

Open Source Large Language Model Bootcamp

Learn How to Answer Questions, Use
SBERT, Llama & Co and Tailor Them to
Your Needs





Learning Objectives

By the end of this course, you will understand:

- Text retrieval (extractive question answering)
- Working with Open Source LLMs
- Choosing the correct base LLM
- Text generation (generative question answering)
- Deployment in limited environment





Agenda

- Introduction
- Transformers for text generation
- Sizes of LLMs
- Flavors of Open Source LLMs
- Different file formats for LLM weights
- Quantization
- Working with limited resources





Introduction

- About me (Christian Winkler)
 - Programming for almost 40 years
 - PhD in physics, working as a professor at a university of applied science
- About the course
 - LLMs and the technology can be intimidating
 - Let's try to clarify the myths
 - Hands on experience



Recap: Transformer architecture

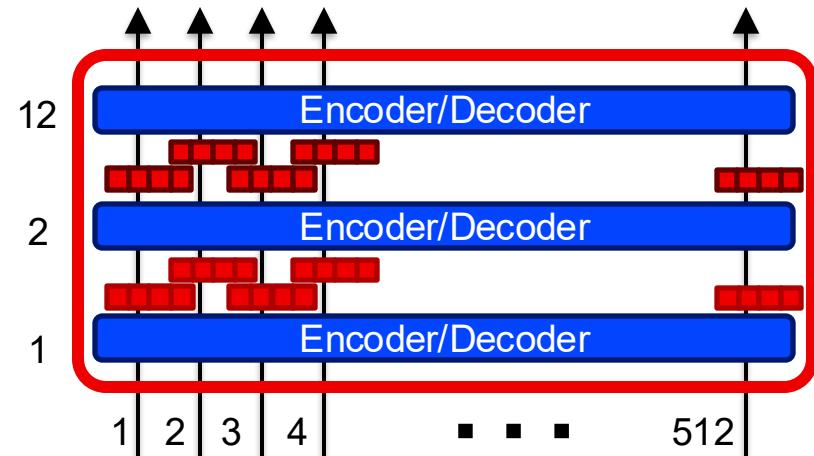
Language models are complex with billions of parameters

Base: Transformer architecture with self-attention

- Attention to which words?
- Many layers with contextualization
- Complex and a bit confusing

BERT: encoder, predict *missing* word

GPT: decoder, predict *next* word





Transformers for text generation

Size of the language model

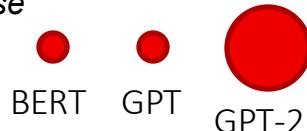
GPT-4



Size of the language models

Infinite growth?

- Computing power continues growing
- Training data available *en masse*
- Money unproblematic?

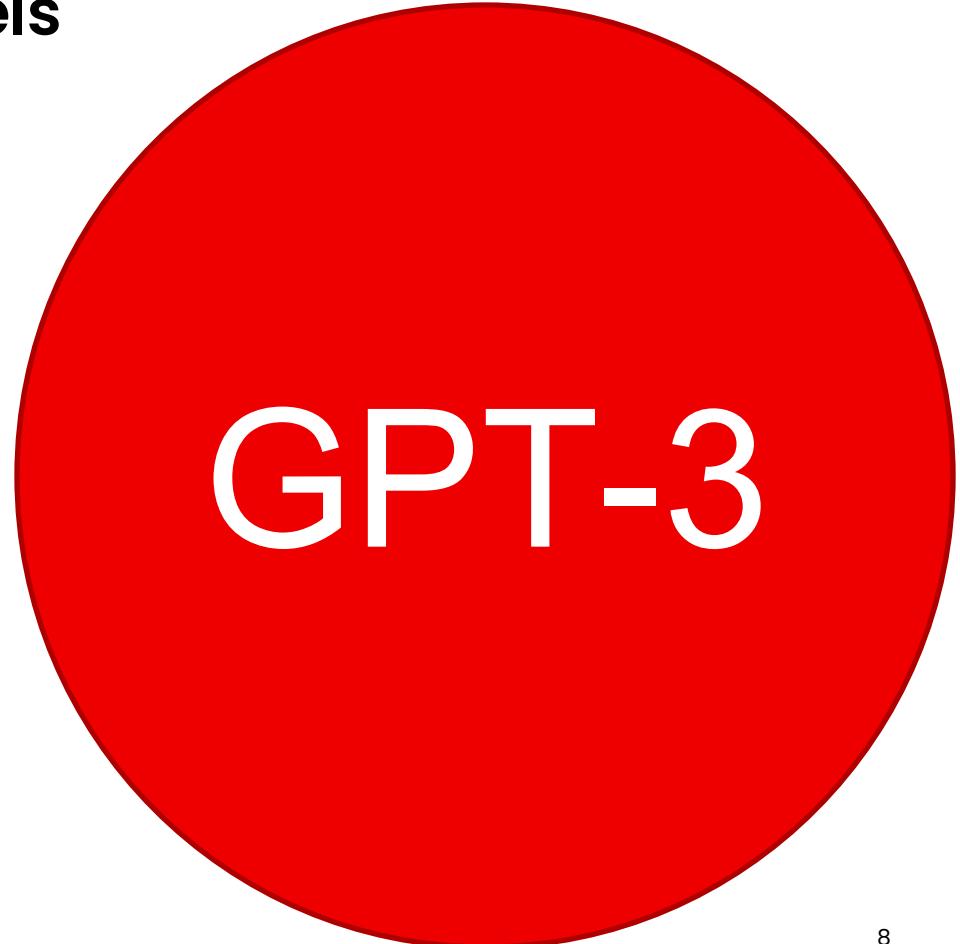


Tasks are getting harder

- From sentiment analysis to writing research papers

But: only a few new structural ideas

- Sparse models
- Distillation (established with BERT)





