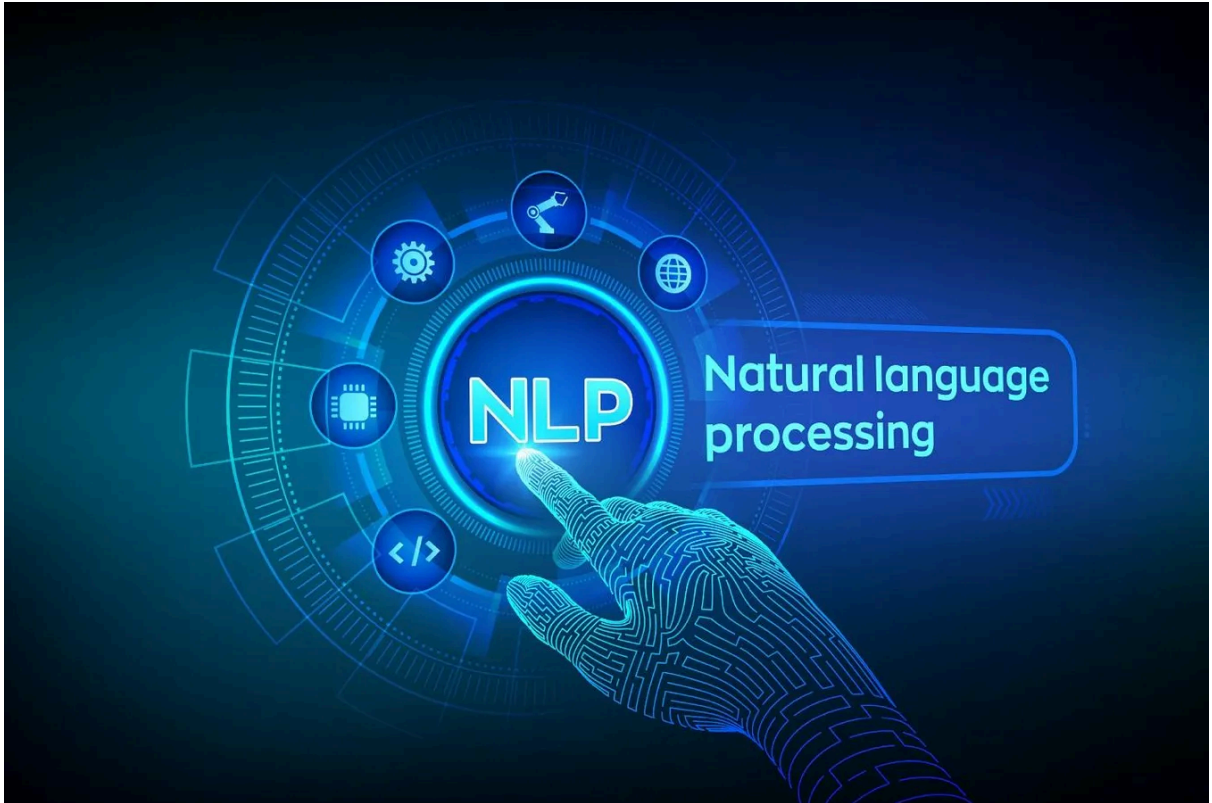


Trabajo Práctico Final

Procesamiento Natural del Lenguaje

TUIA



Alumno: Loza, Santiago

Docentes:

- Alan Geary
- Ariel D'Alessandro
- Juan Pablo Manson
- Andrea Carolina Leon Cavallo



ÍNDICE

INTRODUCCIÓN	pág. 2
RESOLUCIÓN EJERCICIO 1	pág. 3
RESOLUCIÓN EJERCICIO 2	pág. 12
CONCLUSIONES FINALES	pág. 17

INTRODUCCIÓN

En el presente informe se detallan los pasos y los conceptos aplicados para la resolución del Trabajo Práctico Final de la asignatura, el cual consta de dos ejercicios.

En el primer ejercicio se planea desarrollar un chatbot que, a partir de documentos propios, pueda cumplir el rol de **RAG**, de esta forma podremos obtener una respuesta inmediata sobre información de valor que tenemos almacenada.

Para ello se llevará a cabo un Chatbot experto en **Fútbol**.

Lo alimentaremos con información obtenida a través de 3 fuentes de datos, de cada una, obtendremos distinta información relacionada al fútbol.

