Snape
father: Percival Dumbledore
mother: Kendra Dumbledore
sibling: Aberforth Dumbledore
sibling: Ariana Dumbledore
```

1.1.4. Base de datos vectorial de embeddings

1.1.4.1. Dividir los textos

Una vez extraídos los textos de los libros desde los PDFs, se procedió a dividir los textos en fragmentos “chunks” con una longitud significativa para captar el contexto. Al tratarse de libros de narración de fantasía con mucha adjetivación se tomó un fragmento de longitud considerable: 1000 caracteres.

1.1.4.2. Limpieza de fragmentos

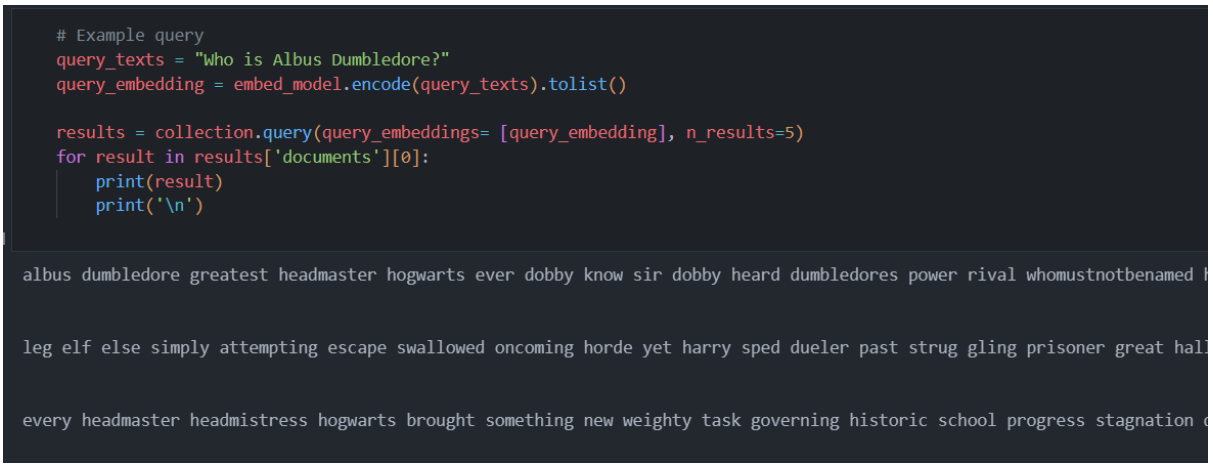
Para mejorar el desempeño del modelo de embeddings, se limpiaron los fragmentos del texto. Esta limpieza consistió en pasar todo el texto a minúsculas, la eliminación de símbolos y stopwords y lematizar.

1.1.4.3. Embeddings

Una vez limpios los fragmentos se utilizó el modelo de embeddings paraphrase-multilingual-mpnet-base-v2 para realizar la vectorización de los fragmentos.

1.1.4.4. Almacenar embeddings en una base de datos ChromaDB

Ahora, obtenidos los embeddings, se guardaron en una base de datos ChromaDB junto con los fragmentos sin limpiar. Para luego obtener los embeddings de un prompt y traer los vectores más cercanos como se observa en la siguiente imagen.



```
# Example query
query_texts = "Who is Albus Dumbledore?"
query_embedding = embed_model.encode(query_texts).tolist()

results = collection.query(query_embeddings= [query_embedding], n_results=5)
for result in results['documents'][0]:
    print(result)
    print('\n')
```

```
albus dumbledore greatest headmaster hogwarts ever dobby know sir dobby heard dumbledores power rival whomustnotbenamed

leg elf else simply attempting escape swallowed oncoming horde yet harry sped dueler past strug gling prisoner great hal

every headmaster headmistress hogwarts brought something new weighty task governing historic school progress stagnation
```

1.2. Implementación de RAG

1.2.1. Traducción de prompts y respuestas

Como se adelantó en la introducción, el enunciado pide que el intercambio sea en español, pero dado a que los dataset están en inglés y el mejor desempeño de los modelos en este idioma, se optó por traducir el prompt al inglés, trabajar en la lengua sajona y luego traducir la respuesta al español nuevamente.

Para llevar a cabo esto se utilizaron los modelos "opus-mt-es-en" y "opus-mt-es-en" de Helsinki-NLP disponibles en HuggingFace.

1.2.2. Clasificación de prompts para elegir la fuente de contexto

Para clasificar las preguntas y elegir la fuente para dar contexto, se utilizó un LLM específicamente "zephyr-7b-beta" diponible en Hugging face y un formato de prompt Few-Shot para obtener respuestas más exactas. De esta manera al introducir un prompt relacionado a un personaje, no sólo indicará que se trata de una pregunta sobre un personaje sino que además devuelve el nombre del personaje como se muestra en el ejemplo a continuación.

