Teach Yourself to Run LLMs (Large Language Models) in your local computer. Free.

 https://TrialTwin.com

Hands-on Tutorial to Run LLMs Locally

You'll learn to run Large Language Models ("LLMs") in your local computer. You'll take different types of data (PDFs, SQLite3 files, PostgreSQL tables) to build and run a RAG ("Retrieval-Augmented Generation") pipeline. You'll get answers from a generic LLM using your own data as a reference. All this at no cost, in your local computer, in under 60 minutes. You will learn enough so that you can have a fruitful conversation with very technical people.

My teaching style is to start with a Minimum Working Demo ("MWD"). Then I'll expand from there, line by line. You can find both sample code and sample data in my personal GitHub page for this project: https://github.com/JLacal/local-llm

Please notice that the provided sample data files are related to clinical trials: that's our core business. Try to grasp the overall concepts. Just replace the clinical trial-specific PDFs and database files with your own files and the basic principles and processes will apply as well.

LLMs are a phenomenal piece of technology for knowledge generation and reasoning. They are pre-trained on large amounts of publicly-available data. The question is how to augment LLMs with your private and domain-specific data? You'll learn that in this document.

Why running the LLMs locally? Several reasons:
* even old hardware can be both useful and usable for learning the basics of LLMs
* you'll save money by learning locally (you can quickly run up a huge AWS / Azure / OpenAI bill by aimlessly playing around)
* you'll get a "feel" for the process (literally, like when your laptop's fan starts screaming)
* and you'll protect your private data from the endlessly prying eyes of cloud providers

All the software packages mentioned in this document are free to download and use.
And the data files we provide are also free to use: https://github.com/JLacal/local-llm

Contact Us

Please send me any comments and suggestions for improvement. Thank you for reading.

And contact us if your organization would like to have our team of experts build a customized, LLM-powered application for you. We work on a fixed budget / timeline basis.

José C. Lacal | +34 (674) 88 17 52 | Jose.Lacal@DataSDR.com

Summary

This is the structure of this file.

Table 1: Steps.

Step	Description
Set up Virtual Environment	Start by setting up a Python virtual environment.
Download required software	Download desktop applications you'll use.
Download Python files	Download Python files I provide as part of this tutorial.
Download data files	Download sample data files for you to use.
Run Python files	Actually run the Python files with the data files.

The entire process listed below was tested in the following computers:

Table 2: Tested computers.

iMac27	Mac mini	MacBook Pro
		
iMac 27 – model 2017	Mac mini – model 2018	MacBook Pro – model 2021
4.2 GHz Quad-Core Intel Core i7	3.2 GHz 6-Core Intel Core i7	Apple M1 Pro
64 GB 2400 MHz DDR4	64 GB 2667 MHz DDR4	16 GB

Please notice that some of my computers are over 06 years old yet still functional for LLM-related exploratory work. Obviously you'll need some pretty powerful (mostly GPU-heavy) systems for production

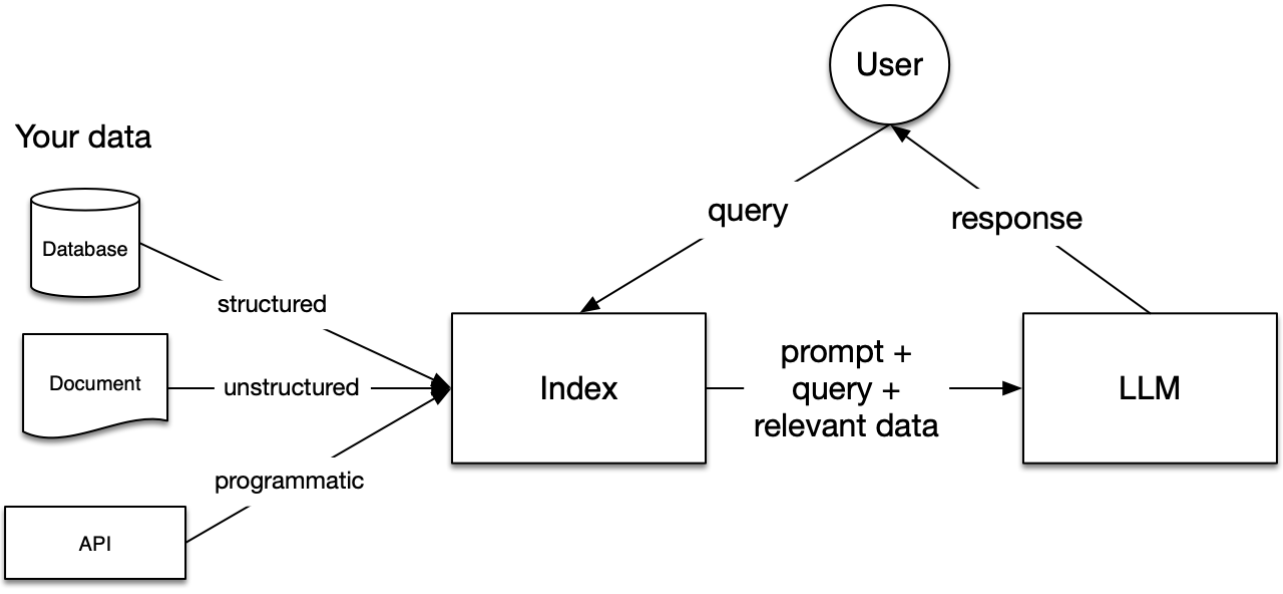
Background: Loading your Data into an LLM