A continuación, detallamos el tipo de información proveniente de cada base de datos:

- **Datos tabulares:** Resultados de todos los mundiales de la FIFA
- **Base de Grafos :** Información de qué equipo eliminó a qué equipo en la instancia final de la Copa de la Liga 2023 Argentina.
- **Base de datos vectorial ChromaDB:** Información del reglamento oficial de la FIFA.

En el segundo ejercicio, debemos realizar una investigación respecto al estado del arte de las aplicaciones actuales de **agentes inteligentes** usando modelos LLM libres.

Plantearemos una problemática a solucionar con un sistema multiagente y definiremos a cada uno de los agentes involucrados en la tarea.

El informe está organizado en secciones, cada una correspondiente a la solución detallada de cada ejercicio. Al final del documento, se presenta una conclusión general derivada de los resultados obtenidos en ambos ejercicios.

El primer ejercicio, que requiere código, lo llevaremos a cabo en **Google Colab**.

Aclaración: El informe del ejercicio 1 resume los puntos principales y las decisiones claves que se tomaron. Para profundizar en el código se debe visualizar en el siguiente Google Colab:

<https://colab.research.google.com/drive/1GPgRNTkkw-hj8-gl6Xr9UYYZuO2Amswn?usp=sharing>

EJERCICIO 1

Comenzamos importando las librerías necesarias para llevar a cabo el ejercicio. Luego de ello, importamos el excel **"Resultados Mundial.xlsx"**, este excel lo usaremos como nuestra fuente de **datos tabulares** y contiene los resultados de todos los partidos de los Mundiales FIFA.

Observamos las variables que tiene nuestro dataframe:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1930-07-13	Bélgica	Estados Unidos	0	3	Mundial FIFA	Montevideo	Uruguay	True
1	1930-07-13	Francia	Mexico	4	1	Mundial FIFA	Montevideo	Uruguay	True
2	1930-07-14	Brasil	Yugoslavia	1	2	Mundial FIFA	Montevideo	Uruguay	True
3	1930-07-14	Perú	Francia	1	3	Mundial FIFA	Montevideo	Uruguay	True
4	1930-07-15	Argentina	Francia	1	0	Mundial FIFA	Montevideo	Uruguay	True

Luego, creamos una función llamada **"find_all_matches"** que se encargará de transformar los datos de nuestro Dataframe en información. Almacenaremos dicha información en una lista llamada **"lista_match_info"**

Por ejemplo:

El registro

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1930-07-13	Bélgica	Estados Unidos	0	3	Mundial FIFA	Montevideo	Uruguay	True

Lo convertiremos en:

“El 1930-07-13, se disputó el encuentro entre las selecciones: Bélgica - Estados Unidos, por el Mundial FIFA 1930. El resultado fue 0-3.El encuentro se llevó a cabo en Uruguay, Montevideo. Copa Mundial FIFA disputada en el país: Uruguay”

Una vez almacenada la información proveniente de los datos tabulares en una lista, continuaremos con los datos de la **base de grafos**.

Creamos una función llamada **"extraer_texto_pdf"** que le podremos ingresar un PDF y nos devolverá una variable con todo el texto del mismo.

Una vez extraída la información de nuestro PDF llamado **"Copa de la Liga 2023"**, la trabajaremos, separando las oraciones en una lista llamada **"lista_oraciones"**.

La lista nos quedará de la siguiente manera:

```
[ 'Huracán terminó en el primer lugar de la Zona A con 26 puntos',
  'RiverPlate terminó en el segundo lugar de la Zona A con 24 puntos',
  'Banfield terminó en el tercer lugar de la Zona A con 23 puntos',
  'RosarioCentral terminó en el cuarto lugar de la Zona A con 23 puntos',
  'Racing terminó en el primer lugar de la Zona B con 24 puntos',
  'GodoyCruz terminó en el segundo lugar de la Zona B con 22 puntos',
  'Belgrano terminó en el tercer lugar de la Zona B con 21 puntos',
  'Platense terminó en el cuarto lugar de la Zona B con 20 puntos',
  'Platense eliminó por penales a Huracán',
  'GodoyCruz eliminó por penales a Banfield',
  'RiverPlate eliminó en los 90 minutos a Belgrano',
  'RosarioCentral eliminó por penales a Racing',
  'Platense eliminó por penales a GodoyCruz',
  'RosarioCentral eliminó por penales a RiverPlate',
  'RosarioCentral venció en los 90 minutos a Platense',
  'RosarioCentral es el actual campeón de la Copa de la Liga 2023']
```

