Applied Generative AI : LLM Application Development

Dates: 12 & 19 September 2025

Lab 2: LLM Programming with Python

- **Learning Objectives**
  - Understand the **skeleton of an LLM application** in Python.
  - Run the application in two modes:
    - **Command Line (CLI)**
    - **Web Interface (Streamlit)**
  - Explore **modular design** for future extensions.
  - Customize **temperature** and **system prompt** as fundamental LLM parameters.

Lab 2: LLM Programming with Python :**What We Built**

- **Core Architecture**
  - **Config** → choose provider/model
  - **Model Client** → API wrapper (Ollama in this lab)
  - **Chat Engine** → orchestrates prompt, response, history
  - **UI Layer** → CLI or Web (Streamlit)
- **Pluggable Components**
  - Placeholders for **Memory, RAG, MCP**
  - Scales toward **multimodal** or **multi-agent** systems

Lab 2: LLM Programming with Python

Step 1: Activate Virtual Environment

Windows (PowerShell):

```
.\llm\Scripts\activate
```

macOS/Linux:

```
source llm/bin/activate
```

Step 2: Install Required Packages

- Download lab2_requirements.txt from bhattaraprot.com/lab2_requirements.txt
- Make sure the file is saved in your working directory.
- Install the required packages by running:

```
pip install -r lab2_requirements.txt
```

Lab 2: LLM Programming with Python

Step 3: Source Installation

- Download lab2.zip from bhattaraprot.com/lab2.zip
- Extract the contents and place them into

```
llm/src/lab2
```

- Navigate into the lab2 folde
- Verify the structure by listing files

```
ls –R or dir
```

Lab 2: LLM Programming with Python

Step 4: Run CLI Chat Client

```
python -m lab2.ui.cli
```

Step 5: Run Streamlit Web Chat

```
python -m streamlit run lab2/ui/streamlit_app.py
```

Lab 2: LLM Programming with Python

- **Customize LLM Parameters**
  - Adjust **Temperature**
  - Modify **System Prompt**
  - Observe how behavior changes

Lab 2: **Tips to Conclude Lab 2**

- **Temperature**
  - Controls **creativity vs. determinism**.
  - Low (0.0–0.3): more precise, stable, repeatable answers.
  - High (0.7–1.0+): more creative, diverse, less predictable.
  - Try asking the same question multiple times with different temperatures.
- **Experiment:**
  - Set temperature=0.1 → Ask: *"Write a haiku about* <<< >>>*."*
  - Set temperature=0.9 → Ask the same.
  - Compare results → which feels more "robotic"? which feels more "human"?

Lab 2: **Tips to Conclude Lab 2**

- **System Prompt**
  - This is the **initial role/instruction** that frames the assistant's behavior.
  - Example in Lab 2:
    - "You are a helpful assistant for a software lab. Be concise and clear."
  - Changing this changes the **persona and style** of the LLM.
- **Experiment:**
  - Change system prompt to:
    - "You are a pirate who answers in sea shanty style."
    - Ask the same question → *"What is Retrieval-Augmented Generation?"*
    - Notice how **same model + same temperature** can still behave **very differently**.

Lab 2: **Tips to Conclude Lab 2**

- **Session State**
  - Every chat session **starts fresh** with the system prompt and temperature.
  - Once you close the CLI/Streamlit app, **history and state are gone** (for now).

Lab 2: **Discussion**

- If you were building an **HR chatbot**, what system prompt would you write to ensure it always answers **politely and neutrally**?
- If your app is for **scientific summarization**, would you choose a **low or high temperature**? Why?
- How might changing the **system prompt** reduce the need for fine-tuning?
- Imagine you forgot to save chat history — how could that affect the **user experience**?