LLM Self-hosted

APPLIED LLM

Ollama: Empowering Local LLM Deployment on Your Machine

By admin 11/04/2024

Ollama is an open-source platform designed to facilitate local LLM deployment across the three major operating systems: Windows (preview), MacOS, and Linux. Equipped with multiple interaction methods including Command Line Interface (CLI), GUI, and compatibility with OpenAI API, Ollama offers comprehensive support for seamless integration and utilization.

Note: This post isn't a step-by-step tutorial for installing Ollama, it serves as a comprehensive FAQ for installing and utilizing Ollama to serve LLM locally.

Contents [hide]

Local LLM News

- 19 Apr 2024: MetaAI release LLAMA3, the model archieve the very good performace compare to other, you can download via ollama here.

Ollama overview

> Note: You should have at least 8 GB of RAM available to run the 7B models, 16 GB to run the 13B models, and 32 GB to run the 33B models.
>
> If you have GPU, VRAM then will serve for faster LLM response.

What can Ollama do?

- Self-hosted a lot of "top" opensource LLM, including LLAMA2 (by Facebook), Mistral, Phi (from Microsoft), Gemma (by Google), ... You can see the full list here
- Multi-platform support: Windows (preview), MacOS, and Linux.
- Support self-hosted opensource embedding model (eg. nomic), and visio model (eg. LLaVA).

- Support serveral GPU types: NVIDIA, AMD and Apple GPU.
- OpenAI compatible API

Some features of Ollama?

- Deploy a model with custom parameters
- Deploy a custom model from .GGUF format.
- 4-bit quantization (save memory).

What is the know limit of Ollama?

- Can only handle request one-by-one.

Install Ollama

To install Ollama on your machine, please follow the latest install document from the author. Visit this download page and pick your OS to read the install instruction.

For Docker deployment, please read the instruction in the README of the project.

Because Ollama is under rapid development, new features are introduced frequently. However, it's important to note that some features are exclusively available with the latest versions. We encourage you to explore the release page regularly to stay updated on the newest features and enhancements.

> The time I write this post, Ollama latest version is `v0.1.31` (Apri 7th 2024).

For me, the install and update using Linux (Ubuntu, Fedora) all the same curl command:

```
curl -fsSL https://ollama.com/install.sh | sh
```

COPY

> I also not tested deploy Ollama on a machine without GPU. But seem Ollama can also deploy LLM without GPU.

Install GUI app/web (Optional): For who want to have an UI to interaction (website or application), you need to install additional community app, pick one you like here.

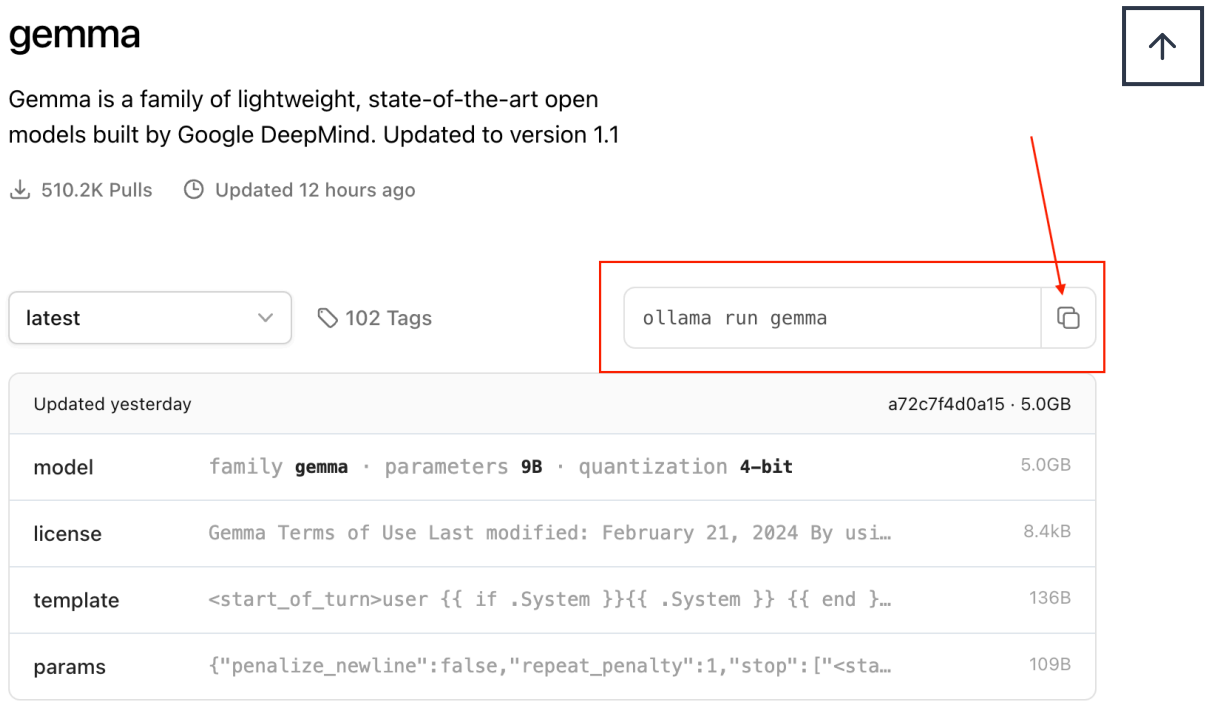
Deploy a model

This section is talk about my experiment on Linux/MacOS, both using command line to deploy a model. Not sure Windows the same (using PowerShell/CMD) or have a better UI management.

Simple deployment

The easy way to deploy a LLM is search for a favourite model from library, then run the command in the instruction.

