# 抢渡长江的数学建模分析

李俊霖

2025 年 3 月 14 日

## 1 任务

我将用 python 构建一个基于 llama 2 的聊天机器人，然后用 Api 对于它进行调用

## 2 结构

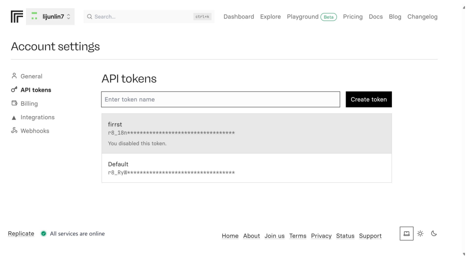
用 Streamlit 作为前端，用 Python 构建一个 Llama 2 聊天机器人，而 LLM 后端则通过对 Replicate 上托管的 Llama 2 模型的 API 调用来处理

## 3 获取 Replicate API 令牌

(1) 转到 Replicate https://llama2.streamlit.app

(2) 使用您的 GitHub 帐户登录。

(3) 转到 API 令牌页面并复制您的 API 令牌。



Account settings

API tokens

firrst
r8_18n**********************************
You disabled this token.

Default
r8_RyW**********************************

Replicate All services are online

新建一个 Api-Token，这个 Api-Token 就是之后你进行远程调用时所需要的

## 4 设置编码环境

本地开发

要设置本地编码环境，请在命令行提示符中输入以下命令：

```
pip install streamlit replicate
```

## 5 构建应用程序

总代码

```python
import streamlit as st
import replicate
import os

# App title
st.set_page_config(page_title=" ␣Llama␣2␣Chatbot")

# Replicate Credentials
with st.sidebar:
    st.title(' ␣Llama␣2␣Chatbot')
    if 'REPLICATE_API_TOKEN' in st.secrets:
        st.success('API␣key␣already␣provided!', icon=' ')
        replicate_api = st.secrets['REPLICATE_API_TOKEN']
    else:
        replicate_api = st.text_input('Enter␣Replicate␣API␣token:', type='password')
        if not (replicate_api.startswith('r8_') and len(replicate_api)==40):
            st.warning('Please␣enter␣your␣credentials!', icon='')
        else:
            st.success('Proceed␣to␣entering␣your␣prompt␣message!', icon=' ')
    os.environ['REPLICATE_API_TOKEN'] = replicate_api

    st.subheader('Models and parameters')
    selected_model = st.sidebar.selectbox('Choose a Llama2 model', ['Llama2-7B', 'Llama2-13B'], key='selected_model')
    if selected_model == 'Llama2-7B':
        llm = 'a16z-infra/llama7b-v2-chat:4f0a4744c7295c024a1de15e1a63c880d3da035fa1f49bfd344fe076074c8eea'
    elif selected_model == 'Llama2-13B':
        llm = 'a16z-infra/llama13b-v2-chat:df7690f1994d94e96ad9d568eac121aecf50684a0b0963b25a41cc40061269e5'
    temperature = st.sidebar.slider('temperature', min_value=0.01, max_value=5.0, value=0.1, step=0.01)
    top_p = st.sidebar.slider('top_p', min_value=0.01, max_value=1.0, value=0.9, step=0.01)
    max_length = st.sidebar.slider('max_length', min_value=32, max_value=128, value=120, step=8)
    st.markdown(' Learn how to build this app in this [blog](https://blog.streamlit.io/how-to-build-a-llama-2-chatbot/)!')

# Store LLM generated responses
if "messages" not in st.session_state.keys():
    st.session_state.messages = [{"role": "assistant", "content": "How may I assist you today?"}]

# Display or clear chat messages
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.write(message["content"])
```

```python

def clear_chat_history():
    st.session_state.messages = [{"role": "assistant", "content": "How may I assist you today?"}]
st.sidebar.button('Clear Chat History', on_click=clear_chat_history)

# Function for generating LLaMA2 response. Refactored from https://github.com/a16z-infra/llama2-chatbot
def generate_llama2_response(prompt_input):
    string_dialogue = "You are a helpful assistant. You do not respond as 'User' or pretend to be 'User'. You only respond once as 'Assistant'."
    for dict_message in st.session_state.messages:
        if dict_message["role"] == "user":
            string_dialogue += "User: " + dict_message["content"] + "\n\n"
        else:
            string_dialogue += "Assistant: " + dict_message["content"] + "\n\n"
    output = replicate.run('a16z-infra/llama13b-v2-chat:df7690f1994d94e96ad9d568eac121aecf50684a0b0963b25a41cc40061269e5',
                           input={"prompt": f"{string_dialogue} {prompt_input} Assistant: ",
                                  "temperature":temperature, "top_p":top_p, "max_length":max_length, "repetition_penalty":1})
    return output

# User-provided prompt
if prompt := st.chat_input(disabled=not replicate_api):
    st.session_state.messages.append({"role": "user", "content": prompt})
    with st.chat_message("user"):
        st.write(prompt)

# Generate a new response if last message is not from assistant
if st.session_state.messages[-1]["role"] != "assistant":
    with st.chat_message("assistant"):
        with st.spinner("Thinking..."):
            response = generate_llama2_response(prompt)
            placeholder = st.empty()
            full_response = ''
            for item in response:
                full_response += item
                placeholder.markdown(full_response)
            placeholder.markdown(full_response)
    message = {"role": "assistant", "content": full_response}
    st.session_state.messages.append(message)
```

## 6 我部署时解决的问题

(1) 首先，创建一个文件夹，用 vscode 打开文件

(2) 设置.env 虚拟环境

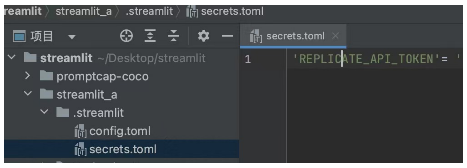
(3) 然后按照部署相关环境，比如 streamlit 和 republic

(4) 之后在此文件夹中新建一个文件夹.streamlit

(5) 在.streamlit 文件夹中新建 secrets.toml 文件, 然后如照片那样在其中输入之前的 Api

(6) 最后进行调用，记住一定要用

```
streamlit run main.py
```



```
streamlit ~/Desktop/streamlit
  promptcap-coco
  streamlit_a
    .streamlit
      config.toml
      secrets.toml
```

```
'REPLICATE_API_TOKEN'= '
```

最后成功了

## 7 声明

此篇文章深度借鉴了 https://blog.streamlit.io/how-to-build-a-llama-2-chatbot/如何构建 Llama 2 聊天机器人这篇文章，这篇文章只是记录一下自己的学习过程与学习心得，希望对于有所帮助