Esquema de Desarrollo del Proyecto RAG

Introducción:

En este esquema, se detallan los pasos a seguir para el desarrollo de nuestro sistema RAG utilizando Llama 3, LangChain y Streamlit. El proyecto se desarrollará en GitHub y contará con la colaboración de ambos miembros del equipo.

Objetivo:

Implementar un sistema RAG funcional y atractivo para nuestra tesis, utilizando las herramientas mencionadas anteriormente. El sistema permitirá a los usuarios introducir queries en español, obtener respuestas relevantes y evaluar el rendimiento del sistema.

Metodología:

1. **Configuración del proyecto en GitHub:**
    - Crear un nuevo repositorio en GitHub con un nombre descriptivo.
    - Clonar el repositorio en ambos ordenadores locales.
    - Instalar las dependencias necesarias (Llama 3, LangChain, Streamlit, etc.) en ambos entornos.
2. **Vectorización de la base de datos:**
    - Utilizar Llama Data para cargar y preprocesar la base de datos de documentos en ambos entornos.
    - Utilizar Llama Vector para vectorizar los documentos preprocesados utilizando el modelo Llama 3 en ambos entornos.
    - Almacenar la base de datos vectorial en un formato compatible con Llama 3.
3. **Implementación del esquema con LangChain:**
    - Definir enlaces de LangChain para cada paso del esquema, incluyendo:
        - Preprocesamiento del query (eliminar caracteres especiales, convertir a minúsculas, tokenizar, stemming/lemmatization).
        - Traducción del query (español a inglés) utilizando un Deep LLM como Google Translate o DeepL Translate.
        - Búsqueda en la base de datos vectorial utilizando Llama 3.
        - Recuperación de los documentos más relevantes.
        - Generación de respuesta inicial en inglés utilizando Llama 3.
        - Evaluación de la respuesta inicial utilizando un RAG corrector (por ejemplo, RAG-T5).
        - Refinamiento de la respuesta (utilizar la respuesta inicial si es satisfactoria o generar una alternativa con el RAG corrector).
        - Traducción de la respuesta final (inglés a español) utilizando el mismo Deep LLM que en la traducción del query.
        - Postprocesamiento de la respuesta (corrección gramatical y ortográfica, formato).
        - Presentación de la respuesta en la interfaz web.
    - Conectar los enlaces de LangChain en un flujo de trabajo lógico para ejecutar el esquema.
4. **Desarrollo de la interfaz web con Streamlit:**
    - Crear una aplicación Streamlit en ambos entornos.

    - Diseñar una interfaz intuitiva y atractiva que permita a los usuarios:
        - Introducir queries en español.
        - Visualizar las respuestas generadas por el sistema.
        - Evaluar la precisión y fluidez de las respuestas.
        - Explorar el funcionamiento del sistema (opcional).
    - Integrar los enlaces de LangChain en la aplicación Streamlit para procesar los queries y mostrar las respuestas.
5. **Pruebas y ajustes:**
    - Probar exhaustivamente el sistema con una variedad de queries en español para asegurar su correcto funcionamiento.
    - Ajustar los parámetros de los diferentes componentes (Llama 3, RAG corrector, Deep LLM) para optimizar el rendimiento y la precisión.
    - Documentar el código de forma clara y concisa para facilitar su comprensión y uso.
    - Utilizar un sistema de control de versiones (Git) para gestionar los cambios, realizar un seguimiento del progreso y colaborar de forma efectiva.

Herramientas y tecnologías:

- **Llama 3:** Modelo de lenguaje para la búsqueda en la base de datos y la generación de respuestas.
- **LangChain:** Herramienta para orquestar los diferentes componentes del sistema.
- **Streamlit:** Framework para crear interfaces web interactivas.
- **Llama Data:** Herramienta para cargar y preprocesar la base de datos de documentos.
- **Llama Vector:** Herramienta para vectorizar los documentos utilizando el modelo Llama 3.
- **Deep LLM:** Google Translate o DeepL Translate para la traducción del query y la respuesta.
- **RAG corrector:** RAG-T5 o similar para evaluar y refinar las respuestas.
- **Git:** Sistema de control de versiones para la gestión del código.

Cronograma:

- **Semana 1:**
    - Configuración del proyecto en GitHub.
    - Vectorización de la base de datos.
    - Implementación del esquema con LangChain (pasos básicos).
- **Semana 2:**
    - Desarrollo de la interfaz web con Streamlit.
    - Implementación del esquema con LangChain (pasos restantes).
    - Integración de LangChain en Streamlit.
- **Semana 3:**
    - Pruebas y ajustes.
    - Documentación del código.
    - Refinamiento
- **Semana 4:**
    - Pruebas exhaustivas y ajustes finales
    - Preparación de la demostración.

