UNIPD 2025
Lab session IV

Local running of LLM

Introduction

Massimo Brunelli

- Graduated in Padova: Chemical Engineering
- Software Developer
- Project Manager for Pharmaceutical industry machines
- Working experience in the robotic sector
- Currently in BSI/smartKYC as Head of Professional Services and Generative AI Technology Lead

Adverse Media Screening: Advanced, multilingual Natural Language Processing (NLP) to pinpoint risk in global web and media sources.

Source of wealth: Identify and corroborate the source of your clients wealth to ensure it corresponds with your data held on file.

Network Mapping & Relationship Risk: Reveal the whole network, from formal and official relationships to friends, associates and family members.

List Screening & Entity Resolution: Name screening that marries sophisticated matching technology with cultural sensitivity.

ESG Risk Intelligence: AI-powered ESG risk intelligence about your supplier, business partner or investee companies.

Periodic Refresh & Continuous Monitoring: Monitor entire client bases, periodically or continuously and receive genuine information deltas, not things you have seen already.

Robo onboarding: Robotically onboard your clients an industry-first approach that not only verifies identities in minutes but conducts full due diligence too.

Batch Remediation: Automatically screen an entire client base overnight. smartKYC will only highlight cases with potential risk.

Today’s goals

smartKYC

Enabling all students to have:

- one or more
- personal,
- local,
- offline,
- Large Language Model to experiment with.

To let you run your personal experiments and to learn more about:

- LLM Infrastructure
- LLM Programming
- LLM configuration and parametrization

Giving you a glimpse of what awaits you after that: there is so much more!

Ollama

smartKYC

What is it?

- Easy-to-install, multi-system, multi-LLM server, fully offline: ollama.com
  - Open source - https://github.com/ollama/ollama
  - Ready to use models - https://ollama.com/search
  - Importing models from Huggingface (OT) - https://huggingface.co/docs/hub/ollama
  - Models customization - https://github.com/ollama/ollama/blob/main/docs/modelfile.md
  - RESTful API server - https://github.com/ollama/ollama/blob/main/docs/api.md
  - Integrate with many tools - https://github.com/ollama/ollama/blob/main/docs/api.md
- It offers
  - Terminal-friendly interface
  - Autoscaling CPU/GPU processing
  - Private LLMs
  - RESTful API

Ollama

smartKYC

the cons

- Entry level
- Not ready for Enterprise installations
- Limited configurability for the LLM executions
- Advanced AI better supported by other tools
- Quantization/Security/LLM management/etc...

Ollama

smartKYC

Why?

- Costs
- Personal privacy
- Ease of use
- Early customization
- Start exploring new technologies:
  - Testing different LLMs
  - Multimodal LLM (OT)
  - RAG systems (OT)
  - Agents (OT)
  - Function calling (OT)

installation

- Minimum requirements:
  - Linux, Mac, Win
  - 10GB free
  - Modern CPU
  - GPU (optional)
  - RAM: “bring it to the party: the more the merrier”

Ollama

smartKYC

Where to start from

- (start from ollama.com)
- Install and update in Linux

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

- Running a model

```
ollama run modelname
```

- So many models

https://ollama.com/search

Introduction

What is it

- Open WebUI is an extensible, feature-rich, and user-friendly self-hosted AI platform designed to operate entirely offline. It supports various LLM runners like Ollama and OpenAI-compatible APIs, with built-in inference engine for RAG, making it a powerful AI deployment solution.

Requirements

- Modern desktop PC with Linux or Windows (or Mac, but never tested)
- Docker (OT)

Install and Run

- Install

```
docker pull ghcr.io/open-webui/open-webui:main
```

- Run

```
docker run -d -p 3000:8080 -e OLLAMA_BASE_URL=http://127.0.0.1:11434 --add-
     host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --
     name open-webui --restart always ghcr.io/open-webui/open-webui:main
```

See https://docs.openwebui.com/ for more options

Connect / Update

- Connect

  `http:/<server-address>:3000/auth`

  - Create Update an admin user
  - Customize the behavior
- Update

```
docker rm -f open-webui
docker pull ghcr.io/open-webui/open-webui:main
```

Some common infrastructure issues

smartKYC

Ollama

- installing Ollama
  - sometimes installing new models triggers the message to upgrade Ollama
  - Ollama doesn't listen to all addresses

```
/etc/systemd/system/ollama.service
    Environment="OLLAMA_HOST=0.0.0.0"
    Environment="OLLAMA_ORIGINS=*"
```

  - then

```
systemctl daemon-reload
systemctl restart ollama
systemctl status ollama
```

Nvidia/CUDA drivers

- NVIDIA drivers are usually preinstalled
- CUDA drivers
  - https://developer.nvidia.com/cuda-downloads
- NVIDIA drivers might not work while Secure Boot is enabled

Some common infrastructure issues

smartKYC

Docker networking

```
docker run
  -p 3000:8080 – publish port pc:container
  -e OLLAMA_BASE_URL=http://127.0.0.1:11434 – env variable
  --add-host=host.docker.internal:host-gateway
  -v open-webui:/app/backend/data – new volder in
  /var/lib/docker/volumes
  --name open-webui
  --restart always – if container stops
```

Models: loading and customizing

Testing an existing model

- REST API calls via curl

```
curl http://localhost:11434/api/generate -d '{
  "model": "gemma3:12b",
  "prompt":"Why is the sky blue? Answer in less than 30 words"
}'
```

Creating your own Modelfile

```
FROM gemma3:12b

# set the temperature to 1 or higher to make it more creative

PARAMETER temperature 0.3

SYSTEM """

You are Flash, a helpful generator of flashcards. Whatever question is asked you give a brief answer and
then you generate a list of flashcard to hel the user to learn the subject of the question. The flashcards
must be generated in json format so that they can be imported in a flashcard program

"""“


ollama create Flash -f ./FlashModelfile
```

Exercise: Connecting to LLM via REST API in Python

smartKYC

```python
import requests
import json

url = "http://{server-ip}:11434/api/generate"
headers = {
    "Content-Type": "application/json"
}

data = {
    "model": "gemma3:12b",
    "prompt": "Why is the sky yellow?",
    "stream": False
}

response = requests.post(url, headers = headers, data=json.dumps(data))

if response.status_code == 200:
    response_text = response.text
    data = json.loads(response_text)
    actual_response = data["response"]
    print(actual_response)
else:
    print("Error:", response.status_code, response.text)
```

Computational resources

Ollama

- Running a model
  - CPU requirements:
    - «11th Gen Intel CPU or Zen4-based AMD CPU, beneficial for its AVX512 support which accelerates matrix multiplication operations needed by AI models. CPU instruction set features matter more than core counts, with DDR5 support in newer CPUs also important for performance due to increased memory bandwidth. »
  - RAM and VRAM requirements
    - 7B model requires ~4 GB
    - 13B model requires ~8 GB
    - 30B model needs ~16 GB
    - 65B model needs ~32 GB
- Disk space for models
  - 50GB SSD should be enough

`/usr/share/ollama/.ollama/models  (ncdu is nice)`

And after that?

smartKYC

Before leaving Ollama

- Importing models from Huggingface
- Multimodal LLM
- RAG systems
- Agents
- Function calling

And after Ollama?

- Vast ecosystem
- Worth mentioning: https://github.com/ggml-org/llama.cpp