

AI-Powered Email Reply with Tone

AI-Powered Email Reply Generator - Detailed Explanation

The **AI-Powered Email Reply Generator** is a tool designed to **automate professional email responses** using **AI models like DeepSeek** via **Ollama**. This system allows users to **select a tone** for their reply (e.g., formal, friendly, apologetic, assertive), **paste an email**, and **generate a response** that follows a structured format.

This ensures **consistent, professional, and efficient email communication** without requiring users to craft replies manually.

File Structure & Detailed Breakdown

This project consists of four key Python files:

1 `main.py` - The User Interface (Gradio)

This file sets up the **Gradio interface**, allowing users to:

- **Paste an email** that needs a response.
- **Select a tone** (Formal, Friendly, Apologetic, Assertive).
- **Choose an AI model** from **available Ollama models**.
- **Click a button** to **generate a structured reply**.

Key Functionalities:

- `get_available_models()` → Retrieves available AI models installed in Ollama.
- `create_reply(tone, original_email, model)` → Calls the AI processing function from `email_reply_generator.py` to generate the reply.

How It Works:

- When the **Generate Reply** button is clicked, the function `create_reply()` calls the AI to generate a response.
- The AI **follows a predefined format**, including a **greeting, structured content, and a closing**.

◆ Example UI Code:

```
generate_button.click(
    fn=create_reply,
    inputs=[tone_selector, original_email_input, model_selector],
    outputs=reply_output
)
```

2 `email_reply_generator.py` - AI Email Processing

This file contains the **core AI logic** responsible for **constructing email responses**.

Key Functionalities:

- `run_ollama_prompt(prompt, model)` → Sends a **structured prompt** to the AI and retrieves the response.
- `generate_reply_email(tone, original_email, model, your_name)` → Constructs a **full email reply** with the chosen tone.

How It Works:

1. **Retrieves a tone-specific template** from `templates.py`.
2. **Formats the AI prompt**, ensuring a structured email with:
 - **Greeting** (e.g., "Dear [Sender],").
 - **AI-Generated Content**.
 - **Closing Signature**.
3. **Calls the AI model via Ollama** to generate the full response.

◆ Example Email Processing Code:

```
prompt = (
    f"{prompt_template}\n\n"
    f"Original Email:\n{original_email}\n\n"
    "Your reply should address the email above. Please include:\n"
    f" - A greeting: {greeting}\n"
    f" - A closing: {closing}\n\n"
    "Compose the complete reply email below:\n"
)
```

3 `templates.py` - Email Tone Templates

This file contains **predefined templates** for **different email tones**, ensuring that AI-generated replies match the user's **preferred communication style**.

Key Functionalities:

- **Stores tone-based prompt templates**, guiding the AI to generate **consistent responses**.

How It Works:

- If the user selects "**Formal**", the AI receives this instruction:

"Compose a formal reply email that is clear, professional, and courteous."

- If the user selects "**Friendly**", the AI is guided differently:

"Compose a friendly reply email with a conversational and approachable tone."

◆ **Example Template Code:**

```
TEMPLATES = {  
    "formal": "Compose a formal reply email that is clear, professional, and courteous.",  
    "friendly": "Compose a friendly reply email with a conversational and approachable tone.",  
    "apologetic": "Compose an apologetic reply email that expresses regret and acknowledges concerns.",  
    "assertive": "Compose an assertive reply email with a confident and clear tone."  
}
```

4 `config.py` - Formatting Rules

This file **stores email formatting rules**, ensuring that replies maintain a **structured format**.

Key Functionalities:

- Stores the **greeting and closing structure** for emails.
- Defines the **default sender name** (can be changed dynamically).

How It Works:

- Every AI-generated email **automatically** includes:

```
EMAIL_REPLY_FORMAT = {  
    "greeting": "Dear {sender},",  
    "closing": "Sincerely,\n{your_name}"  
}
```

- This ensures that **all emails include a greeting and signature**, making them **structured and professional**.

How Everything Works Together

- 1 User pastes an email into the Gradio UI (`main.py`).
- 2 User selects a tone & AI model.
- 3 The system retrieves the tone template (`templates.py`).
- 4 A structured email is constructed (`email_reply_generator.py`).
- 5 The AI generates a formatted reply and displays it in the UI.

How to Run the Project

Run the application:

```
python main.py
```

- 3 **Paste an email, choose a tone, and generate a reply!**