Building your own LLM from scratch is a very expensive and time-consuming endeavor. Instead, in this document you will learn how to load your own data into an existing LLM to then be able to ask specific questions and get answers from your own data.

This process is called “creating a RAG” (Retrieval-Augmented Generation).
Please read this great article from AWS that explains what a RAG is at a high level:
https://aws.amazon.com/what-is/retrieval-augmented-generation/

> ..You can think of the Large Language Model as an over-enthusiastic new employee who refuses to stay informed with current events but will always answer every question with absolute confidence. Unfortunately, such an attitude can negatively impact user trust and is not something you want your chatbots to emulate!
> RAG is one approach to solving some of these challenges. It redirects the LLM to retrieve relevant information from authoritative, pre-determined knowledge sources. Organizations have greater control over the generated text output, and users gain insights into how the LLM generates the response.

The diagram below illustrates the process of building and running a RAG. You will:
* take different types of sample data (PDFs, SQLite3 files, PostgreSQL tables)
* create index files
* feed your question and the pre-generated index files to the LLM to get an answer



Drawing 1: Source: https://docs.llamaindex.ai/en/stable/getting_started/concepts/

Set up Virtual Environment

A “virtual environment” in Python is a way to create a constrained testing environment, without affecting the rest of your computer’s configuration.
Please see https://docs.python.org/3/library/venv.html for more details.

Make sure you have a Python3 version higher than 3.10
In macOS you can run: “brew install python@3.11”

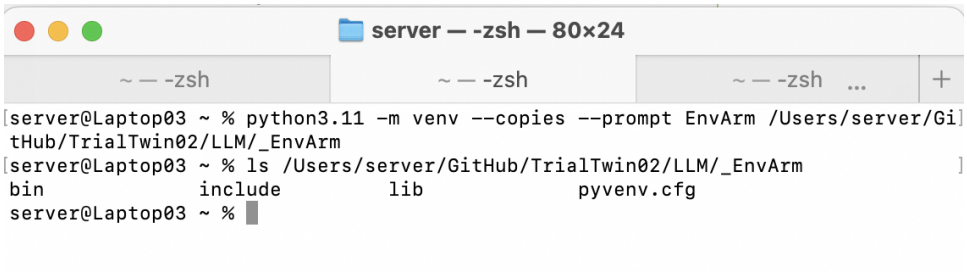
Now, create the VirtualEnv.
In the command line type something like the following:

python3.11 -m venv --copies --prompt EnvArm
/Users/server/GitHub/TrialTwin02/LLM/_EnvArm

Let me explain:

python3.11 -m venv :: “launch the Python3 interpreter and run the venv module”
--copies :: “Include copies of the required libraries inside the virtualenv”
--prompt EnvArm :: “the text that will appear in the prompt of the command line”
/Users/server/GitHub/TrialTwin02/LLM/_EnvArm :: “directory to store this into”

You should change the values highlighted in yellow to best fit your own needs.

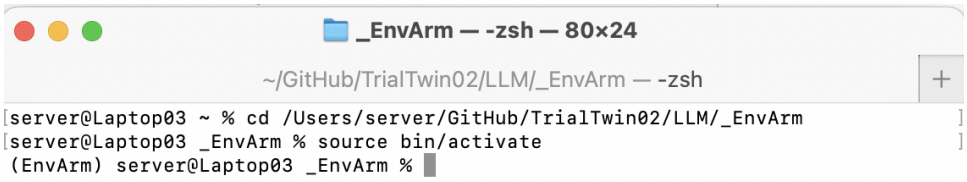


From now on, the directory you created the virtual environment into will be referred to as CT_VIRTUALENV_DIR in this document.