Crearemos una base de grafos y añadiremos la información que es de nuestro interés a la misma. La información que nos interesa, es qué equipo eliminó a qué equipo. Una vez cargada la información, haremos una consulta a la base para ver que se haya cargado de manera correcta.

```
# Realizamos una consulta para obtener la informacion de nuestra BDG
q1 = """
    SELECT ?s ?p ?o WHERE {
        ?s ?p ?o .
    }
    """

#Ejecutamos la consulta
results = g.query(q1)

#Lista donde almacenaremos los resultados
resultados_grafos = []

# Iteramos sobre los resultados y la informacion en una lista
for r in results:
    res = f"{r['s']} {r['p']} a {r['o']} en la Copa de la Liga 2023.\n"
    print(res)

Platense eliminó a GodoyCruz en la Copa de la Liga 2023.

RosarioCentral venció a Platense en la Copa de la Liga 2023.

RosarioCentral eliminó a Racing en la Copa de la Liga 2023.

Platense eliminó a Huracán en la Copa de la Liga 2023.

RiverPlate eliminó a Belgrano en la Copa de la Liga 2023.

RosarioCentral eliminó a RiverPlate en la Copa de la Liga 2023.

GodoyCruz eliminó a Banfield en la Copa de la Liga 2023.
```

Como podemos observar, hemos cargado de manera correcta la información de nuestro interés a la base de grafos.

Luego de ya tener la información proveniente de la base de grafos, continuaremos con los datos provenientes de la base **ChromaDB**.

La información que almacenaremos en nuestra base de datos **ChromaDB**, se encuentra subida a un dropbox en formato ZIP y contiene un PDF, el cual, es el reglamento oficial de fútbol proporcionado por la FIFA (144 páginas)

Adjunto link:

https://www.dropbox.com/scl/fi/4n7xpxf3vm7aj6jgybsq1/reglas_fifa.zip?rlkey=2e87qu97bih7mhe0nxlqzob1s&dl=0.

En primer lugar, extraeremos el ZIP del dropbox, y luego leeremos nuestro PDF utilizando la librería **"langchain.documents_loaders"**.

```
# Cargamos y procesamos el archivo
loader = DirectoryLoader('./reglas_fifa', glob="/*.pdf", loader_cls=PyPDFLoader)

#Cargamos los documentos
documents = loader.load()
```

Esto, nos permite cargar y almacenar nuestra información de la siguiente manera:

Podemos ver que la variable **"documents"**, es una lista con información de cada una de las páginas.

```
documents[10]

Document(page_content='9 REGLA 1 - EL TERRENO DE JUEGO\nMetas\nLas metas se colocarán en el centro de cada línea de meta.Consistirán en dos postes verticales, equidistantes de los banderines de esquina \ny unidos en la parte superior por una barra horizontal (travesaño). Los postes y el travesaño deberán ser de madera, metal u otro material aprobado. Deberán tener forma cuadrada, rectangular, redonda o elíptica y no deberán constituir ningún peligro para los jugadores.\nLa distancia entre los postes será de 7.32 m y la distancia del borde inferior del \ntravesaño al suelo será de 2.44 m.\n7.32 m (8 yd)2.44 m\n(8 ft)', metadata={'source': 'reglas_fifa/reglas_fifa.pdf', 'page': 10})
```

Además, nos permite acceder directamente al contenido

```
documents[10].page_content

'9 REGLA 1 - EL TERRENO DE JUEGO\nMetas\nLas metas se colocarán en el centro de cada línea de meta.Consistirán en dos postes verticales, equidistantes de los banderines de esquina \ny unidos en la parte superior por una barra horizontal (travesaño). Los postes y el travesaño deberán ser de madera, metal u otro material aprobado. Deberán tener forma cuadrada, rectangular, redonda o elíptica y no deberán constituir ningún peligro para los jugadores.\nLa distancia entre los postes será de 7.32 m y la distancia del borde inferior del \ntravesaño al suelo será de 2.44 m.\n7.32 m (8 yd)2.44 m\n(8 ft)'
```

Y nos permite acceder a los metadatos que nos detallan el nombre del PDF y la página en la que se encuentra

```
documents[10].metadata

{'source': 'reglas_fifa/reglas_fifa.pdf', 'page': 10}
```

Una vez hecho esto, definimos una función "**limpieza**" que dado un texto eliminará las **stopwords** (palabras de parada), convertirá a minúsculas y llevará las palabras a su formato base.