```
# Ejemplo)
prompt = 'Where does Minerva MacGonagall work?'
answer = choose_data_source(prompt, api_key=api_key)
print('\nRESPUESTA: \n', answer)
```

```
RESPUESTA:
 "Personaje: [Minerva McGonagall]" for questions or tasks about Minerva McGonagall's identity, such as where she works.
```

1.2.3. Interpretar la respuesta

Ya que se le pidió un formato de respuesta muy específico al LLM, se puede utilizar una función muy sencilla con expresiones regulares para obtener la información necesaria. La función formatea la información en una lista con dos elementos. El primero con el tema de la pregunta ("Personaje", "Novelas", "Historia") y el segundo en `None` a menos que sea una pregunta sobre un personaje, dónde también se obtendrá el personaje en cuestión y se colocará en esta posición para realizar la query en la base de datos de grafos. A continuación, un ejemplo:

```
# Ejemplo de uso:
respuesta_limpia = answer_cleaner(answer)
print(respuesta_limpia)
```

```
4]
['Personaje', 'Minerva McGonagall']
```

1.2.4. Devolver el contexto

Teniendo identificado el tipo de prompt y, por ende, la fuente que debe dar el contexto, se armó una función que se le pasa el binomio resultante de la función anterior y devuelve el texto predefinido del dataframe de los libros, vectoriza y busca el prompt en la ChromaDB o hace el proceso descrito anteriormente para obtener los datos de un personaje en la base de datos de grafos. Continuando con el ejemplo con el que se venía trabajando:

```python
    # Ejemplo
    contexto = devolver_contexto(respuesta_limpia, prompt)[0]

    print(contexto)
```

```
Minerva McGonagall  :
sex or gender: female
country of citizenship: United Kingdom
name in native language: Minerva McGonagall
given name: Minerva
date of birth: 1935-10-04T00:00:00Z
occupation: professor
occupation: employee
occupation: head teacher
occupation: school teacher
employer: Hogwarts
employer: Department of Magical Law Enforcement
position held: Headmaster of Hogwarts
member of: Order of the Phoenix
member of: Gryffindor
```

1.2.5. Preguntar al agente con el contexto agregado

El último paso es darle al mismo LLM que ayudó a clasificar el prompt, el prompt original y el contexto, y pedirle que no use conocimiento previo.

1.2.6. Resultados

A continuación se muestra un ejemplo para una situación dónde se debe utilizar cada una de las fuentes de datos.

```
--------------------------------------------------------------------
PREGUNTA:
Qué pasó en la cámara de los secretos?
--------------------------------------------------------------------
FUENTE ESCOGIDA: Base de datos vectorial de Embeddings
--------------------------------------------------------------------
CONTEXTO BRINDADO:
nooooooo scene whirled darkness became complete harry felt falling crash landed spreadeagl
--------------------------------------------------------------------
RESPUESTA:
En la cámara de los secretos, Harry encontó el diario de Riddle, se desmayó y se desplomó
```

```
--------------------------------------------------------------------------
PREGUNTA:
¿Cuáles son los hermanos de Ron Weasley?
--------------------------------------------------------------------------
FUENTE ESCOGIDA: Base de datos de grafos
--------------------------------------------------------------------------
CONTEXTO BRINDADO:
Ron Weasley  :
sex or gender: male
country of citizenship: United Kingdom
name in native language: Ron Weasley
given name: Ron
given name: Ronald
date of birth: 1980-03-01T00:00:00Z
place of birth: Devon
father: Arthur Weasley
mother: Molly Weasley
sibling: Ginny Weasley
sibling: Bill Weasley
sibling: Percy Weasley
sibling: Charlie Weasley
sibling: Fred Weasley
sibling: George Weasley
occupation: student
occupation: entrepreneur
...