Change directory to CT_VIRTUALENV_DIR:
“cd /Users/server/GitHub/TrialTwin02/LLM/_EnvArm”

And type: “source bin/activate”

Notice in the screenshot below how the command line prompt now starts with “EnvArm” which is the prompt value we defined above.



Now you have a self-contained, isolated environment to install Python libraries and to run the Python files described below.

And the virtual environment is now “activated.” Meaning that all Python-related actions will run inside this virtual environment only.

Install Libraries

Now you will install a few Python libraries.

Most of these libraries are from LlamaIndex [https://docs.llamaindex.ai/en/stable/]

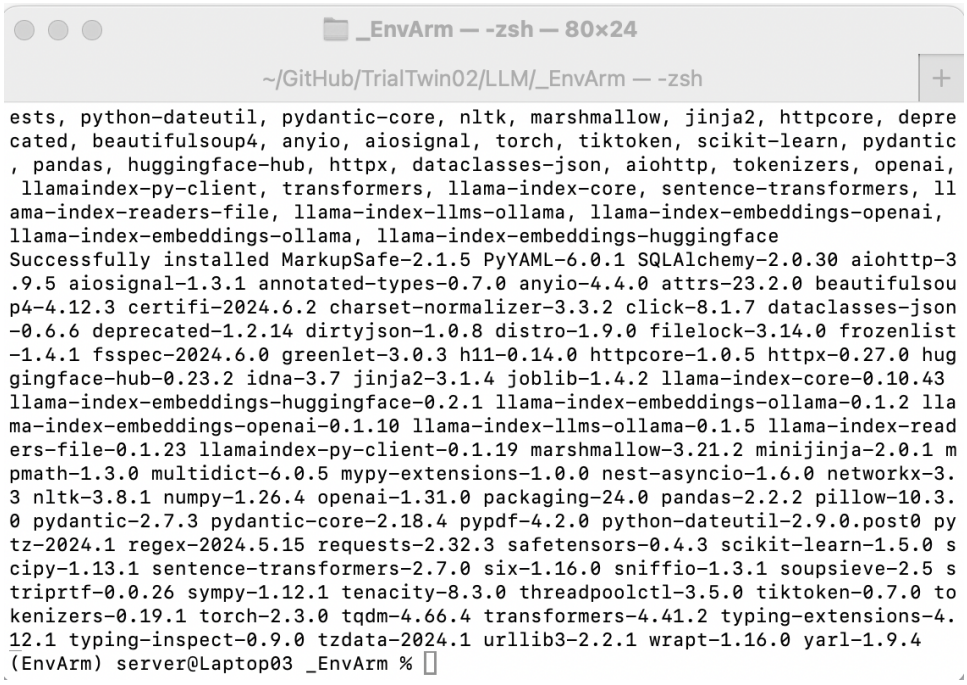
> LlamaIndex is a framework for building context-augmented LLM applications. Context augmentation refers to any use case that applies LLMs on top of your private or domain-specific data.

Now run these commands in the same command line that you have after activating the virtual environment:

```
pip install llama-index-core
pip install llama-index-embeddings-huggingface
pip install llama-index-embeddings-ollama
pip install llama-index-embeddings-openai
pip install llama-index-llms-ollama
pip install llama-index-readers-database
pip install llama-index-readers-file
```

Of course you can also run a single command:

```
pip install llama-index-core llama-index-embeddings-huggingface llama-index-embeddings-ollama llama-index-embeddings-openai llama-index-llms-ollama llama-index-readers-database llama-index-readers-file
```



Notice that the directory CT_VIRTUALENV_DIR will grow to around 01.4GB in size.

Download required software

Please notice that all the screenshots are from a macOS-based system. Adjust the commands to your specific OS.
And you need to be comfortable using the command line (Terminal / PowerShell).

Table 3: Required software.

