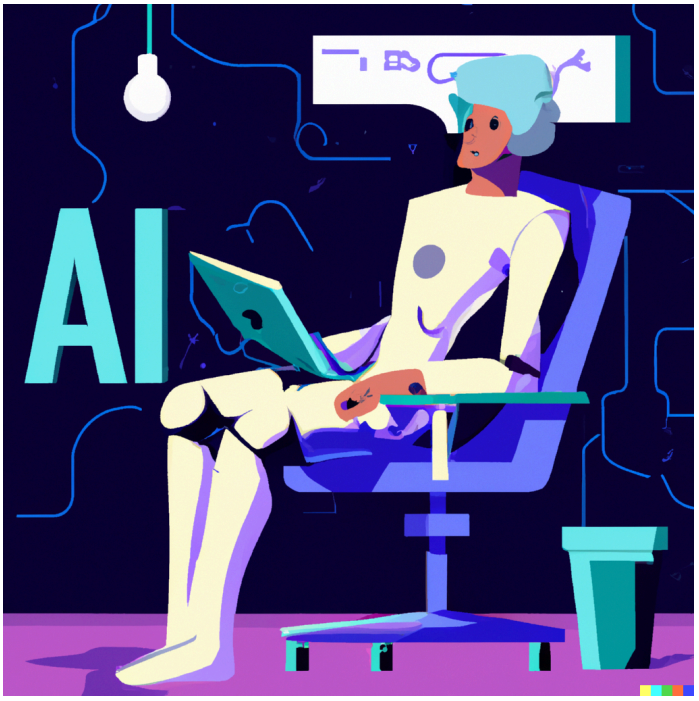
"Conversaciones Épicas: Explorando la Sabiduría de la Tierra Media a través de un Chatbot Tolkieniano"

El proyecto propuesto tiene como objetivo la creación de un chatbot experto en el fascinante mundo de "El Señor de los Anillos" y "El Hobbit" utilizando la técnica Retrieval Augmented Generation (RAG). Este sistema permitirá a los usuarios entablar conversaciones en español con el chatbot, formulando preguntas sobre diversos aspectos del universo literario creado por J.R.R. Tolkien.



Autor: Alejo Lo Menzo

FCEIA
FACULTAD DE
CIENCIAS EXACTAS,
INGENIERIA Y AGRIMENSURA

1. Descarga e Importación de Librerías:

Para la utilización del Chatbot debemos instalar diversas bibliotecas y herramientas mediante el comando !pip install. Entre las bibliotecas notables se encuentran kaggle, sentence-transformers, PyPDF2, langchain, python-decouple, PyMuPDF, gdown, chromadb, fpdf, SPARQLWrapper, wikidataintegrator, openpyxl, y pycryptodome. Estas bibliotecas proporcionan funcionalidades variadas necesarias para el proyecto.
Por último debemos importar estas Librerías y configurar algunos elementos del entorno, como la supresión de advertencias mediante warnings.filterwarnings("ignore").

```
!pip install langchain
!pip install python-decouple
!pip install PyMuPDF
!pip install gdown
!pip install chromadb
!pip install fpdf
!pip install SPARQLWrapper
!pip install wikidataintegrator
!pip install openpyxl
!pip install pycryptodome --force
```

```
from google.colab import files, drive
import zipfile
import os
from PyPDF2 import PdfReader
import re
import pandas as pd
import numpy as np
import requests
from langchain.text_splitter import RecursiveCharacterTextSplitter
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

2. Carga del Token de Hugging Face:

En este segmento de código, se configuran las variables de entorno necesarias para el proyecto. Se asigna el token de Hugging Face a la variable 'HUGGINGFACE_TOKEN' utilizando el método config de la biblioteca “Decouple”. Este token se guarda en un archivo .env para asegurar la confidencialidad. En caso de que no se proporcione un valor en el archivo .env, se utilizará un valor predeterminado. Este enfoque garantiza una gestión segura de información sensible y facilita la flexibilidad en la configuración del proyecto.

3. Carga de Fuentes de Datos:

3.1. Texto:

Para la fuente de texto elegí la trilogía del "Señor de los Anillos" y "El Hobbit". Los archivos se descargan de una carpeta de google drive y se encuentran en formato PDF.

Después implementamos una función para extraer el texto de los mismos y otra función para limpiar los saltos de línea ya que el splitter de texto de Langchain utiliza los saltos de línea como guía para dividir el texto.

3.2. Datos Numéricos Tabulares:

Para manejar datos numéricos tabulares, empleé un pequeño conjunto de datos que contiene información sobre el año de lanzamiento, la duración, la fecha de estreno, presupuesto y recaudación total de la Trilogía del Señor de los Anillos y El Hobbit. Este conjunto de datos es de mi propia creación y está en formato XLSX.
Los datos fueron cargados en un DataFrame de Pandas, y posteriormente se creó una función para imprimir el DataFrame completo en formato de texto con sentido semántico. Aunque hubiera sido posible proporcionar sólo la información solicitada en el prompt para optimizar el contexto, se optó por mostrar todo el texto cuando la consulta hacía referencia a las novelas físicas, priorizando así el tiempo de ejecución y considerando el tamaño reducido del dataset.

3.3. Base de Datos de Grafo:

Opté por Wikidata como la base de datos de grafos para realizar consultas en línea. Esta fuente se emplea exclusivamente cuando se formulan preguntas relacionadas con datos de personajes. En este proceso, se realiza una consulta para obtener el ID del personaje y, posteriormente, se obtienen las etiquetas de los objetos correspondientes a un conjunto de predicados previamente seleccionados.