Aplicamos dicha función a los contenidos de todas las páginas y luego realizamos el split de textos utilizando "**RecursiveCharacterTextSplitter**".

```
#Normalizamos los textos quitandole las stopwords y pasandolo a minusculas
for document in documents:
    document.page_content = limpieza(document.page_content)

# Dividimos los documentos limpios utilizando RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=800,
    chunk_overlap=150,
    is_separator_regex = True
)

texts = text_splitter.transform_documents(documents)
```

Luego, cargamos nuestro modelo de embeddings:

```
#Cargamos nuestro modelo de embeddings
embed_model = SentenceTransformer('intfloat/multilingual-e5-large')
```

Y aplicamos embeddings a todos nuestros textos generados recientemente.

Ahora, crearemos nuestra base vectorial **ChromaDB** y almacenaremos la información de la siguiente manera:

```
#Creamos la lista con nuestros ids
ids_docs = [f"doc{i}" for i in range(1, len(texts)+1)]

#Cremos una variable id que iremos iterando para que vaya accediendo a los elementos de la lista
id = 0

#iteramos sobre los textos procesados
for text in texts:
    collection.add(

        #Embeddings
        embeddings= embedding(text.page_content),

        #Cuerpo de cada pagina
        documents=text.page_content,

        #Metadatos
        metadata= {
            'source': text.metadata["source"],
            'page': text.metadata["page"],
        },

        #Ids
        ids = ids_docs[id]
    )

#Aumentamos el id que hace referencia a la pagina
id += 1
```

Una vez insertada la información dentro de la base **ChromaDB**, haremos una consulta a la misma.

```
#Probamos una consulta a nuestra base de datos ChromaDB
consulta = "¿Cuándo se concederá un tiro libre directo al equipo adversario?"
embedding_consulta = embedding(consulta)

results = collection.query(
    query_embeddings=embedding_consulta, # Consulta pasada como embedding
    n_results=3 # Traemos los 3 resultados más cercanos
)

# Imprimir los documentos y sus distancias
for id, doc, distancia in zip(results["ids"], results["documents"], results["distances"]):
    print(f"IDs: {id}")
    print(f"Documento: {doc}")
    print(f"Distancia: {distancia}")
    print('-' * 40)

IDs: ['doc73', 'doc89', 'doc88']
Documento: ['adversario si jugador comete siguientes tres infracciones:• sujetar adversario • escupir adversario • tocar balón deliberadamente man
Distancia: [0.25709542632102966, 0.2584502398967743, 0.2628633379936218]
-----
```

Una vez que hayamos obtenido y procesado la información de todas las bases de datos, construiremos un clasificador. Este clasificador será el encargado de determinar a qué base de datos se debe consultar según lo que quiera saber el usuario.

Utilizaremos **"TfidfVectorizer"** como vectorizador y un modelo de **Regresión Logística** que llevará a cabo la clasificación.

De esta manera funcionará el clasificador



Además, definimos nuestras etiquetas de la siguiente manera:

0 → Reglamento

1 → Copa de la Liga 2023

2 → Resultados Mundiales

Una vez entrenado nuestro modelo de clasificación con las etiquetas necesarias, probaremos su funcionamiento.

```
# Nueva frase para clasificar
nueva_frase = "Segun el reglamento de la Fifa ¿Cuándo se saca una tarjeta roja?"

# Clasificamos la nueva frase
clasificador(nueva_frase)
```

0


```
# Nueva frase para clasificar
nueva_frase = "En la Copa de la Liga 2023 ¿Quien eliminó a Racing?"

# Clasificamos la nueva frase
clasificador(nueva_frase)
```

1

```
# Nueva frase para clasificar
nueva_frase = "En el Mundial FIFA 1886 ¿Cual fue el resultado de Argentina vs Inglaterra?"

# Clasificamos la nueva frase
clasificador(nueva_frase)
```

2

Se puede observar el buen funcionamiento del clasificador.

Una vez realizado el clasificador, llevaremos a cabo el desarrollo de nuestro Modelo **LLM**.

Como modelo LLM elegiremos el modelo "**Zephyr 7B Beta**" de Hugging Face.

