对话机器人streamlit

streamlit 介绍

Streamlit 是一个非常方便的 Python 库，用来快速构建数据驱动的 Web 应用。在这个项目中，Streamlit 将用于展示聊天界面并与后端进行交互。

API链接：https://docs.streamlit.io/develop/api-reference。

安装streamlit：

```
pip install streamlit==1.39.0
```

streamlit布局

导入依赖

```
import streamlit as st
import requests
```

页面设置

首先，使用 st.set_page_config 来配置页面的标题和图标。

```
st.set_page_config(page_title="Chatbot", page_icon="🤖",
layout="centered")
st.title("🤖 聊天机器人")
```

侧边栏配置

`st.sidebar` 是 Streamlit 库中的一个功能，用于在应用程序的侧边栏中添加内容。侧边栏是一个垂直排列的区域，通常位于页面的左侧，用于放置导航链接、控件（如滑块、按钮、选择框等）和其他用户界面元素。

在侧边栏中，我们允许用户配置一些模型的超参数，如 temperature（温度）和 max_tokens（最大输出长度）。这些参数通常用于控制模型生成文本的多样性和长度。

```
with st.sidebar:
    st.title('Chatbot')
    sys_prompt = st.text_input("系统提示语（可选）：", value="You are
a helpful assistant.")
    history_len = st.slider("保留历史消息的数量", min_value=-1,
max_value=10, value=-1)
    temperature = st.slider('temperature', min_value=0.01,
max_value=5.0, value=0.1, step=0.01)
    top_p = st.slider('top_p', min_value=0.01, max_value=1.0,
value=0.9, step=0.01)
    max_tokens = st.slider('max_length', min_value=64,
max_value=4096, value=512, step=8)
    stream = st.checkbox("stream", value=True)
```

聊天历史记录管理

聊天记录会保存在 st.session_state 中。

```
if 'history' not in st.session_state:
    st.session_state.history = []
```

历史记录会显示在页面的主区域

```
for message in st.session_state.history:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])
```

清空聊天记录

在侧边栏中提供一个按钮，用户可以随时清空聊天记录。

```
def clear_chat_history():
    st.session_state.history = []

st.sidebar.button('Clear Chat History',
on_click=clear_chat_history)
```

接受用户输入并显示消息

用户可以在聊天输入框中输入消息。每次用户发送消息后，都会将消息显示到页面上并将其保存在历史记录中。

```
if prompt := st.chat_input("来和我聊天~"):
    with st.chat_message("user"):
        st.markdown(prompt)
```

与 FastAPI 后端通信

后端 API 使用 FastAPI 搭建，用于处理用户的输入并返回模型生成的回复。我们通过 requests.post 向后端发送请求，并获取返回的流式响应。

发送请求

当用户发送消息后，构建请求数据并向后端发送 POST 请求。通过 response.iter_content() 来逐渐接收后端传输的数据。每次接收到一个新的数据块时，就会更新聊天界面上助手的消息。

```
# 发送请求到 FastAPI 后端
try:
    response = requests.post(backend_url, json=data, stream=True)
    if response.status_code == 200:
        chunks = ""
        # 用于流式更新的占位符
        assistant_placeholder = st.chat_message("assistant")
        assistant_text = assistant_placeholder.markdown("")

        if stream:
            for chunk in response.iter_content(chunk_size=None,
decode_unicode=True):
                # 处理响应内容
                chunks += chunk
                # 实时更新助手消息
                assistant_text.markdown(chunks)
                # 体验流式输出
                # sleep(0.2)
        else:
            for chunk in
response.iter_content(decode_unicode=True):
```

```
                # 处理响应内容
                chunks += chunk
                # 实时更新助手消息

            assistant_text.markdown(chunks)

        # 将用户的输入加入历史记录
        st.session_state.history.append({"role": "user",
"content": prompt})
        # 将助手的回复加入历史记录
        st.session_state.history.append({"role": "assistant",
"content": chunks})

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

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

```
streamlit run chatbot_streamlit.py
```