Here is an exaple to run Gemma 7B model from Terminal:




```
1 | ollama run gemma
```

COPY

Advanced deployment

Ollama have feature named "Modelfile" to help people share their model to Ollama libray. This also a feature allow you customize your deployed model.

Using Modelfile, you have the capability to inherit an existing Large Language Model (LLM) while customizing parameters such as prompt, seed, temperature, and more.

Example:

In my case using LLM for RAG Chatbot, I want the LLM:

- Return the same answer on every request when the input not change, its mean I need to set a fix `seed`.
- More focused and deterministic, don't want the model *creative*, its mean I need to reduce the `temperature`.

So, I create a Modelfile localy with the following content:

COPY

```
# Modelfile generated by "ollama show"
# To build a new Modelfile based on this one, replace t
FROM gemma:latest

TEMPLATE """<start_of_turn>user
{{ if .System }}{{ .System }} {{ end }}{{ .Prompt }}<en
<start_of_turn>model
{{ .Response }}<end_of_turn>
"""
PARAMETER stop "<start_of_turn>"
PARAMETER stop "<end_of_turn>"

# my changes
PARAMETER temperature 0.0
PARAMETER seed 1996
```

The other values in the Modelfile aboce are copy from the original Gemma Modelfile in the Ollama library.

Ollama FAQs

Calling Ollama via Python?

Below is Python code using Python requests to calling Ollama for:

- Embedding request
- Chat request
- Generate request

Notes:

- Currently, Ollama format could be `None` or `json`.
- `retried` is number of retried when failed (eg. json error, timeout, ...)
- Function will return `None` when still failed after all tries.

COPY

```
import requests
import json
import traceback
import logging

logger = logging.getLogger(__name__)

OLLAMA_TIMEOUT = 30 # seconds
EMBEDDING_TIMEOUT = 30
OLLAMA_HOST = "http://localhost:11434"

def ollama_chat(content, model="openhermes", format=Nor
    data = {
        "model": model,
        "messages": [{"role": "user", "content": conter
        "stream": False,
        "format": format,
    }
    try:
        response = requests.post(
            url=OLLAMA_HOST + "/api/chat",
            data=json.dumps(data),
```

```
            timeout=OLLAMA_TIMEOUT,
        )
        response.raise_for_status()
        if format == "json":
            return json.loads(response.json()["message"
        return response.json()["message"]["content"]
    except Exception:
        logger.exception("Request ollama_chat error!")
        traceback.print_exc()
        if retried <= 0:
            return None
        return ollama_chat(content, model=model, format

def ollama_generate(content, model="openhermes", format
    data = {"model": model, "prompt": content, "stream"
    try:
        response = requests.post(
            url=OLLAMA_HOST + "/api/generate",
            data=json.dumps(data),
            timeout=OLLAMA_TIMEOUT,
        )
        response.raise_for_status()
        if format == "json":
            return json.loads(response.json()["response
        return response.json()["response"]
    except json.decoder.JSONDecodeError:
        return ollama_generate(content, model=model, fo
    except Exception:
        if retried <= 0:
            logger.exception("Request ollama_generate e
            raise
        return ollama_generate(content, model=model, fo

def embed(text):
    try:
        response = requests.request(
            "POST",
            EMBEDDING_HOST + "/api/embed",
            headers={"Content-Type": "application/js
            data=json.dumps({"docs": text}),
            timeout=EMBEDDING_TIMEOUT,
```

```
            )
            return response.json()["embeddings"]
        except Exception:
            logger.exception("Request embedding error")
            return None
```

How to increase LLM context length in Ollama?

When using API, add the `num_ctx` in the `options` parameters, like this:

```
curl http://localhost:11434/api/generate -d '{
  "model": "gemma",
  "prompt": "Why is the sky blue?",
  "options": {
    "num_ctx": 4096
  }
}'
```

Allow calling Ollama by from another PC / from browser?

> Notes: This is for Linux OS.

By default, Ollama host is `localhost`, this mean you can not calling Ollama from another computer.

Also, you can not call Ollama from browser (web calling) because the problem of CORS.

This configuration below allow you can do the above jobs:

```
# This will open Ollama service config using Vim editor
sudo vim /etc/systemd/system/ollama.service

# Then add these two lines with your value you want
```

```
Environment="OLLAMA_HOST=0.0.0.0:11434"
Environment="OLLAMA_ORIGINS=https://behitek.com"
```

Then, you need to reload the service

```
sudo systemctl daemon-reload
sudo systemctl restart ollama.service
```

In some case, you have firewall blocking from allow you calling from external. In this case, update your firewall to allow port 11434.

```
sudo ufw allow 11434
sudo ufw reload
```

How to make Ollama always ready to serve?

By default, if models are not using after 5 minutes, Ollama will unload the model for memory saving. Then, next time you calling Ollama will take longer because the model need to load into RAM first.

To prevent this memory saving, make a request to the model

```
curl http://localhost:11434/api/generate -d '{"model":
```

To undo, use this

```
curl http://localhost:11434/api/generate -d '{"model":
```

Where is Ollama FAQs?

Ollama itselft has a FAQ document, you can read it here

What can you do with local LLM?

- Privacy-first for any application using LLM.
- Cost saving, you don't have to pay any cost at a small level of demand.

Here are some idea a local LLM can help you:

- RAG chatbot (your home assistant)
- Blogging assistant
- Code assistant (like Github Copilot)
- Data processing, LLM can help you clean data by doing summarization, or convert data to a structure format, etc.

#llm #ollama

← PREVIOUS

Danh sách liên kết đơn (Singly Linked List)