<https://huggingface.co/HuggingFaceH4/zephyr-7b-beta>

Se eligió dicho modelo ya que es un modelo muy completo, su implementación es sencilla y da buenos resultados.

Los parámetros se han seteado a base de prueba y error. Con los parámetros tal como están obtuvimos el mejor rendimiento de nuestro modelo.

En primer lugar, se define la función "**requests_modelo**". Esta función se encarga de realizar una solicitud al modelo de Hugging Face.

Por otro lado, la función "**llm**" se ha diseñado con todos los parámetros predefinidos necesarios para llevar a cabo consultas con el contexto adecuado. Esta función se utiliza para interactuar con el modelo de lenguaje, permitiendo realizar consultas utilizando el contexto establecido.

A continuación, podemos observar que el modelo funciona de la manera esperada:

```
#Probamos el modelo pasandole un contexto manualmente
contexto = "Rosario Central vencio por 1 a 0 a Platense en la Final de la Copa de la Liga 2023 y se consagro Campeon."
pregunta = "Quien salio Campeon de la Copa de la Liga 2023?"
respuesta = llm(contexto, pregunta)

print(f'Pregunta usuario: {pregunta}')
print(f'AI respuesta: {respuesta}')
```

Pregunta usuario: Quien salio Campeon de la Copa de la Liga 2023?
AI respuesta: Rosario Central se consagro Campeon de la Copa de la Liga 2023 al vencer por 1 a 0 a Platense en la Final.

Una vez desarrolladas las funciones para interactuar con nuestro modelo, procederemos al desarrollo de cuatro funciones cruciales. 3 de estas funciones estarán diseñadas para consultar información específica extraída de fuentes de datos particulares, dependiendo de la necesidad de la información. Y la 4ta función, es la que se encargará de decidir a qué función se debe invocar.

Las funciones que desarrollaremos serán las siguientes:

Los función "**query_desde_chroma**" realizará el siguiente procedimiento en caso de tener que obtener información desde la base de datos vectorial:

- 1 - Obtenemos la consulta
- 2 - Le realizamos embedding a la consulta
- 3 - Consultamos a nuestra base **ChromaDB** (nos traemos los dos mejores resultados)
- 4 - Convertimos los dos mejores resultados en un string
- 5 - Llamamos al modelo utilizando el string como contexto

Prueba de la función:

```
consulta = "¿Los accesorios de joyería están permitidos en jugadores y arbitros?"
query_desde_chroma(consulta)
```

'Respuesta: Los accesorios de joyería están prohibidos estrictamente para jugadores (collares, anillos, braza-letes, pendientes, bandas cuero goma) durante el partido. Los arbitros también no pueden llevar joyas y/o adornos personales, excepto relojes para cronometrar el partido. Se permite cubrir joyas con cinta adhesiva. Se recomienda mostrar tolerancia y permitir el uso de gafas deportivas seguras para jugadores, especialmente los jóvenes. Se deberá controlar si el vestido o la joya constituye peligro durante el partido y se deberá prohibir el uso si se considere peligroso o se utiliza de manera peligrosa. Se prohíbe el uso de sistemas electrónicos para jugadores y/o el cuerpo técnico, excepto con la aprobación de la asociación miembro o organizador de la competencia para los dispositivos de seguimiento de jugadores. La información de datos se permite para la recepción y el uso por parte de la área técnica, siempre que se considere que no presente peligro para los jugadores y se autorice por los árbitros.'

Los función "**query_desde_grafos**" realizará el siguiente procedimiento en caso de tener que obtener información desde la información que obtuvimos desde la base de grafos:

- 1 - Obtenemos la consulta
- 2 - Procesamos la consulta
- 3 - Obtenemos las entidades de nuestra consulta
- 4 - Hacemos una consulta a nuestra base de grafos filtrando las oraciones que contienen nuestras entidades
- 5 - Llamamos al modelo utilizando las oraciones

Prueba de la función:

```
consulta = "En la Copa de la liga 2023, Quién eliminó a Racing?"
query_desde_grafos(consulta)
```

'RosarioCentral eliminó a Racing en la Copa de la Liga 2023.'