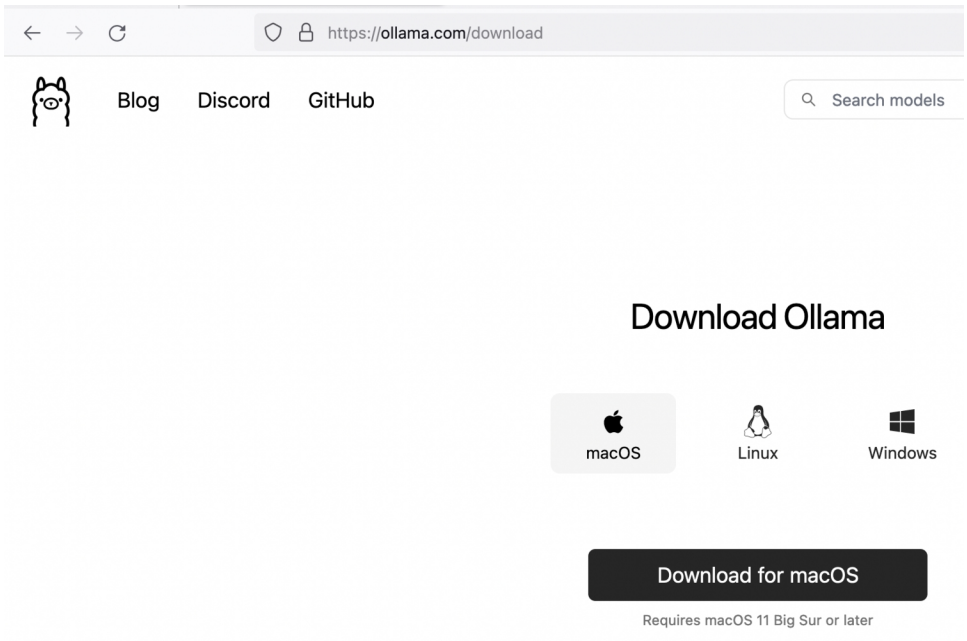
Software	Location
DB Browser for SQLite	https://sqlitebrowser.org/dl/
Ollama	https://ollama.com/download
LM Studio	https://lmstudio.ai/

DB Browser

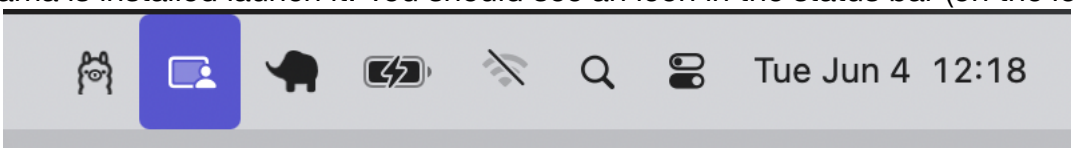
DB Browser allows you to work with SQLite database files.
Download and install the application.

Ollama

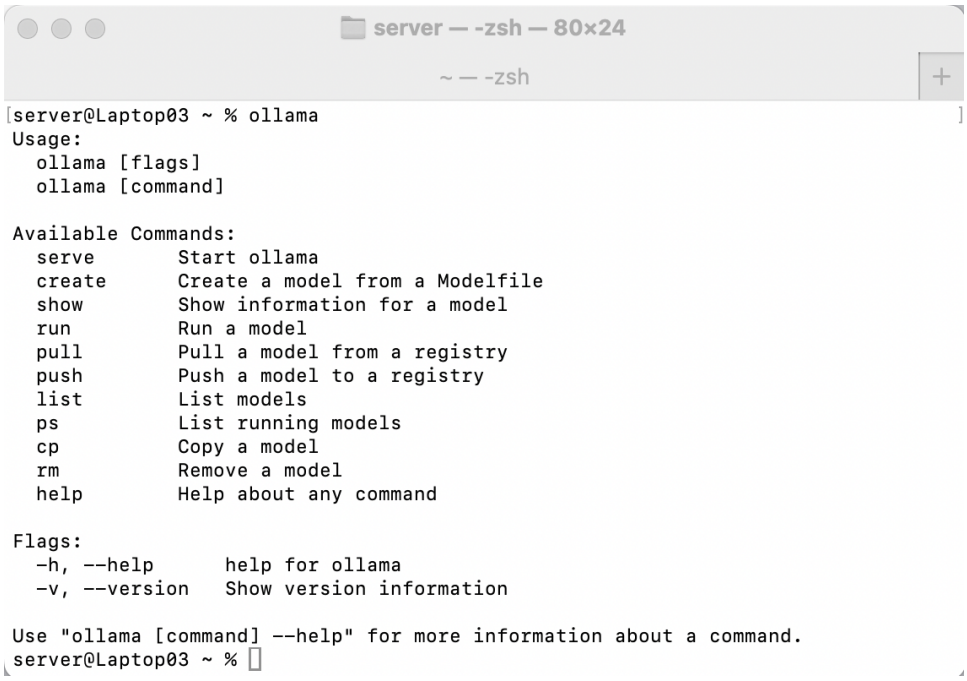
Ollama is a tool that makes it easy to run LLMs. And you can run the LLMs locally, without the need to connect to a cloud-based service.