--------------------------------------------------------------------------
RESPUESTA:
Los hermanos de Ron Weasley hijo Bill, Percy, Charlie, Fred y Ginny Weasley.
```

```
PREGUNTA:
Cuántas unidades se vendieron de "La cámara secreta"?
--------------------------------------------------------------------------
FUENTE ESCOGIDA: Dataframe
--------------------------------------------------------------------------
CONTEXTO BRINDADO:
Book Name: Harry Potter and the Sorcerer's Stone
Copies Sold in UK: 4.2 millons
Copies Sold WorldWide: 120 millons
Publish Date: 26 June 1997
Pages: 223
Words: 76,944
Audiobook: 9 hrs and 33 mins
Total Awards Won: 8

Book Name: Harry Potter and the Chamber of Secrets
Copies Sold in UK: 3.5 millons
Copies Sold WorldWide: 77 millons
Publish Date: 2 July 1998
Pages: 251
Words: 85,141
Audiobook: 11 hrs and 5 mins
Total Awards Won: 5
...

--------------------------------------------------------------------------
RESPUESTA:
Se vendieron 3.5 millones de unidades de "La cámara secreta" en el Reino Unido, y se vendieron 77 millones de unidades
```

2. Ejercicio 2: Agentes Inteligentes

2.1. ¿Qué es un agente inteligente impulsado por LLM?

Para responder a la pregunta, podemos imaginar que el LLM (Large Language Model) es el cerebro, y puede llamar a diferentes agentes o herramientas, que son sus manos y pies, para realizar automáticamente una amplia gama de tareas.

Los agentes de LLM pueden dividir las tareas grandes y complejas en pasos más pequeños y manejables. También son capaces de reflexionar y aprender de las acciones y errores del pasado, con el fin de optimizar los pasos futuros y mejorar los resultados finales.

Su destreza radica en comprender lenguaje natural (NLU), lo que permite una multitud de aplicaciones. Con los LLM, ahora podemos comunicarnos con las máquinas tal como lo haríamos con otro humano.

2.2. ¿Por qué se destacan los agentes de LLM?

Dominio del lenguaje
Los LLM han aportado una nueva ventaja a través de sus capacidades de comprensión del lenguaje natural (NLU). Esto hace que la interacción hombre-máquina real sea una realidad.

Perspicacia para la toma de decisiones
Los LLM están equipados para razonar y decidir, lo que los hace expertos en resolver problemas intrincados.

Flexibilidad
Su adaptabilidad garantiza que se puedan moldear para diversas aplicaciones.

Interacciones colaborativas
Pueden colaborar con humanos u otros agentes, allanando el camino para interacciones multifacéticas.

Los LLM por sí solos no pueden realizar una amplia gama de aplicaciones en el mundo real. Para liberar todo el potencial de los LLM, necesitamos construir sistemas que puedan adquirir y aplicar conocimientos para resolver problemas prácticos. Ahí es donde entra en juego el sistema de agente autónomo impulsado por LLM. Sin agentes ni herramientas, los LLM son como cerebros en frascos: impresionantes pero aislados del mundo real.

2.3. Casos de uso para agentes de LLM

Aplicaciones de un solo agente
Los agentes de LLM se pueden utilizar como asistentes personales para ayudar a los usuarios a liberarse de las tareas diarias y el trabajo repetitivo. Pueden analizar, planificar y resolver problemas de forma independiente, lo que reduce la presión laboral de las personas y mejora la eficiencia en la resolución de tareas. A continuación, algunos ejemplos:

- **Cambiar la reserva de vuelo de un cliente**
  Aquí, el agente de LLM primero debe comprender qué información se necesita, como la política de cambios de la aerolínea y los vuelos alternativos disponibles (planificación). A continuación, puede llamar a herramientas como API de documentación y bases de datos de vuelos para recopilar los detalles necesarios (uso de herramientas).