Roles y responsabilidades:

- **[Mi nombre]:**
  - Enfoque en el desarrollo de los enlaces de LangChain para la generación y evaluación de respuestas.
  - Integración de LangChain en la aplicación Streamlit.
  - Optimización y pruebas del sistema.
- **[Nombre compañero]:**
  - Enfoque en la vectorización de la base de datos con Llama Data y Llama Vector.
  - Desarrollo de la interfaz web y el diseño con Streamlit.
  - Manejo del preprocesamiento y la traducción de queries/respuestas.
  - Pruebas y optimización del sistema.

Colaboración y comunicación:

- **Sincronizaciones frecuentes:** Acordar horarios regulares para sincronizar el trabajo, discutir avances y desafíos.
- **GitHub como eje central:** Utilizar GitHub activamente para subir código, crear issues y discutir soluciones a problemas.
- **Comunicación abierta:** Fomentar una comunicación clara y directa para abordar cualquier obstáculo de forma rápida y eficiente.

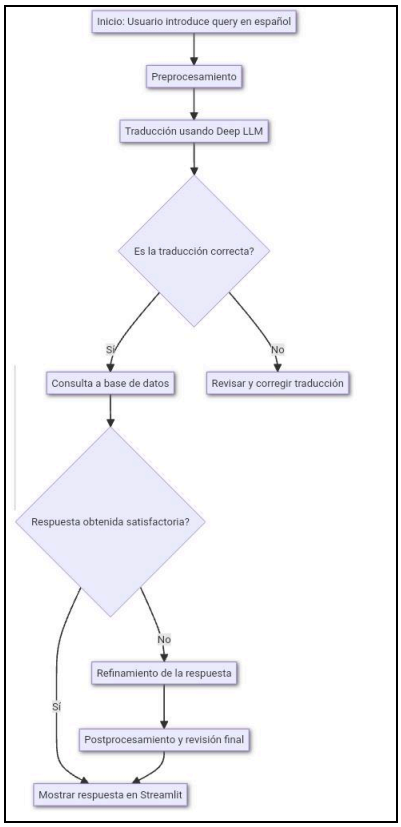
Evaluación del rendimiento:

- **Conjunto de datos de prueba:** Preparar un conjunto de datos de prueba con queries en español y respuestas esperadas de calidad.
- **Métricas de evaluación:** Seleccionar métricas relevantes para evaluar la precisión y fluidez de las respuestas. Ejemplos: BLEU, ROUGE, etc.
- **Evaluación comparativa:** Considerar comparar el rendimiento del sistema con y sin el RAG corrector.

Demostración para la tesis:

- **Preparación:** Pulir el funcionamiento del sistema y la interfaz de usuario para una demostración convincente.
- **Guión:** Redactar un guión de demostración que explique claramente el funcionamiento del sistema y sus componentes.
- **Práctica:** Ensayar la demostración con anticipación para estar preparados en la defensa de la tesis.

AQUÍ TENEMOS EL DIAGRAMA BÁSICO DE NUESTRO RAG



ANEXO DIAGRAMA RAG IMAGEN No XX

Se ha embebido el diagrama básico del RAG en un código html para poder visualizar sin ayuda de editores externos el código Mermaid utilizado para elaborarlo.

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Diagrama de Decisión - Sistema RAG</title>

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/mermaid/9.2.0/mermaid.min.css">

    <script
```

```
src="https://cdnjs.cloudflare.com/ajax/libs/mermaid/9.2.0/mermaid.min.js"></script>
    <script>mermaid.initialize({ startOnLoad: true });</script>
    <style>
        /* Estilos generales para cada tipo de nodo */
        .node rect {
            stroke-width: 3px;
            filter: drop-shadow(2px 2px 3px #666);
        }
        /* Colores y estilos específicos por tipo de nodo */
        .node.id1 rect { fill: #f0f0f0; stroke: #333; } /* Inicio */
        .node.id2 rect, .node.id3 rect, .node.id5 rect { fill: #e0ffff; stroke: #4d97d9; } /*
Procesos */
        .node.id4 rect, .node.id6 rect { fill: #ffffff; stroke: #666; } /* Decisiones */
        .node.id7 rect, .node.id9 rect { fill: #90ee90; stroke: #008000; } /* Final y outputs */
    </style>
</head>
<body>
    <div class="mermaid">
        graph TD
            A[Inicio: Usuario introduce query en español] --> B[Preprocesamiento];
```

```
        B --> C[Traducción usando Deep LLM];

        C --> D{Es la traducción correcta?};

        D -->|Sí| E[Consulta a base de datos];

        D -->|No| F[Revisar y corregir traducción];

        E --> G{Respuesta obtenida satisfactoria?};

        G -->|Sí| H[Mostrar respuesta en Streamlit];

        G -->|No| I[Refinamiento de la respuesta];

        I --> J[Postprocesamiento y revisión final];

        J --> H;

    </div>

</body>

</html>
```