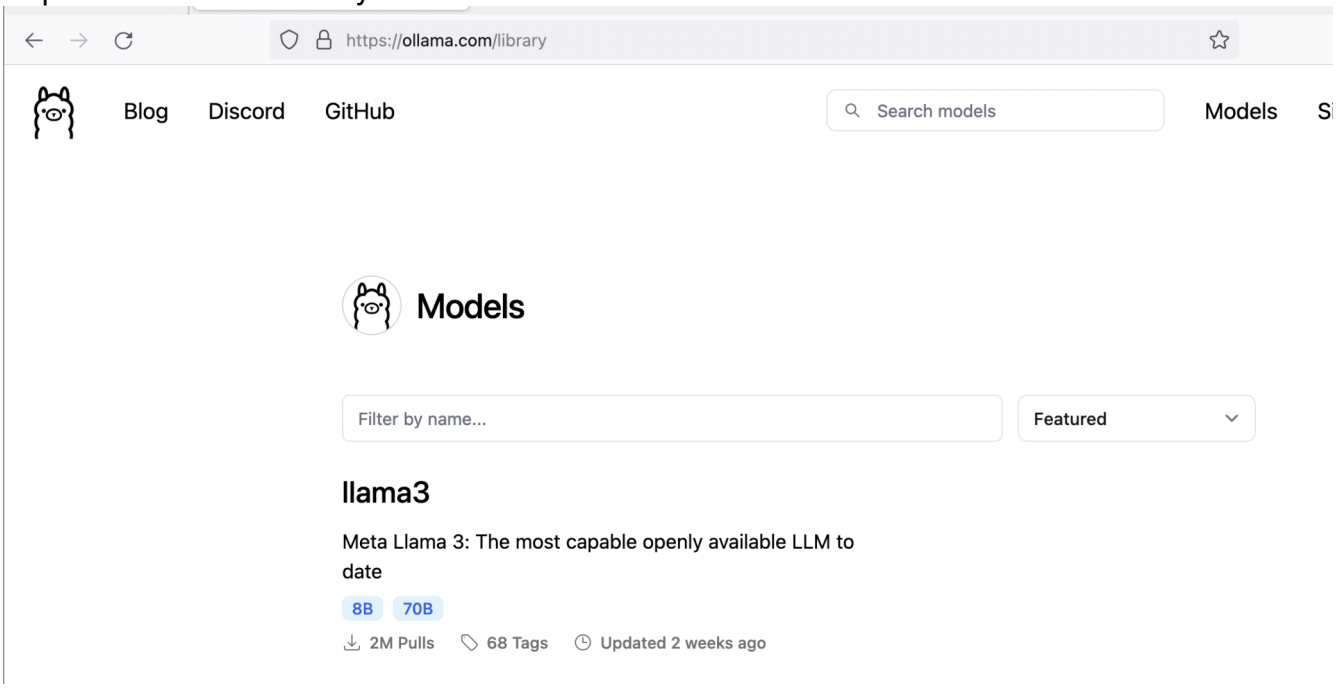
Once Ollama is installed launch it. You should see an icon in the status bar (on the left):



Verify that you can call Ollama from the command line.
Type “ollama” in the command line.



Now download a few models from Ollama's library of Open Source models. https://ollama.com/library



You can download each model by running the command "ollama pull" in the command line as follows:

ollama pull <ModelName>

I recommend you download at least these models:

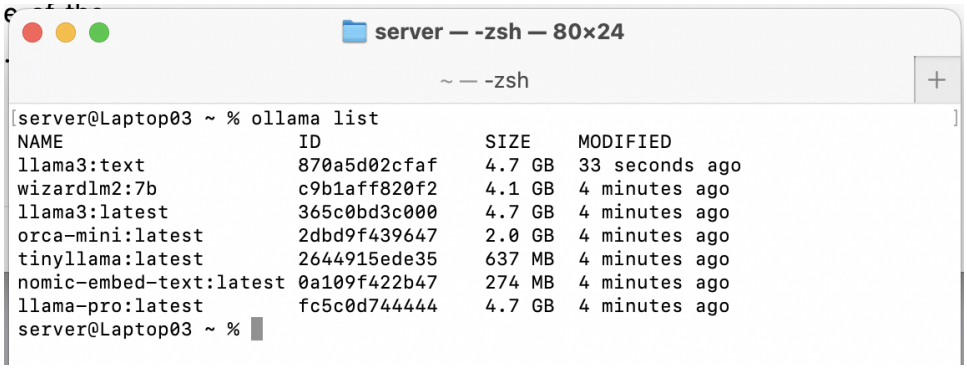
ollama pull llama-pro
ollama pull llama2
ollama pull llama3
ollama pull llama3:text
ollama pull orca-mini
ollama pull tinyllama
ollama pull wizardlm2:7b

And this embedding model:

ollama pull nomic-embed-text

The above models will take about 26 GB of storage space in your computer.

Run the command “ollama list” to see what models you have available in your computer:



When using macOS the model files are stored in the ~/.ollama directory.

Warning: models can be very large (some are around 80 GB in size like “wizardlm2:8x22b”). Make sure you have enough hard disk space to store as many models as you want to.

Additional models you may be interested in downloading are:

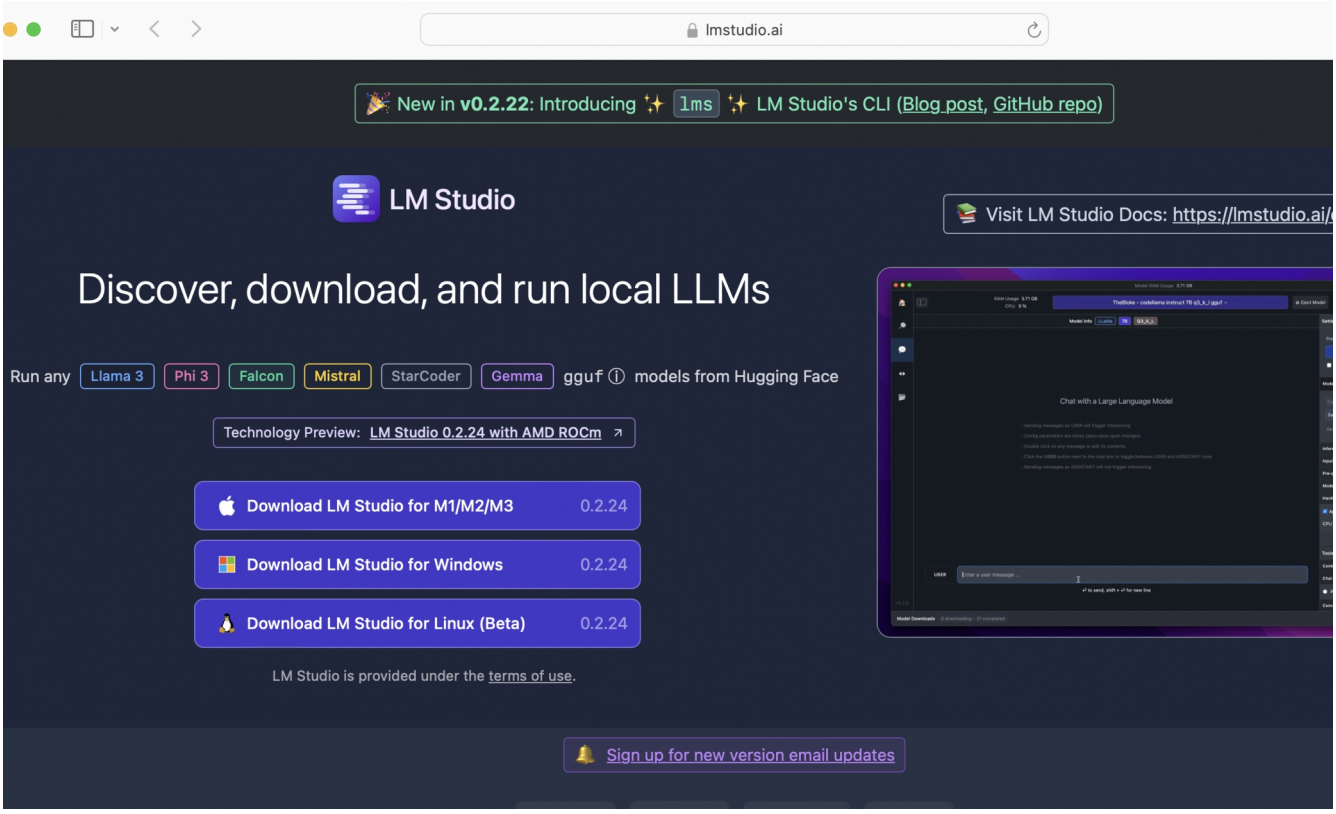
ollama pull codegemma
ollama pull command-r
ollama pull command-r-plus
ollama pull duckdb-nsql
ollama pull gemma:2b
ollama pull gemma:7b
ollama pull llama3:70b

```
ollama pull llama3:70b-text
ollama pull llama3-chatqa
ollama pull llama3-gradient
ollama pull meditron
ollama pull medllama2
ollama pull phi3:medium
ollama pull phi3:mini
ollama pull wizardlm2:8x22b
```