La función **"query_desde_df"** realizará el siguiente procedimiento en caso de tener que obtener información desde la información que obtuvimos desde los datos tabulares:

- 1 - Obtenemos la consulta
- 2 - Procesamos la consulta
- 3 - Obtenemos las entidades de nuestra consulta
- 4 - Nos filtramos las oraciones en las que se encuentren dichas entidades.
- 5 - Llamamos al modelo utilizando las oraciones

Prueba de la función:

```
consulta = "En el Mundial FIFA 2022, ¿Cual fue el resultado de Argentina vs México?"
query_desde_df(consulta)

'Argentina 2-0 México en el Mundial FIFA 2022 disputado en Qatar el 26 de noviembre.'
```

La función **"modelo"** realizará el siguiente procedimiento:

- 1 - Clasifica el prompt de acuerdo a las siguientes etiquetas:

0 → Reglamento

1 → Copa de la Liga 2023

2 → Resultados Mundiales

2 - Llama a la función correspondiente de acuerdo a qué información se desea obtener. Además, indica de donde se extrajo la información.

```
consulta = "En la Copa de la liga 2023, Quién eliminó a RiverPlate?"
modelo(consulta)

'RosarioCentral eliminó a RiverPlate en la Copa de la Liga 2023.\n\nDatos extraídos de Base de datos de grafos'
```

Por último, se desarrolla un algoritmo para interactuar con un Chatbot en base a todas las funciones definidas anteriormente. El resultado es el siguiente:

Encendiendo Chatbot...

Hola! Quien anda por ahi?

Tu nombre: Santi

Hola Santi! Soy un chatbot experto en Futbol.

Me destaco por mi conocimiento en lo siguiente:

- Reglamento de la FIFA
- Informacion sobre la fase final de la Copa de la Liga 2023
- Resultados de todos los mundiales FIFA.

Nadie me ha puesto un nombre... te gustaria designarme uno?

1=Si

2=No

Respuesta:

1

Genial, que emocion! Jamas me habian puesto un nombre... Como me deseas llamar?

Nombre para chatbot: Soccer IA

Genial! Soccer IA suena estupendo!!

En que puedo ayudarte? Si quieres salir recordá escribir #Salir#.

Pregunta Santi: Segun el Reglamento, ¿Cual es el máximo de jugadores que debe tener un equipo para disputar un partido?

Soccer IA: Según el Reglamento, un equipo debe tener un máximo de 11 jugadores para disputar un partido. (Regla 18)

Datos extraídos de Base de datos vectorial chromaDB

Pregunta Santi: En la Copa de la Liga 2023, Quién eliminó a GodoyCruz?

Soccer IA: Platense eliminó a GodoyCruz en la Copa de la Liga 2023.

Datos extraídos de Base de datos de grafos

Pregunta Santi: En el Mundial FIFA 2010, Cual fue el resultado de Argentina vs Nigeria?

Soccer IA: Argentina 1-0 Nigeria en el Mundial FIFA 2010, disputado en South Africa, Johannesburg el 12 de junio de 2010.

Datos extraídos de Datos tabulares

Pregunta Santi: #Salir#

Adios, hasta la proxima Santi!.

EJERCICIO 2

¿A qué llamamos Estado del Arte?

El término "**estado del arte**" hace referencia a las aplicaciones de agentes inteligentes basados en LLM libres hace alusión a las investigaciones y avances más recientes en este ámbito.

Agentes inteligentes basados en LLM libres

Las aplicaciones de los agentes inteligentes basados en LLM libres están experimentando una evolución constante. Con el transcurso del tiempo, estos agentes se irán utilizando para resolver más y más problemas.

Estas aplicaciones se pueden categorizar en dos amplias áreas:

Aplicaciones de Machine Learning (Aprendizaje Automático)

Estos agentes se emplean en tareas de aprendizaje automático, como clasificación, regresión, entre otras. Han logrado avances significativos en tareas como:

- **Clasificación:** Estos agentes se emplean para mejorar la precisión en la clasificación de texto en diversos ámbitos, como noticias o artículos científicos.
- **Regresión:** Estos agentes se utilizan para mejorar la precisión en la predicción de valores continuos del texto, como el número de palabras, la puntuación o el sentimiento de un texto.

Aplicaciones de Procesamiento del Lenguaje Natural

Estos agentes están diseñados para procesar el lenguaje natural. Han logrado avances significativos en tareas como:

- **Traducción automática:** Mejorar la exactitud y la naturalidad en las traducciones entre distintos idiomas de manera automatizada.