- **Transferencia de la llamada de un paciente del hospital**
  El agente de LLM necesita comprender qué departamento solicitó el paciente (memoria a corto plazo: comprensión en contexto). A continuación, puede comprobar el horario de oficina, los números de guardia y otra información relevante (planificación y uso de herramientas). Esto permite conectar dinámicamente al paciente con el destino correcto, es decir, la extensión del departamento durante el horario de oficina, o el médico de guardia durante el horario de oficina.

Sistemas multiagente
Los agentes de LLM pueden interactuar entre sí de manera colaborativa o competitiva. Esto les permite avanzar a través del trabajo en equipo o las interacciones adversarias. En estos sistemas, los agentes pueden trabajar juntos en tareas complejas o competir entre sí para mejorar su rendimiento.

Cooperación Persona-Agente
Los agentes de LLM pueden interactuar con los humanos, brindándoles asistencia y realizando tareas de manera más eficiente y segura. Pueden entender la intención humana y adaptar su comportamiento para brindar un mejor servicio.

Dominios especializados
Los agentes de LLM pueden capacitarse y especializarse para dominios específicos, como el desarrollo de software, la investigación científica u otras tareas específicas de la industria. Pueden aprovechar su formación previa en corpus a gran escala y su capacidad para generalizar a nuevas tareas para proporcionar experiencia y apoyo en estos dominios.

2.4. AutoGen

AutoGen es un marco que permite el desarrollo de aplicaciones LLM utilizando múltiples agentes que pueden conversar entre sí para resolver tareas. Los agentes de AutoGen son personalizables, conversables y permiten la participación humana sin problemas. Pueden operar en varios modos que emplean combinaciones de LLM, entradas humanas y herramientas.

Algunos de los agentes integrados en AutoGen son:

- **AssistantAgent**
  Se trata de un agente que actúa como asistente de IA, utilizando LLM de forma predeterminada, pero que no requiere la intervención humana ni la ejecución de código. Puede escribir código, generar texto, responder preguntas y más, en función de los mensajes que recibe de otros agentes.

- **UserProxyAgent**
  Este es un agente que actúa como un proxy para los humanos, solicitando información humana a medida que la respuesta del agente en cada interacción cambia de forma predeterminada y también tiene la capacidad de ejecutar código y llamar funciones. También puede utilizar LLM para generar respuestas cuando no se proporciona información humana o no se realiza la ejecución del código.

- **CodeExecutionAgent**
  Este es un agente que ejecuta código y devuelve los resultados a otros agentes. También puede manejar errores y excepciones en el proceso de ejecución del código.

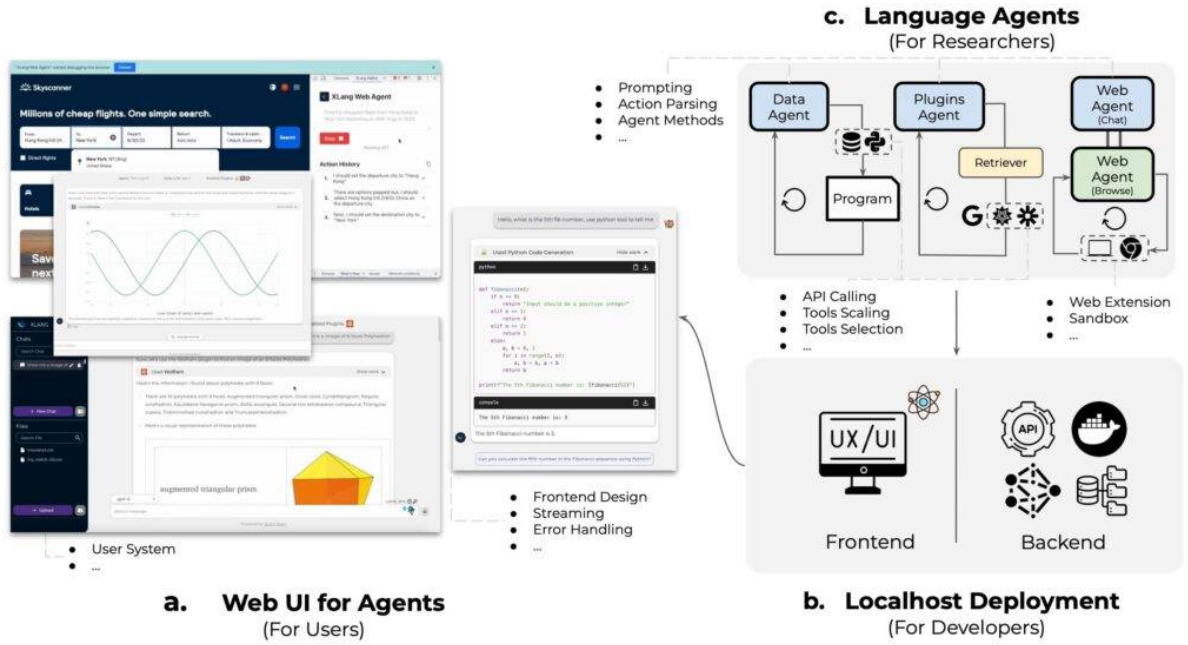
- **TeachableAgent**
  Este es un agente que aprende de los comentarios humanos y mejora su rendimiento con el tiempo. También puede hacer preguntas a otros agentes o humanos para aclarar sus dudas o aprender nuevos conceptos.

- **Agente RAG**
  El agente Proxy de usuario con recuperación aumentada recupera fragmentos de documentos relevantes en función de la similitud de

incrustación y los envía junto con la pregunta al agente Asistente con recuperación aumentada.

2.5. Proyectos de agentes LLM de código abierto

2.5.1. OpenAgents



OpenAgents es una plataforma abierta para utilizar y alojar agentes lingüísticos en la vida cotidiana. Los agentes lingüísticos son sistemas que pueden comprender y comunicarse en lenguaje natural, como chatbots, asistentes de voz o IA conversacional.