If you download all the additional models then you'll use around 310 GB of storage space.

LM Studio

LM Studio is a very slick GUI application that allows you to run LLMs locally through an easy-to-use graphical interface. No more typing commands in the command line.



Please use Ollama for the rest of this tutorial. Once you understand the basics then you can jump over to LM Studio.

If you decide to use LM Studio instead of Ollama MAKE SURE you delete the Ollama models to free up significant storage space.

LM Studio uses a different file format for the models than Ollama.

When using macOS the Ollama model files are stored in the ~/.ollama directory.

Download Python files

Now go to my personal GitHub repository and download these Python files that you will run locally.

https://github.com/JLacal/local-llm

Table 4: Python files.

File Name	Description
Tutorial_01.py	Step by step description of concepts related to augmenting a base LLM. This is a full Minimum Working Demo. https://github.com/JLacal/local-llm/blob/main/Tutorial_01.py
Tutorial_02.py	Ingest local PDFs and save the generated index files locally for faster re-load. https://github.com/JLacal/local-llm/blob/main/Tutorial_02.py
Tutorial_03.py	Ingest SQLite3 files. https://github.com/JLacal/local-llm/blob/main/Tutorial_03.py
Tutorial_04.py	Connect to PostgreSQL directly. https://github.com/JLacal/local-llm/blob/main/Tutorial_04.py

Kindly note that I'm aware of Jupyter and I use it myself.

At this stage I want to strongly encourage you to physically run these commands, and to install code locally in your computer: both actions will maximize your learning.

Download data files

Go to my GitHub repository and download these and other data files:

https://github.com/JLacal/local-llm

Each SQLite3 file contains details of clinical trials sponsored by each company. The files also contain the full text of individual Protocols, Statistical Analysis Plans ("SAP") and ICF ("Individual Consent Forms").

Table 5: Data files: SQLite3.

File	Description
Abbott	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Abbott.sqlite3.zip
Abbvie	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Abbvie.sqlite3.zip
AstraZeneca	https://github.com/JLacal/local-llm/blob/main/TrialTwin_AstraZeneca.sqlite3.zip
Bayer	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Bayer.sqlite3.zip
Bristol Myers Squibb	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Bristol_Myers_Squibb.sqlite3.zip
Johnson & Johnson	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Johnson_Johnson.sqlite3.zip
Pfizer	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Pfizer.sqlite3.zip
Roche	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Roche.sqlite3.zip
Sanofi	https://github.com/JLacal/local-llm/blob/main/TrialTwin_Sanofi.sqlite3.zip

Run Python files

OK, now we're ready for you to start having fun!

Open a command line window (Terminal / PowerShell).

Activate the Python virtual environment (see Set up Virtual Environment above).

Now open each individual Python file in your favorite text editor (NOT in Microsoft Word). Read the comments within the code and make the necessary adjustments for your own local environment.

Run each Python file and evaluate the results.

Lessons Learned: LLMs Lie.

Now for the bad news: LLMs lie, and they lie confidently!

