对话机器人gradio

gradio 介绍

Gradio 是一个简单易用的 Python 库，能够帮助开发者快速搭建用户友好的 Web 应用，特别适合用于机器学习模型的展示。本课程将使用 Gradio 来搭建一个可以与 FastAPI 后端交互的对话机器人。

Gradio 组件讲解

首先介绍 Gradio 的核心组件，包括：

- Gradio Blocks：用于组织界面布局的容器。
- Slider：用于调整生成参数，如 temperature 和 top_p。
- Textbox：用户输入对话的地方。
- Button：发送用户输入或清空历史记录。
- Chatbot：用于显示对话历史的组件。

代码分步讲解

安装gradio

```
pip install gradio==5.0.2
```

导入依赖库

```
import gradio as gr
import requests
```

定义后端 API 的 URL

这个 URL 是指向后端 FastAPI 的聊天接口。

```
backend_url = "http://127.0.0.1:6606/chat"
```

定义与后端交互的函数

- prompt：用户的输入。
- sys_prompt：系统提示语（可用于引导模型行为）。
- history：保存对话历史记录的变量。
- temperature：用于控制生成的多样性，数值越高，生成的文本越随机。
- top_p：用于控制采样的多样性。
- max_tokens：最大生成文本的长度。
- stream：是否启用流式输出。

```
def chat_with_backend(prompt, sys_prompt, history, history_len,
temperature, top_p, max_tokens, stream):

    # 构建请求数据
    data = {
        "query": prompt,
        "sys_prompt": sys_prompt,
        "history_len": history_len,
        "history": history,
        "temperature": temperature,
        "top_p": top_p,
        "max_tokens": max_tokens,
    }

    # 发送请求到 FastAPI 后端
    try:
        response = requests.post(backend_url, json=data,
stream=True)
        if response.status_code == 200:
            chunks = ""

            if stream:
                for chunk in
response.iter_content(chunk_size=None, decode_unicode=True):
                    if chunk:
                        chunks += chunk
                        chat_history_display = [(entry["role"],
entry["content"]) for entry in history]
                        chat_history_display.append(("user",
prompt))
```

```
                        chat_history_display.append(("assistant",
chunks))
                        # # 体验流式输出
                        # sleep(0.1)
                        yield chat_history_display,
gr.update(value='')
            else:
                for chunk in
response.iter_content(chunk_size=None, decode_unicode=True):
                    chunks += chunk

                chat_history_display = [(entry["role"],
entry["content"]) for entry in history]
                chat_history_display.append(("user", prompt))
                chat_history_display.append(("assistant", chunks))
                yield chat_history_display, gr.update(value='')

            history.append({"role": "user", "content": prompt})
            history.append({"role": "assistant", "content":
chunks})

        else:
            return "请求失败，请检查后台服务器是否正常运行。"
    except Exception as e:
        return f"发生错误：{e}"
```

清空对话历史

此函数用于清空当前的对话历史记录。

```
def clear_history(history):
    history.clear()
    return "", ""
```

使用 Gradio 搭建前端界面

gr.Blocks()表示页面的容器，gr.Row()表示一行，两个gr.Row()表示分成了两行，在第二行
使用gr.Column分成了两列，scale用来控制两列的占比。第二列中又被分为两行，第一行显示聊天记录（聊天窗）

，第二行用来显示输入框等组件。clear_button和submit_button分别是两个按钮组件，用于清空历史和发送。
点击clear_button将运行clear_history函数，输入参数是history，输出是chatbot, prompt（清空聊天窗和输入框），
点击submit_button将运行chat_with_backend函数，输入参数是prompt, sys_prompt, history, history_len, temperature, top_p, max_tokens, stream，输出是chatbot, prompt（更新聊天窗，清空输入框）。
history = gr.State([])记录对话状态。

```
with gr.Blocks() as demo:

    with gr.Row():
        gr.Markdown("## 🤖 聊天机器人")

    with gr.Row():
        with gr.Column(scale=1) as sidebar_left:
            history_len = gr.Slider(minimum=-1, maximum=10,
value=-1, label="保留历史消息的数量")
            temperature = gr.Slider(minimum=0.01, maximum=2.0,
value=0.7, step=0.01, label="temperature")
            top_p = gr.Slider(minimum=0.01, maximum=1.0,
value=0.9, step=0.01, label="top_p")
            max_tokens = gr.Slider(minimum=64, maximum=4096,
value=512, step=8, label="max_length")
            stream = gr.Checkbox(label="stream", value=True)

        with gr.Column(scale=5) as main:
            with gr.Row():
                chatbot = gr.Chatbot(label="Chat History")

            with gr.Row():
                prompt = gr.Textbox(label="", placeholder="来和我聊
天~")
                sys_prompt = gr.Textbox(label="系统提示语",
value="You are a helpful assistant.")

                submit_button = gr.Button("发送")
                clear_button = gr.Button("清空历史记录")

                history = gr.State([])
```

```
                clear_button.click(clear_history, [history],
[chatbot, prompt])

                submit_button.click(chat_with_backend,
                                    [prompt, sys_prompt, history,
history_len, temperature, top_p, max_tokens, stream],
                                    [chatbot, prompt])

demo.launch()
```

启动聊天

启动vllm

```
python -m vllm.entrypoints.openai.api_server --model
/root/hqyj_ai/models/models/Qwen/Qwen2___5-0___5B-Instruct --
served-model-name Qwen2___5-0___5B-Instruct &
```

启动fastapi

```
python chatbot_fastapi.py
```

启动webui

```
python chatbot_gradio.py
```