📚 LLM Text Improver App

✅ Overview

This application allows users to enhance and correct texts written in Spanish using a locally running Large Language Model (LLM): **Mistral-7B-Instruct-v0.1** in **GGUF format**. It features a lightweight and accessible **graphical interface built with Tkinter**, and **does not require any internet connection or external APIs**.

The model processes user input using instruction-based prompts and returns a rewritten version that is more fluent, grammatically correct, and stylistically improved, while maintaining the original meaning.

📥 Model Download

To use this application, download the Mistral-7B-Instruct model (GGUF format) from the following link:

🔗 mistral-7b-instruct-v0.1.Q4_0.gguf

🧰 Tools Used

Tool	Purpose
`llama-cpp-python`	Loads `.gguf` models locally and efficiently on CPU and/or GPU.
`Tkinter`	Provides a simple, cross-platform GUI without external dependencies.
`Mistral-7B-Instruct`	Open-source LLM optimized for following natural language instructions.

🧾 Code Explanation

This section explains how the code works and how the model is integrated into the graphical interface.

🔍 1. Model Initialization

```
llm = Llama(
    model_path="models/mistral-7b-instruct-v0.1.Q4_0.gguf",
    n_ctx=2048,
    n_threads=6,
    n_gpu_layers=30
)
```

- Loads the **Mistral-7B-Instruct** model from a local `.gguf` file using `llama-cpp-python`.
- `n_ctx` defines the context length (max tokens the model can handle).
- `n_threads` sets how many CPU threads to use for inference.
- `n_gpu_layers` specifies how many layers to run on GPU (if available) to speed up execution.

💬 2. Text Enhancement Function

```
def mejorar_texto(prompt_usuario):
    prompt = f"[INST] Mejora y corrige el siguiente texto en español, manteniendo el sentido original: {prompt_usuario} [/INST]"
    respuesta = llm(prompt, max_tokens=512, temperature=0.7, stop=["</s>"])
    return respuesta["choices"][0]["text"].strip()
```

- Constructs an **instructional prompt** to guide the model in correcting and improving the input text.
- Calls the model with:
  - `max_tokens`: maximum number of tokens to generate.
  - `temperature`: creativity level (0.7 is balanced).
  - `stop`: token(s) that signal where to stop generation.
- Returns the model's response as clean text.

🧵 3. Background Processing with Threads

```
def procesar_texto():
    ...
    threading.Thread(target=tarea).start()
```

- Retrieves user input from the GUI.
- Disables the button and displays a “Processing...” message.
- Runs the model interaction in a **separate thread** to avoid freezing the GUI.
- Once the model finishes, it updates the output box with the result or an error message.

🖼️ 4. Graphical User Interface (GUI)

```
ventana = tk.Tk()
...
ventana.mainloop()
```

- The GUI is built with **Tkinter**, a built-in Python library for graphical interfaces.
- Components:
  - A `Text` box for the user to input the original text.
  - A `Button` to trigger the improvement function.
  - Another `Text` box to display the improved result.
- The layout uses `pack()` for simplicity and responsiveness.

This structure keeps the application lightweight, offline, and user-friendly — allowing efficient use of a powerful LLM locally without APIs or command-line tools.

