Informe RAG:

Tecnicatura Universitaria en Inteligencia Artificial

Fecha: 27/02/2024

Integrante:

- Mateo Rovere

Profesor:

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

Se puede encontrar todo el repositorio en mi github:
https://github.com/Mateorovere/TP2_NLP

RAG (Retrieval-Augmented Generation):

Introducción:

RAG (**Generador Aumentado por Recuperación**) es un modelo de lenguaje de última generación que combina la potencia de la generación de lenguaje con la precisión de la recuperación de información.

¿Cómo funciona?

1. **Entrada:** Se le proporciona al modelo una consulta o un texto corto como entrada.
2. **Recuperación:** RAG busca en una gran base de datos de documentos relevantes para la consulta.
3. **Generación:** El modelo utiliza la información recuperada para generar un texto nuevo y original que responde a la consulta.

Ventajas de RAG:

- **Precisión:** La información recuperada por RAG ayuda a garantizar que el texto generado sea preciso y relevante para la consulta.
- **Creatividad:** RAG puede generar textos nuevos y originales que van más allá de la simple recuperación de información.
- **Versatilidad:** RAG se puede utilizar para una amplia gama de tareas, como la generación de resúmenes, la traducción automática, la escritura creativa y la respuesta a preguntas.

RAG es un modelo de lenguaje poderoso y versátil que tiene el potencial de revolucionar la forma en que interactuamos con la información.

Yo elegí implementar RAG que sea experto sobre la anatomía humana a partir de libros de fuentes confiables, a partir de un archivo csv (que contiene información de los sistemas del cuerpo) y de datos de wikidata.

Tuve unos problemas importando llama-cpp-python, así que tuve que correr esta linea de codigo:

```
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install
llama-cpp-python==0.1.78 numpy==1.23.4 --force-reinstall --upgrade
--no-cache-dir --verbose
```

Luego importe todas las librerías necesarias:

```
import os
import shutil
import getpass

from urllib.request import urlretrieve
from zipfile import ZipFile
from PyPDF2 import PdfReader

import csv
import chromadb
import pandas as pd

from huggingface_hub import hf_hub_download
from llama_cpp import Llama
from sentence_transformers import SentenceTransformer

import requests
import wikipedia
```

Teniendo en cuenta que el entorno de Colab tiene como límite 13 mil millones de parámetros para los modelos, elegí a la versión de 13b de LLAMA 2, dado que cumplía ese requisito y que tenía buen performance en el lenguaje natural.

```
model_name_or_path = "TheBloke/Llama-2-13B-chat-GGML"
model_basename = "llama-2-13b-chat.ggmlv3.q5_1.bin"

model_path = hf_hub_download(repo_id=model_name_or_path,
filename=model_basename)

modelo_llm = Llama(
    model_path=model_path,
    n_threads=2,
    n_batch=512,
    n_gpu_layers=128,
    n_ctx=2048)
```

El modelo de embedding que elegí es el “intfloat/multilingual-e5-base”, dado que la versión “large” del mismo me daba error por CUDA debido a un OOM por el colab.

```
modelo_emb = SentenceTransformer('intfloat/multilingual-e5-base')
```

Luego a las bases de datos vectoriales estaban con chromaDB

```
chroma_client = chromadb.Client()

collection = chroma_client.get_or_create_collection(name='Anatomia')
clasificador =
chroma_client.get_or_create_collection(name='Clasificador')
tabla = chroma_client.get_or_create_collection(name='tabla')
```

Para hacer split en los textos use RecursiveCharacterTextSplitter con un chunk_size de 500, lo hice de este modo porque intente con chunk_size mayores, pero tenia problemas por OOM, asi que fui reduciendo el valor y con 500 encontré que era el mayor valor que no me daba errores.

```
def split_text_into_parts(text):
    # No need for max_length condition, split the text as is
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=500,
separators=['\n\n','\n'])
    parts = text_splitter.split_text(text)
    return parts
```

Y por ultimo la respuesta de este codigo seria el RAG que incluye el clasificador de la base de datos que mejor se adapte a la pregunta de ahi elige si usar los libros, la tabla csv o Wikipedia:

```
while True:
    longitud_maxima = 100

    print('\n' + '-' * longitud_maxima + '\n')

    try:
        consulta = str(input("Ingrese su consulta ('q' para salir): "))
        print()
        if consulta.lower() == 'q':
```

```
            print()
            print('Saliendo...')
            break
    except Exception as e:
        print(f"Error al leer la entrada del usuario: {e}")
        continue

    resultado = "\n\nBuenos días/tardes Usuario(a), como consultor en
anatomia humana, puedo informarte que " + consulta_anato(consulta)

    lineas_resultado = resultado.split('\n')

    resultado_segmentado = [linea[i:i+longitud_maxima] for linea in
lineas_resultado for i in range(0, len(linea), longitud_maxima)]

    for linea in resultado_segmentado:
        print(linea)

    print()

    actualizar_clasificador(consulta)
```