OpenAgents tiene como objetivo facilitar el desarrollo y la implementación de agentes lingüísticos para diversas tareas del mundo real, como análisis de datos, navegación web o herramientas diarias. OpenAgents también proporciona una interfaz de usuario web para que los usuarios interactúen con los agentes y un diseño modular para que los desarrolladores e investigadores integren diferentes modelos de lenguaje y herramientas.

Se puede obtener más información sobre OpenAgents en su repositorio de GitHub:
xlang-ai/OpenAgents: OpenAgents: An Open Platform for Language Agents in the Wild (github.com)
y en su paper:
OpenAgents: An Open Platform for Language Agents in the Wild | Papers With Code

2.5.2. ChatDev

ChatDev es un proyecto que tiene como objetivo crear software personalizado utilizando una idea de lenguaje natural (a través de la colaboración de múltiples agentes impulsada por LLM). Se basa en la idea de agentes comunicativos para el desarrollo de software, que es un paradigma novedoso que aprovecha los grandes modelos de lenguaje (LLM) durante todo el proceso de desarrollo.

ChatDev consta de varios agentes inteligentes que desempeñan diferentes roles, como Chief Executive Officer (CEO), Chief Product Officer (CPO), Chief Technology Officer (CTO), programador, reviewer, tester, diseñador de arte, etc. Estos agentes forman una estructura organizativa de múltiples agentes y colaboran participando en programas especializados. seminarios funcionales, como diseño, codificación, testing y documentación.

ChatDev ofrece un marco fácil de usar, altamente personalizable y extensible para estudiar inteligencia colectiva y generar software a través de la comunicación en lenguaje natural.

Más información en su sitio web: ChatDev (toscl.com) o en su repositorio de GitHub: 10cl/chatdev: ChatDev IDE is an tools for building your ai agent, Whether it's NPCs in games or powerful agent tools, you can design what you want for this platform. (github.com)

2.5.3. Chaindesk

Chaindesk es una plataforma sin código que te permite crear su propio chatbot de IA personalizado y entrenado con tus datos en segundos. Puede cargar datos de cualquier fuente, como texto, páginas web, archivos o sitios web, y luego hacer preguntas, extraer información y resumir documentos con IA.

También puedes integrar tu chatbot de IA en varias plataformas, como Slack, Whatsapp o tu propio sitio web. Chaindesk admite más de 100 idiomas y utiliza el modelo text-embedding-ada-002 de OpenAI para la búsqueda semántica.

Se puede obtener más información sobre Chaindesk en su sitio web: Chaindesk - ChatGPT AI Chat Bot for your business o en su repositorio de GitHub: gmpetrov/databerry: The no-code platform for building custom LLM Agents (github.com).

2.5.4. Superagent

Superagent es un marco que permite a los desarrolladores crear asistentes de inteligencia artificial que puedan interactuar con los usuarios utilizando

lenguaje natural y aprovechar el poder de los modelos de lenguaje grandes (LLM) como GPT-3. Superagent también proporciona una plataforma en la nube que permite a los desarrolladores implementar sus asistentes de IA en producción sin preocuparse por la infraestructura, las dependencias o la configuración.

Superagent es un proyecto de código abierto. Se puede encontrar más información sobre Superagent en su sitio web: Superagent | Docs o en su repositorio de GitHub: homanp/superagent: 🥷 The open framework for building AI Assistants (github.com).

2.6. Conlusión