- **Resumen de texto:** Producir resúmenes más precisos y compactos de documentos o textos extensos.
- **Generación de texto:** Crear una variedad de contenido creativo, como poemas, código, narrativas, canciones, correos electrónicos, cartas, entre otros, de manera automatizada.

Problemática

Coordinación y planificación de una fiesta de cumpleaños.

Se quiere realizar una fiesta de cumpleaños y la organización es esencial para que todo salga bien y a tiempo. La falta de coordinación y comunicación entre las partes que llevan adelante la fiesta puede afectar de manera crítica a la misma. Para abordar este problema, se piensa un sistema multiagente para llevar a cabo la organización de la fiesta.

Agentes involucrados

Este problema contará con 4 agentes involucrados y la función de cada uno será la siguiente:

1. Agente de Planificación:

Objetivo principal: Coordinar y llevar adelante la planificación de la fiesta.

Funciones:

- Se encarga de la gestión de la fecha y hora en la que va llevarse a cabo la fiesta.
- Contratación del salón para la fiesta.
- Supervisa que la fiesta y las tareas asignadas se cumplan correctamente.

2. Agente de Comunicación:

Objetivo principal: Mantener a los invitados informados sobre la fiesta.

Funciones:

- Se encargará de mantener al tanto a los invitados de toda la información que deban saber: lugar, fecha, hora.
- Informará a la Cocina si se requieren platos especiales (vegetarianos, celíacos)

3. Agente de Cocina:

Objetivo principal: Gestionar que todos los platos salgan en orden y a tiempo.

Funciones:

- Garantizar que la comida salga en excelente estado.
- Comunicar en caso de retrasos al agente de comunicación.

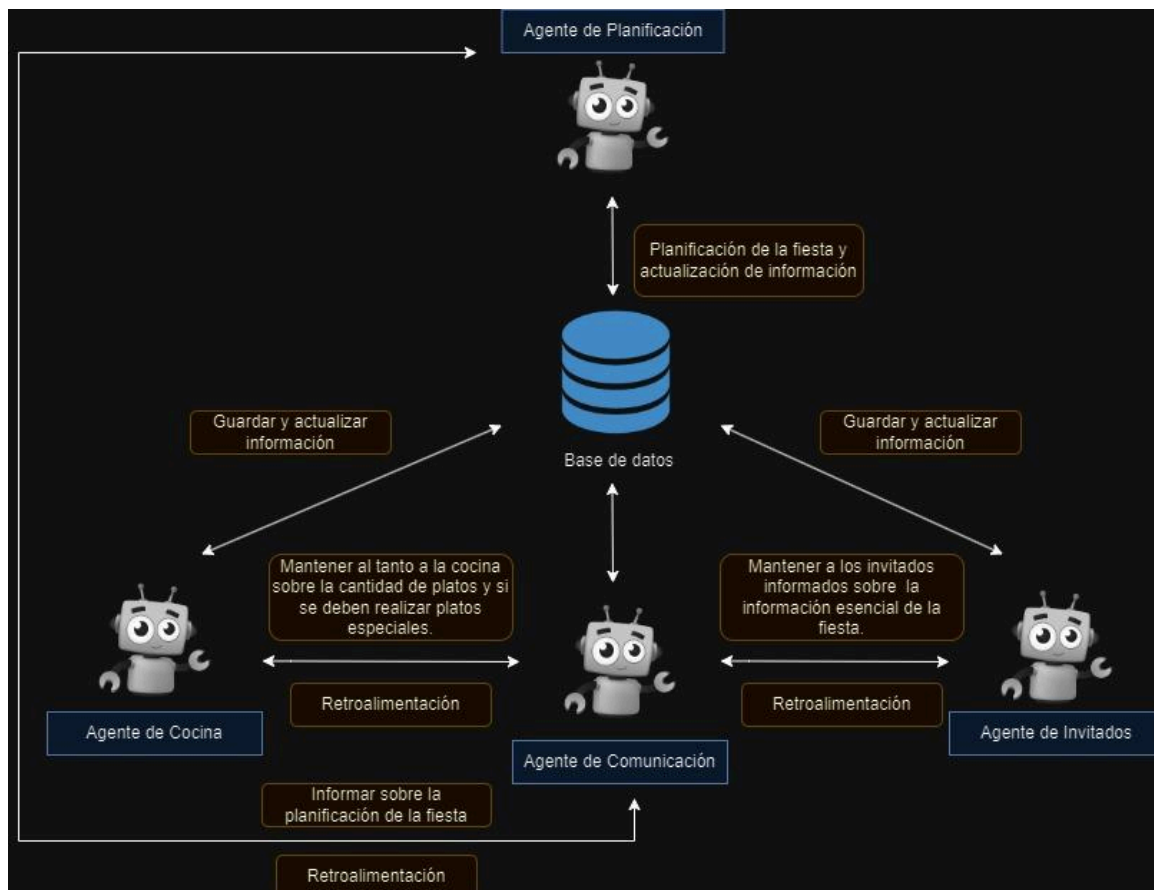
4. Agente de Invitados:

Objetivo principal: Representar a los invitados de la fiesta.

