# **Self-Learning Chatbot with OpenHermes**

## ****1. Objective****

To develop a chatbot interface powered by the OpenHermes model via Ollama, which can:

* Take prompts from users.
* Retrieve and include recent chat history from a database.
* Use uploaded PDFs for contextual understanding.
* Store new prompt-response pairs for continual improvement (self-learning simulation).

## ****2. System Architecture****

┌─────────────┐

│ User Input │

└─────┬───────┘

│

▼

┌─────────────┐ PDF Upload

│ Streamlit ├─────────────► Extract Text (PyPDF2)

└─────┬───────┘

│

▼

┌──────────────┐

│ Postgres DB │◄───── Save Prompt-Response

│ chat\_history │─────► Fetch Last N Conversations

└─────┬────────┘

│

▼

┌──────────────┐

│ OpenHermes │ (via Ollama API)

└──────────────┘

## ****3. Database Setup****

### PostgreSQL Table

CREATE TABLE chat\_history (

id SERIAL PRIMARY KEY,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

prompt TEXT,

response TEXT

);

* prompt: User's question
* response: Model's answer
* timestamp: When the chat occurred (default: current time)

## ****4. Streamlit Application****

### ****Required Libraries****

* streamlit – for building the web app interface
* requests – to call the Ollama API
* PyPDF2 – to extract text from uploaded PDFs
* psycopg2 – for PostgreSQL interaction

Install with: pip install streamlit requests PyPDF2 psycopg2

## ****5. Complete Code****

import streamlit as st

import requests

import PyPDF2

import psycopg2

from datetime import datetime

# --- Database Connection ---

def insert\_chat(prompt, response):

try:

conn = psycopg2.connect(

host="localhost",

database="SLM",

user="postgres",

password="Vallika@2003"

)

cursor = conn.cursor()

cursor.execute(

"INSERT INTO chat\_history (prompt, response) VALUES (%s, %s)",

(prompt, response)

)

conn.commit()

cursor.close()

conn.close()

except Exception as e:

st.error(f"DB Error: {e}")

def fetch\_recent\_history(n=5):

try:

conn = psycopg2.connect(

host="localhost",

database="SLM",

user="postgres",

password="Vallika@2003"

)

cursor = conn.cursor()

cursor.execute("SELECT prompt, response FROM chat\_history ORDER BY timestamp DESC LIMIT %s", (n,))

records = cursor.fetchall()

cursor.close()

conn.close()

return records

except Exception as e:

st.error(f"DB Error while fetching history: {e}")

return []

# --- Streamlit UI ---

st.title("💬 Chat with OpenHermes (Ollama)")

# Upload PDF

uploaded\_pdf = st.file\_uploader("📄 Upload a PDF", type=["pdf"])

# Extract text

pdf\_text = ""

if uploaded\_pdf is not None:

reader = PyPDF2.PdfReader(uploaded\_pdf)

for page in reader.pages:

pdf\_text += page.extract\_text() or ""

# User prompt

user\_input = st.text\_area("💬 Enter your question:", height=100)

if st.button("Send") and user\_input.strip():

# Fetch past history

history = fetch\_recent\_history()

past\_context = "\n".join([f"Q: {p}\nA: {r}" for p, r in history])

# Build prompt

if pdf\_text:

final\_prompt = f"""You are an assistant. Here's a document provided by the user:

---

{pdf\_text}

---

Previous Q&A:

{past\_context}

Current Question:

{user\_input}

"""

else:

final\_prompt = f"""

Previous Q&A:

{past\_context}

Current Question:

{user\_input}

"""

# Call Ollama API

response = requests.post(

"http://localhost:11434/api/generate",

json={

"model": "openhermes",

"prompt": final\_prompt,

"stream": False

}

)

if response.ok:

result = response.json()

answer = result["response"]

st.text\_area("🧠 Response:", value=answer, height=200)

insert\_chat(user\_input, answer)

else:

st.error("❌ Failed to get response from Ollama.")

## ****6. Self-Learning Simulation Logic****

* The system does **not train OpenHermes** in the traditional sense (e.g., fine-tuning or updating weights).
* However, it **simulates self-learning** by:
  + Storing every interaction in the database.
  + Reusing recent prompt-response pairs as **context** in new prompts.
  + Thus, it **appears** to "remember" past interactions.

## ****7. How to Check Self-Learning Behavior****

You can verify that the system is using past context by:

* Asking related follow-up questions.
* Checking if it refers to earlier answers.
* Viewing the chat\_history table in PostgreSQL to confirm the prompt-response logs.

Example SQL query:

SELECT \* FROM chat\_history ORDER BY timestamp DESC LIMIT 10;

## ****8. Conclusion****

This project provides a lightweight, database-connected, context-aware chatbot interface using OpenHermes and Ollama. It demonstrates:

* Simple database integration for long-term memory.
* Contextual chaining for simulated learning.
* Extendability to incorporate user documents via PDF parsing.

For future improvements:

* Add a UI component to view past chats.
* Implement user authentication and session-based memory.
* Optionally explore RAG (Retrieval Augmented Generation) to enhance factual grounding.