```
# Uso de la función
label_personaje_hobbit_senior_anillos = "Thorin"  # Reemplaza con el personaje deseado
print(obtener_valores_por_personaje_tolkien(label_personaje_hobbit_senior_anillos))
```

```
Thorin :
Descripción: personaje de El hobbit, de J. R. R. Tolkien, un enano
Reino: Desconocido
Raza: masculino
Miembro de: Compañía de Thorin
Enemigo: Desconocido
Papel: espadachín
```

4. Base de datos Vectorial de Embeddings:

Una vez que ya tenemos los PDFs en el formato necesario vamos a realizar la base de datos vectorial con los embeddings.
Primero utilizamos RecursiveCharacterTextSplitter de langchain para dividir los textos en Chunks. Después mejoramos la calidad de los embeddings pasando todo el texto a minúsculas, eliminando los símbolos, las stop words y lematizando los chunks de texto.
Por último al tener los chunks limpios llamamos el modelo de embeddings paraphrase-multilingual-mpnet-base-v2 y almacenamos los embeddings 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 was Bilbo Bolsón?"
query_embedding = embed_model.encode(query_texts).tolist()

results = collection.query(query_embeddings= [query_embedding], n_results=3)
for result in results['documents'][0]:
    print(result)
    print('\n')
```

```
creía que bilbo hubiera muerto cuando le preguntaban dónde está entonces se encogía de hombros vivía solo

cómo lo descubrió preguntó frodo en cuanto al nombre se lo dijo bilbo mismo muy tontamente luego le fue d

que tenía la costumbre de desaparecer con una detonación un relámpago para reaparecer con una detonación
```

5. Implementación del RAG:

5.1. Traducción de Prompts y Respuestas:

Decidí traducir las preguntas de entrada al inglés para facilitar la interacción con el chatbot, dado que los modelos de lenguaje natural tienden a ser más avanzados en inglés debido a su entrenamiento. Utilicé los modelos "opus-mt-es-en" y "opus-mt-en-es" de Helsinki-NLP, disponibles en HuggingFace, para realizar la traducción al inglés. Luego, trabajé en este idioma y finalmente, volví a traducir al español la respuesta final.

5.2. Clasificación para Elegir la Fuente de Contexto:

Para la clasificación de preguntas y la selección de la fuente contextual, empleé un Modelo de Lenguaje de Aprendizaje Profundo (LLM) específico, el "zephyr-7b-beta", disponible en Hugging Face. Utilicé un formato de prompt Few-Shot para obtener respuestas más precisas en función del contexto proporcionado.

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

```
['Personaje', 'Frodo Bolsón, Boromir, Gandalf, etc.']
```

5.3. Interpretación de Respuesta:

Dado que solicité un formato de respuesta específico al LLM, ahora puedo utilizar una función para obtener la información necesaria. La información se formateará en una lista con dos elementos: el primero indicará el tema de la pregunta ("Personaje", "Libros", "Películas"), y el segundo será "None" a menos que se trate de una pregunta sobre un personaje. En ese caso, también se proporcionará el nombre del personaje para realizar la consulta en la base de datos de grafos.

5.4. Devolución de Contexto:

Habiendo identificado el tipo de prompt y, por ende, la fuente que debe proporcionar el contexto, se creó una función que recibe el binomio resultante de la función anterior. Esta función devuelve el texto predefinido del DataFrame de los libros, vectoriza y busca el prompt en la base de datos ChromaDB. Si el prompt hace referencia a un personaje, se realiza el proceso descrito anteriormente para obtener los datos correspondientes de ese personaje en la base de datos de grafos. Siguiendo con el ejemplo con el que se venía trabajando:

```
# Ejemplo
contexto = devolver_contexto(respuesta_limpia, prompt)[1]

print(contexto)
```

```
Base de datos de grafos
```

6. Preguntar al Agente con el contexto aclarado:

El último paso es preguntarle al LLM pero añadiendo a la pregunta el contexto y pidiéndole que no utilice su conocimiento previo.

7. Resultados:

```
# Prompt:
prompt = 'Personaje: Bilbo Bolsón quien fue y que hizo'

# Llamar a la función del RAG
respuesta, fuente, contexto = RAG(prompt, "TOKEN_HUGGINGFACE")

# Imprimir el resultado
print('--------------------------------------------------------------------------')
print(f'PREGUNTA:\n{prompt}')
print('--------------------------------------------------------------------------')
print(f'FUENTE ESCOGIDA: {fuente}')
print('--------------------------------------------------------------------------')
print(f'CONTEXTO BRINDADO:\n{contexto}')
print('--------------------------------------------------------------------------')
print(f'RESPUESTA:\n{respuesta}')
```

```
--------------------------------------------------------------------------
PREGUNTA:
Personaje: Bilbo Bolsón quien fue y que hizo
--------------------------------------------------------------------------
FUENTE ESCOGIDA: Base de datos vectorial de Embeddings
--------------------------------------------------------------------------
CONTEXTO BRINDADO:
cómo lo descubrió preguntó frodo en cuanto al nombre se lo dijo bilbo mismo muy tontamente luego le fue di
--------------------------------------------------------------------------
RESPUESTA:
Bilbo Bolsón fue un hobbit mencionado en la novela "El Señor de los Anillos" del escritor J.R.R. Tolkien.
```