Funciones:

- Llegar a horario al lugar de la fiesta.
- Proporciona una retroalimentación del servicio.

Estructura funcionamiento de los agentes



- **"Agente de Planificación":** El lugar 'Alamos' está disponible para la fiesta el sábado próximo a las 8:00 PM. ¿Qué te parece?
- **"Agente de Comunicación":** ¡Perfecto! Enviaré un mensaje a los invitados para confirmar su disponibilidad ese día y a esa hora.

Interacción entre el **"Agente de Comunicación"** y el **"Agente de Invitados"**:

- **"Agente de Comunicación":** Hola 'Pedro', quería informarte que la fiesta de cumpleaños de 'Santiago' se llevará a cabo el sábado próximo a las 8:00 PM en el lugar 'Alamos'. ¿Podrías confirmar si podrás asistir?, en caso de asistir. ¿Necesitarás algún menú especial?,
- **"Agente de Invitados":** ¡Hola! Sí, estaré encantado de asistir. No necesito ningún menú especial. Gracias por la invitación.

Interacción entre el **"Agente de Cocina"** y el **"Agente de Comunicación"**:

- **"Agente de Comunicación":** Serán 100 invitados, por lo tanto necesitaremos 100 platos.
- **"Agente de Cocina":** Perfecto, ¿Hay algún invitado que necesite algún menú especial?
- **"Agente de Comunicación":** Si, necesitamos 10 platos vegetarianos

Pudimos ver cómo se lleva a cabo una conversación entre agentes, y las responsabilidades que cada uno tiene.

Bibliografía :

Unidad 7 de material brindado por la cátedra

<https://www.unite.ai/es/c%C3%B3mo-los-grandes-modelos-de-lenguaje-llm-impulsar%C3%A1n-las-aplicaciones-del-futuro/>

<https://www.b2chat.io/blog/b2chat/sistemas-multiagente-que-son-como-funcionan/>

https://www.youtube.com/watch?v=D6LtZ6QIZel&ab_channel=sololomejor

https://es.wikipedia.org/wiki/Sistema_multiagente

CONCLUSIONES FINALES

El presente trabajo práctico consistió en la resolución de dos ejercicios integradores que nos han llevado a aplicar varias técnicas de procesamiento de Lenguaje Natural. Por un lado, la implementación de un chatbot experto utilizando la técnica **RAG** (Ejercicio 1) ha sido un gran desafío. A través de este ejercicio, se ha logrado demostrar cómo integrar múltiples fuentes de conocimiento, incluyendo documentos de texto, datos numéricos en formato tabular, una base de datos de grafos y datos provenientes de una base vectorial, para proporcionar respuestas respecto a la información que se necesite obtener.

Durante el desarrollo de este proyecto, se ha explorado y aplicado una variedad de técnicas y herramientas de procesamiento del lenguaje natural, así como también habilidades de manejo de datos y consultas a bases de datos.

Finalmente, es importante destacar las amplias oportunidades que este proyecto ha abierto. La necesidad de investigación, pruebas y adaptaciones constantes ha permitido un profundo aprendizaje de los conceptos dados.

En relación al ejercicio 2, ha brindado una comprensión amplia del panorama actual de las aplicaciones de agentes inteligentes que hacen uso de modelos de LLM libres. Esta exploración ha sido sumamente enriquecedora, ya que constituye un cierre integral para la materia, donde convergen y se aplican los diversos conceptos y conocimientos adquiridos a lo largo de la misma. Este ejercicio no solo ha permitido comprender la relevancia y el potencial de los modelos de LLM en el desarrollo de multiagentes, sino que también ha proporcionado una visión práctica de cómo estos pueden ser implementados para abordar problemas y desafíos concretos en diferentes ámbitos.