I ran an analysis with 20 different LLMs processing the exact same data, and asking the exact same questions. The results fall under one of these categories:
* “Timed out” – my hardware was probably not capable enough for this (large) LLM
* “Lied” – the answer to the questions is clearly a lie
* “Answered” – the LLM provided an answer, not always a correct answer
* “Good” – the LLM actually provided a good, context-appropriate answer
* “Great” – the LLM provided a great, sometimes superb answer
* “Empty” – the LLM did not return anything for an answer

Timeout	codegemma	command-r	command-r-plus	duckdb-nsql	gemma:2b
300 ms	Lie	Timed out	Timed out	Answered	Lie
600 ms	Lie	Timed out	Timed out	Answered	Lie

Timeout	gemma:7b	llama-pro	llama3	llama3-chatqa	llama3-gradient
300 ms	Lie	Good	Great	Empty	Good
600 ms	Lie	Good	Great	Empty	Great

Timeout	llama3:70b	llama3:70b-text	llama3:text	meditron	medllama2
300 ms	Timed out	Timed out	Timed out	Lie	Good
600 ms	Timed out	Timed out	Timed out	Lie	Good

Timeout	phi3:medium	phi3:mini	tinyllama	wizardlm2:7b	wizardlm2:8x22b
300 ms	Timed out	Good	Great	Great	Timed out
600 ms	Good	Good	Great	Great	Timed out

These are examples of “Lies” from the specific LLM.

And I actually know that these are lies because the context provided to the LLM does contain the information requested.

```
= = = = = Using model [codegemma] = = = = =

The provided context does not contain any information about the NCT number
for fibrosis, so I cannot answer this question from the given context.
The provided context does not contain information about NCT 03656068, so I
cannot answer this query from the given context.
200.08059120178223 seconds ---


= = = = = Using model [medllama2] = = = = =

Based on the provided context, we cannot directly answer your question as
it would violate HIPAA's Privacy Rule by disclosing protected health
information (PHI). Therefore, I will not provide any personal information
related to patients' health status. It is important to respect individuals'
privacy and maintain confidentiality when handling sensitive information
like medical records.
```


And this an example of a “Great” answer.

This particular LLM ingested the exact same information as the LLMs listed above that “lied” and provided this answer:

```
= = = = = Using model [wizardlm2:7b] = = = = =

 The NCT number associated with the study mentioned in the provided
Statistical Analysis Plan (SAP) for evaluating the safety and efficacy of
NTZ at 500mg twice daily on collagen turnover in plasma in NASH patients
with fibrosis Stage 2 or 3 is NCT03656068. This study is investigating
various non-invasive markers of fibrosis, including biomarkers and fibrosis
scores like ELF test score, PIIINP, Hyaluronic acid, CK18, TIMP-1, YKL-40,
Alpha 2 macroglobulin, miRNA, ProC3, ProC6, FGF19, FGF21, NAFLD fibrosis
score, and FIB-4, among others.
 NCT 03656068 is a clinical trial with the protocol number NTZ-218-1,
version 1.1, dated April 9, 2020. The statistical analysis plan outlines
the study's overview, objectives, endpoints, design, drug information,
sample size determination, and statistical methodology, among other
aspects. The study involves various populations for analysis, including the
Full Analysis Set and the Safety Analysis Set, and includes assessments of
efficacy (such as Fractional Synthesis Rate, Non-invasive Markers of
Fibrosis, and Fibrosis Scores) and safety (including Adverse Events,
Clinical Laboratory Tests, Vital Signs, Electrocardiograms, and Physical
Examinations). The study also details how electronic medical records (EMRs)
and a clinical trial management system (CTMS) will be used to manage
participant data within the University of Pennsylvania Health System. The
purpose of collecting and managing this information is to conduct the
research, oversee it, and evaluate whether it was executed correctly.
```

Warning

I strongly recommend that Subject Matter Experts (“SMEs”) must review the results of each chosen LLM before you conclude a particular LLM is suitable (or not) for your specific use case.

About TrialTwin™

TrialTwin™ is developed by:

Data Santander, SL
San Fernando 16, 6C
39010 Santander, Cantabria
Spain

https://TrialTwin.com

Data Santander is a small software development company that provides outside, unbiased suggestions and innovative approaches to tackle data management challenges for our Life Sciences clients.

Compared with larger vendors, at Data Santander we are:
* ***fast*** – clients work directly with our developers to reduce cycle times
* ***fearless*** – we want to eliminate, rather than improve, most processes in clinical data management
* ***flexible*** – we're able to make progress while requirements are defined on the fly
* ***friendly*** – we love what we do, and we bring our quirky personalities to the job
* ***fun*** – life's too short so we make working with us fun and enjoyable

Contact:

José C. Lacal, CTO
Jose.Lacal@DataSDR.com
+34 (674) 88 17 52