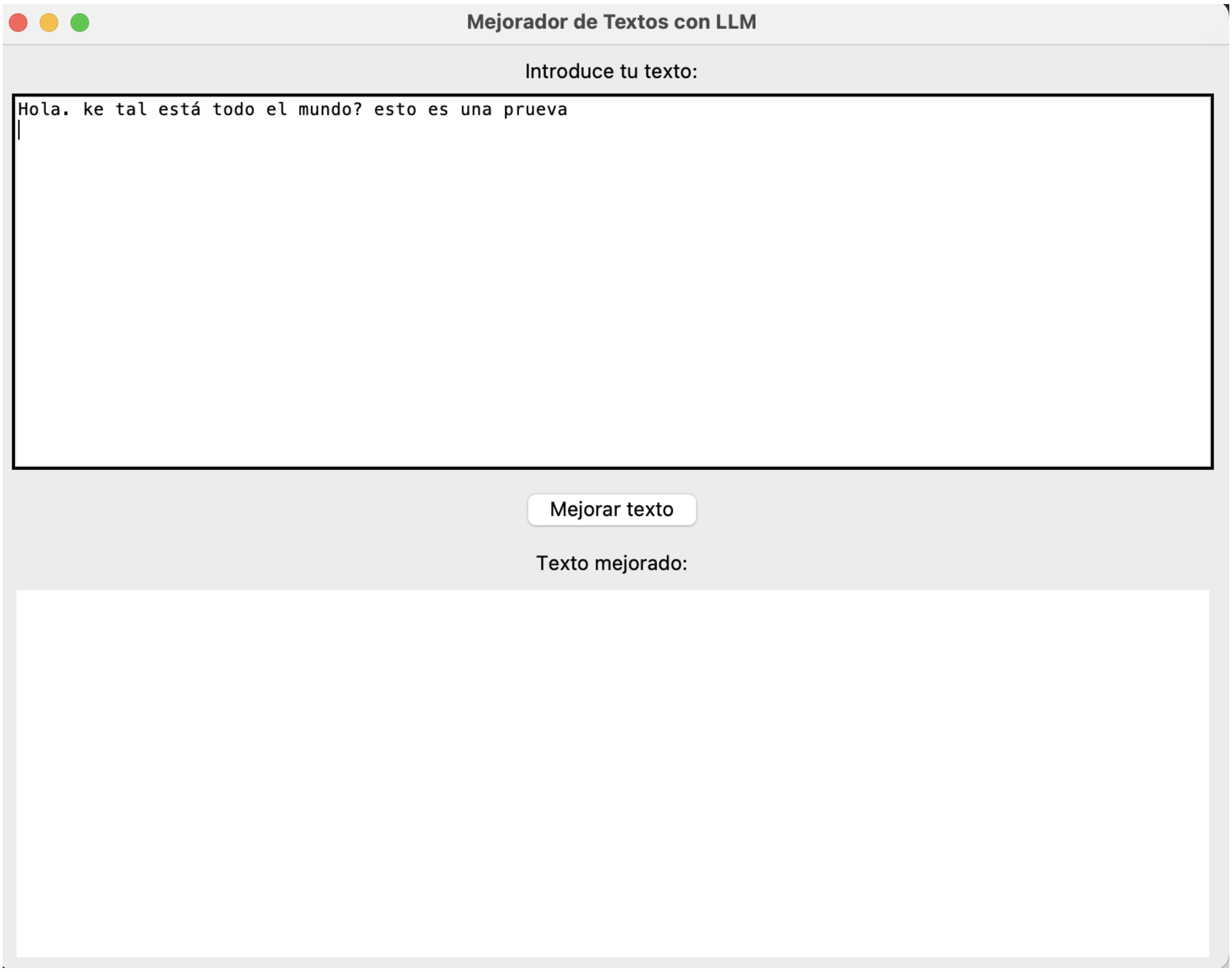
🧠 How It Works

1. The user writes or pastes a text they want to improve in the top input box.
2. By clicking the **"Improve Text"** button, the application sends the input to the local LLM using an instruction format like:
3. The model generates a corrected and improved version of the text, which is then displayed in the output box below.

