LLM Categorizer Demo

Este proyecto es una **demo funcional** de un sistema inteligente que procesa preguntas de usuarios, **las categoriza automáticamente**, **busca documentos relevantes** y **genera respuestas** usando un modelo de lenguaje como GPT-4.

Fue desarrollado como **proyecto personal** y sirve como referencia para soluciones conversacionales en sectores como construcción, educación, soporte técnico, entre otros.

¿Qué hace esta aplicación?

1. Carga documentos en texto o PDF.
2. Fragmenta y vectoriza el contenido usando **FAISS** y **OpenAI embeddings**.
3. Acepta preguntas de usuarios.
4. Recupera los fragmentos más relevantes con búsqueda semántica.
5. Construye un prompt con contexto y lo envía al **LLM (GPT-4)**.
6. Devuelve una respuesta clara y contextualizada al usuario.

Tecnologías utilizadas

- Python
- Streamlit
- LangChain
- FAISS
- OpenAI API
- dotenv

Estructura del proyecto

```
llm-categorizer-demo/
├── app/
│   ├── main.py                # Interfaz Streamlit
│   ├── config.py              # Clave API
│   ├── generate_embeddings.py # Carga y vectorización de documentos
│   ├── search_documents.py    # Búsqueda semántica con FAISS
│   └── generate_response.py   # Generación de respuesta con GPT
├── data/
│   └── documentos_raw/        # Tus archivos .txt o .pdf
├── vectorstore/               # Índice FAISS generado
├── .env                       # Clave API privada
├── requirements.txt
└── README.md
```

⚙️ Instalación y ejecución

1. Cloná el repositorio

```
git clone https://github.com/tu-usuario/llm-categorizer-demo.git
cd llm-categorizer-demo
```

2. Instalá dependencias

```
pip install -r requirements.txt
```

3. Configurá la clave de OpenAI

Crea un archivo `.env` en la raíz con:

```
OPENAI_API_KEY=sk-tu-clave-aqui
```

4. Colocá documentos en la carpeta

```
/data/documentos_raw/
```

Podés usar `.txt` o `.pdf`.

5. Generá los embeddings

```
python app/generate_embeddings.py
```

6. Ejecutá la aplicación

```
streamlit run app/main.py
```

🧪 Ejemplo de uso

1. Subí un archivo `.txt` con contenido técnico.
2. Hacé una consulta desde la interfaz.
3. El sistema buscará los fragmentos más relevantes.
4. El modelo generará una respuesta basada en los documentos.

🧠 Casos de uso

- Chatbots internos basados en documentación de empresa
- Asistentes técnicos entrenados con manuales
- Sistemas de ayuda en sectores como:
  - Construcción
  - Educación
  - Soporte de software

📌 Créditos

Desarrollado por Jerónimo Martínez, Data Scientist especializado en soluciones aplicadas con IA.

🌐 DEMO en Vivo

Probá la aplicación funcionando en:

👉 Streamlit Cloud