Documentation: Surya's Langchain Chatbot – Chat with SQL Database

Overview

This chatbot enables users to interact with an **SQL database (SQLite or MySQL)** using **LangChain** and **Groq's AI model**. It provides a conversational interface where users can query the database using natural language or SQL commands, and the AI will process and return relevant results.

Features

- **Database Interaction**: Supports SQLite and MySQL for querying structured data.
- **AI-Powered Query Processing**: Uses Groq's gemma2-9b-it model to interpret and execute queries.
- **User-Friendly Interface**: Built with Streamlit for a seamless experience.
- **Dynamic Database Configuration**: Adjusts automatically based on user input.
- **Session-Based Chat History**: Maintains previous conversations for continuity.

How It Works

1. User Interface

The chatbot runs on a **Streamlit-based UI** with a sidebar for selecting database options and entering a **Groq API key**. Without a valid API key, the chatbot will not proceed.

2. Database Connection

The system supports **SQLite (default)** and can be extended to **MySQL**. If SQLite is chosen, it connects to a predefined database file. For MySQL, users need to provide **host, username, password, and database name**.

3. AI Model for Query Processing

Groq's gemma2-9b-it language model processes user queries. It understands both **natural language** and **direct SQL queries**, translating them into executable database commands when needed.

4. SQL Agent Execution

The chatbot uses a **LangChain SQL Agent**, which:

- Parses user queries.
- Determines the best approach to fetch relevant data.
- Executes SQL queries against the database.
- Returns meaningful responses.

5. Chat History Management

The chatbot maintains **session-based chat history**, ensuring that previous interactions are accessible and the conversation remains context-aware. Users can also clear chat history from the sidebar.

How to Use

1. **Run the chatbot** by executing the Streamlit script.
2. **Enter your Groq API key** in the sidebar.
3. **Select the database** you want to interact with.
4. **Ask a question** (e.g., *"How many students are in the database?"*).
5. **Receive AI-generated responses** based on database queries.

Future Improvements

- **Expanding MySQL Support**: Add interactive input fields for MySQL credentials.
- **Multi-Database Support**: Connect to PostgreSQL and other databases.
- **Enhanced Query Interpretation**: Improve natural language to SQL translation.
- **Advanced Security Features**: Add authentication for database access.

Code for this

```
import streamlit as st

from pathlib import Path

from langchain.agents import create_sql_agent

from langchain.sql_database import SQLDatabase

from langchain.agents.agent_types import AgentType

from langchain.callbacks import StreamlitCallbackHandler

from langchain.agents.agent_toolkits import SQLDatabaseToolkit

from sqlalchemy import create_engine

import sqlite3

from langchain_groq import ChatGroq

# Setting up the title

st.set_page_config(page_title="Surya's Langchain: Chat with SQL DB")

st.title("Surya's Langchain Chatbot: Chat with SQL DB")

localdb = "use_localdb"

mysql = "use_mysql"

radio_opt = ["Use SQLite 3 with my Database"]

selected_opt = st.sidebar.radio(label="Choose the DB you want to chat with",
options=radio_opt)
```

```
if radio_opt.index(selected_opt) == 0:

   db_uri = localdb

# Get API key

api_key = st.sidebar.text_input(label="Groq API key", type="password")

# Validate required inputs

if not db_uri:

   st.info("Please enter the database information.")

   st.stop()

if not api_key:

   st.info("Please add your Groq API Key.")

   st.stop()

# LLM Model

llm = ChatGroq(groq_api_key=api_key, model="gemma2-9b-it", streaming=True)

# Function to configure the database

@st.cache_resource(ttl=7200)

def configure_db(db_uri, mysql_host=None, mysql_user=None, mysql_password=None,
mysql_db=None):

   if db_uri == localdb:

       dbfilepath = Path("student.db").absolute()
```

```
        creator = lambda: sqlite3.connect(f"file:{dbfilepath}?mode=ro", uri=True)

        return SQLDatabase(create_engine("sqlite:///", creator=creator))

    elif db_uri == mysql:

        if not (mysql_host and mysql_user and mysql_password and mysql_db):

            st.error("Please provide all MySQL connection details.")

            st.stop()

        return
SQLDatabase(create_engine(f"mysql+mysqlconnector://{mysql_user}:{mysql_password}
@{mysql_host}/{mysql_db}"))

# Configure DB connection

if db_uri == mysql:

    db = configure_db(db_uri, mysql_host, mysql_user, mysql_password, mysql_db)

else:

    db = configure_db(db_uri)

# Initialize the agent

toolkit = SQLDatabaseToolkit(db=db, llm=llm)

agent = create_sql_agent(

    llm=llm,

    toolkit=toolkit,

    verbose=True,
  agent_type=AgentType.ZERO_SHOT_REACT_DESCRIPTION

)

# Store chat messages in session state

if "messages" not in st.session_state or st.sidebar.button("Clear message history"):

  st.session_state["messages"] = [{"role": "assistant", "content": "How can I help you?"}]

for msg in st.session_state.messages:

  st.chat_message(msg["role"]).write(msg["content"])

# Handle user input

user_query = st.chat_input(placeholder="Ask anything from the Database")

if user_query:

  st.session_state.messages.append({"role": "user", "content": user_query})

  st.chat_message("user").write(user_query)

  with st.chat_message("assistant"):

    streamlit_callback = StreamlitCallbackHandler(st.container())

    response = agent.run(user_query, callbacks=[streamlit_callback])

    st.session_state.messages.append({"role": "assistant", "content": response})

    st.write(response)
```