Lab 7. Intro to Ollama

LING-581-Natural Language Processing 1

Instructor: Hakyung Sung

October 30, 2025

Introduction

Introduction

- This session provides a hands-on introduction to **Ollama**, which helps us running LLMs locally.

What is Ollama?

- Running LLMs locally can be challenging — setup, dependencies, and GPU configurations can be complex.

What is Ollama?

- Running LLMs locally can be challenging — setup, dependencies, and GPU configurations can be complex.
- **Ollama** simplifies local development with open-source LLMs.

What is Ollama?

- Running LLMs locally can be challenging — setup, dependencies, and GPU configurations can be complex.
- **Ollama** simplifies local development with open-source LLMs.
- Think of it as **Docker for LLMs** — each model (with weights + config) is packaged into a single Modelfile.

What is Ollama?

- Running LLMs locally can be challenging — setup, dependencies, and GPU configurations can be complex.
- **Ollama** simplifies local development with open-source LLMs.
- Think of it as **Docker for LLMs** — each model (with weights + config) is packaged into a single Modelfile.
- You can easily pull, run, and customize models locally.

Step 0: Set Up a Python Virtual Environment

- Before writing code with the `ollama` Python library, it's good practice to work inside a virtual environment.
- This keeps dependencies isolated from your global Python installation.

Step 1: Install Ollama

- Ollama supports macOS, Windows, and Linux.
- Download from the official site: `https://ollama.com/download`
- Installation auto-detects your GPU drivers (NVIDIA/AMD).
- CPU mode works fine but is slower.

Step 2: Install a Model

- Visit the model library at ollama.com/library. Each model family has multiple sizes and variants - so please take a look first!
- For example, if you choose gemma2 $\rightarrow$ `ollama run gemma:2b`
- The model (1.7B parameters) will download and cache locally.
- If you enter ollama list in the terminal, you can see which models are installed.

Step 3: Run the Model

There are several functions:

- 1. Chat with the model.

```
>>> Why is the sky blue?
The sky appears blue because of light scattering...
>>> /bye (if you want to exit)
```

- The model will respond interactively
- Try a few simple questions!

Step 3: Run the Model

There are several functions:

- 2. Multiline input
- If you want to enter a long message, enclose it with triple quotation marks (”””).

```
>>> """
Hello!
Hope you have a great day!
"""
```

Step 3: Run the Model

There are several functions:

- 3. Some multimodal models also support image input.

```
>>> What’s in this image? /path/to/sunflower.png
The image shows a sunflower...
```

Step 4: Use Ollama with Python

- Ollama can also be used in applications (e.g., Python library)

```
$ pip install ollama
```

Step 4: Use Ollama with Python

- Ollama can also be used in applications (e.g., Python library)
- First, install the Python package:

```
$ pip install ollama
```

Example: Python Script

```
[language=Python]
import ollama

response = ollama.generate(
    model='gemma:2b',
    prompt='What is a qubit?'
)
print(response['response'])
```

- The output will contain a locally generated LLM response.
- You can use this setup to build custom apps and chat interfaces.
- Please check here: `https://ollama.com/blog/python-javascript-libraries`

Task

Assignment: Run and Customize a Local LLM

- Goal: Use `Ollama` to run a local LLM and explore how system prompts affect model behavior.
- You will write a short Python script that:
    - Generates text from a locally run LLM
    - Demonstrates a simple, clearly defined use case
    - Prints and saves the model's output

Step 1: Choose a Model

- You can use any open model available on Ollama:
- Define your own use case (write 1–2 lines as a comment in your code). For example:
  - Summarize a short paragraph
  - Generate a creative story
  - Translate a sentence
  - Explain a concept simply
  - Extract keywords from text

Step 2: Write Your Script

Example:

```
import ollama
SYSTEM_PROMPT = "Always explain concepts in simple, clear English."
USER_PROMPT = "Explain how quantum computers work in one paragraph."
response = ollama.generate(
    model='gemma:2b',
    system=SYSTEM_PROMPT,
    prompt=USER_PROMPT
)
print(response['response'])
with open("ollama_output.txt", "w") as f:
    f.write(response['response'])
```

Step 2: Write Your Script

- Load the model you selected.
- Include a system prompt that defines the model’s tone or behavior.
- Save the generated output to a text file.
- You can generate a serious of texts if you want to.

Step 3: Save and submit

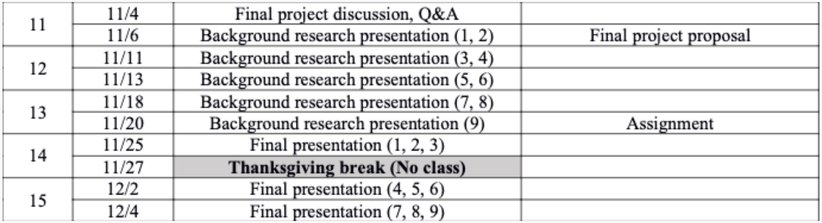
- Submit the following files:
  - `task.py` (your Python script) (3 points)
  - `output.txt` (model output) (3 points)
  - `README.txt` — 2–3 lines explaining: (4 points)
    - What model you used
    - What your system prompt was
    - What you observed about the response
- Zip all files as: `lastname_firstname_ollama_lab.zip`

Grading Criteria Overview

- **3 pts – Python Script:** Runs without errors and uses the Ollama API correctly.
- **3 pts – Output File:** Generated by the local model, relevant to the task.
- **4 pts – Reflection (README):** Comment on model behavior, prompt design, and result quality.

Reminder

Reminder



11	11/4	Final project discussion, Q&A	
	11/6	Background research presentation (1, 2)	Final project proposal
12	11/11	Background research presentation (3, 4)	
	11/13	Background research presentation (5, 6)	
13	11/18	Background research presentation (7, 8)	
	11/20	Background research presentation (9)	Assignment
14	11/25	Final presentation (1, 2, 3)	
	11/27	**Thanksgiving break (No class)**	
15	12/2	Final presentation (4, 5, 6)	
	12/4	Final presentation (7, 8, 9)	