🧱 LLM Categorizer Demo (Gemini + Hugging Face)

Este proyecto es una **demo funcional** de un sistema inteligente que procesa preguntas de usuarios, **busca documentos relevantes** y **genera respuestas** usando un modelo de lenguaje (Gemini Pro) y búsqueda semántica con embeddings de Hugging Face.

🚀 ¿Qué hace esta aplicación?

1. Carga documentos en texto o PDF.
2. Fragmenta y vectoriza el contenido usando **FAISS** y **Hugging Face 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 **modelo Gemini**.
6. Devuelve una respuesta clara y contextualizada al usuario.

🧰 Tecnologías utilizadas

- Python
- Streamlit
- LangChain
- FAISS
- Hugging Face Transformers
- Google Generative AI (Gemini)
- 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 Gemini
├── data/
│   └── documentos_raw/        # Tus archivos .txt o .pdf
├── vectorstore/               # Índice FAISS generado
├── .env.example               # Archivo de ejemplo para configuración
├── 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 Gemini

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

```
GEMINI_API_KEY=tu_clave_de_gemini
```

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 Gemini 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.

📄 Licencia

Este proyecto se publica bajo la licencia MIT.