La evolución de la IA, especialmente el auge de los agentes LLM, significa un cambio monumental en el ámbito digital. Estos agentes, con su capacidad de comprender, crear e interactuar, no son sólo herramientas sino colaboradores potenciales en diversos ámbitos.

De cara al futuro, se espera entrar en una era que es llamada "Software Empresarial 2.0". En la era del software empresarial 2.0, los agentes de LLM se convertirán en el centro de mando central de todo el sistema de software empresarial. En el sistema, los LLM son expertos de la industria y tomadores de decisiones. Comprenderán el conocimiento empresarial específico del dominio y llamarán dinámicamente a diferentes herramientas para automatizar la finalización de tareas. El ecosistema tiene comunicación y operación de circuito cerrado. Para entonces, los agentes de LLM tendrán la capacidad de ayudar a resolver problemas intrincados en diversas industrias y aprender por sí mismos de su propia experiencia.

2.7. Problemática

2.7.1. Problemática a Solucionar con el Sistema Multiagente

La problemática que se va a abordar con el sistema multiagente es automatizar la interacción entre un agente asistente (capaz de generar código) y un agente proxy de usuario (encargado de ejecutar el código y proporcionar retroalimentación como GPT-4). La tarea específica podría ser cualquier fin que se pueda lograr mediante la programación en Python. Podría ser hacer un análisis de datos a un dataset de clientes de un banco o consultar y comparar los cambios en los precios de las acciones YTD (Year-To-Date) de las empresas META y TESLA mediante la consulta de librerías financieras de Python.

2.7.2. Definición de Agentes Involucrados

Agente Asistente

- Es un agente basado en un modelo de lenguaje (LLM) capaz de generar código Python en respuesta a las solicitudes del usuario.
- Su función principal es proponer soluciones codificadas para la tarea encomendada.
- Puede analizar y depurar código, así como generar nuevo código en caso de errores.

Agente Proxy de Usuario

- Actúa como intermediario entre el usuario y el agente asistente.
- Ejecuta el código proporcionado por el agente asistente y devuelve la salida o resultados al mismo.
- Determina la terminación de la conversación.

2.7.3. Esquema del Sistema Multiagente

El sistema multiagente sigue un flujo de interacción donde el agente asistente y el agente proxy de usuario se comunican para resolver la tarea encomendada. Aquí está el esquema general del sistema:

1. **Inicio de Conversación**
   - El agente proxy de usuario inicia la conversación con el agente asistente, proporcionando una tarea específica.

2. **Generación y Ejecución de Código**
   - El agente asistente propone soluciones codificadas para la tarea encomendada, generando código Python.
   - El agente proxy de usuario ejecuta el código proporcionado y recopila la salida.

3. **Retroalimentación y Depuración**
   - Si la ejecución del código es exitosa, el agente proxy de usuario devuelve la salida al agente asistente.
   - Si hay errores en la ejecución, el agente proxy de usuario proporciona información de error al agente asistente, que a su vez puede generar nuevo código corregido.

4. **Iteración**
   - La comunicación entre el agente asistente y el agente proxy de usuario continúa iterativamente hasta que se completa la tarea o se alcanza un límite predefinido de respuestas automáticas.

2.8. Fuentes

1. Zhang, J. (2023). *How LLM Agents are Unlocking New Possibilities*. Wiz Holdings. Recuperado de How LLM Agents are Unlocking New Possibilities - WIZ AI

2. Mansouri, B. (6 de diciembre de 2023). *What Are LLM Agents? An Overview of Their Capabilities*. GPT Pluginz. Recuperado de What Are LLM Agents ? An Overview of Their Capabilities (gptpluginz.com)