💻 Interface Demo

🖊️ Step 1: Enter your text

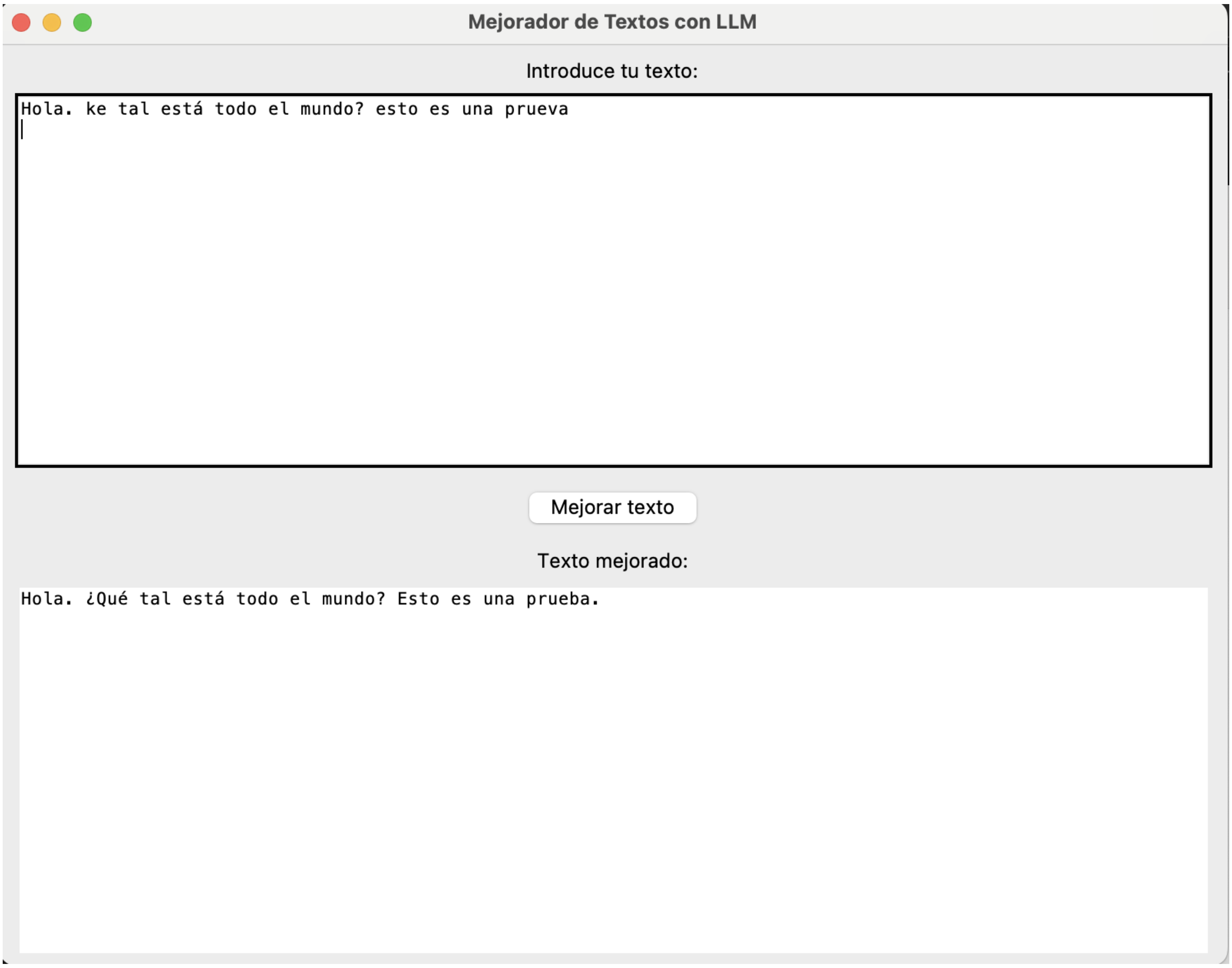
The user types or pastes the original text into the top input field.



```
Hola. ke tal está todo el mundo? esto es una prueva
```

🔁 Step 2: Click "Improve Text"

Once the button is pressed, the model processes the input and returns a cleaner and more correct version.



```
Hola. ke tal está todo el mundo? esto es una prueva
```

```
Hola. ¿Qué tal está todo el mundo? Esto es una prueba.
```

✅ Advantages

- 💻 **Runs fully offline** — no data is sent to external servers.
- 🔒 **Privacy-first** — everything happens locally.
- ⚡ **Fast and lightweight** — optimized for performance with `llama-cpp-python`.
- 🧠 **Powerful text correction** using one of the best open-source instruction-tuned LLMs.

📁 Project Structure

```
llm-text-improver/
├── models/
│   └── mistral-7b-instruct-v0.1.Q4_K_M.gguf
├── main.py
├── screenshots/
│   ├── image1.png       # Initial input interface
│   └── image2.png       # Output after text improvement
└── README.md
```