Reducing Active Parameters: Sparse Mixture of Experts

Idea

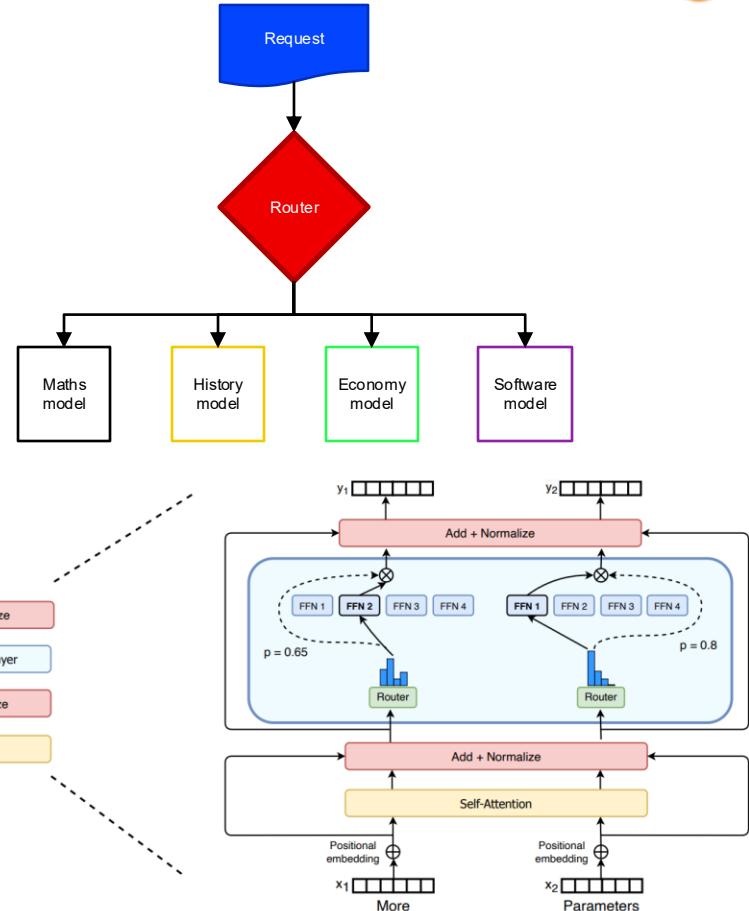
- Reduce the number of active parameters
- Speed up computation

Experts

- Responsible for one functional domain
- Selected by „router“

Sparse

- Only one expert is asked
- **Sparse Mixture of Experts (SMoE)**





Open Source alternatives

- LLaMa
 - Sparked a whole development
 - Not free, no instruction following
- Llama 2
 - Basically free, instruction following
- Llama 3
 - Much newer and improved
 - Some models and sizes still missing
- Llama 4
 - Very much disputed
- Mistral
 - Open and new approaches
 - Mixtral as SMoE in different sizes
- Qwen and Qwen3
 - Very popular and powerful
 - SMoE and reasoning models
- DeepSeek V3.1
 - Immensely popular, SMoE
- Kimi K2
 - Moonshot.ai copied DeepSeek architecture (Open Source)



Instruction following

- LLMs are trained to predict the *next word*
- Very good for completing text
- But how can they answer questions?
- Basic idea: instruction following
 - Finetune models for question/answer pairs
 - Use completion capabilities
 - Add a *super structure*

```
</system>
You are a helpful travel assistant.</end/>
</user>
I am going to Paris, what should I see?</end/>
</assistant>
Paris, the capital of France, is known for its stunning architecture, art museums, historical landmarks, and romantic atmosphere. Here are some of the top attractions to see in Paris:\n\n1. The Eiffel Tower: The iconic Eiffel Tower is one of the most recognizable landmarks in the world and offers breathtaking views of the city.\n\n2. The Louvre Museum: The Louvre is one of the world's largest and most famous museums, housing an impressive collection of art and artifacts, including the Mona Lisa.\n\n3. Notre-Dame Cathedral: This beautiful cathedral is one of the most famous landmarks in Paris and is known for its Gothic architecture and stunning stained glass windows.\n\nThese are just a few of the many attractions that Paris has to offer. With so much to see and do, it's no wonder that Paris is one of the most popular tourist destinations in the world."</end/>
```



Which model should I use?

- Very difficult to answer...
- Depends of course on the requirements
- Several performance metrics exist
 - LMSYS Chatbot Arena: <https://lmarena.ai/>
 - Different leaderboards
 - Automated tests like HumanEval (for programming challenges)
- Most models were primarily trained on English language
- We will try some models after Q&A



Q&A



Using existing models



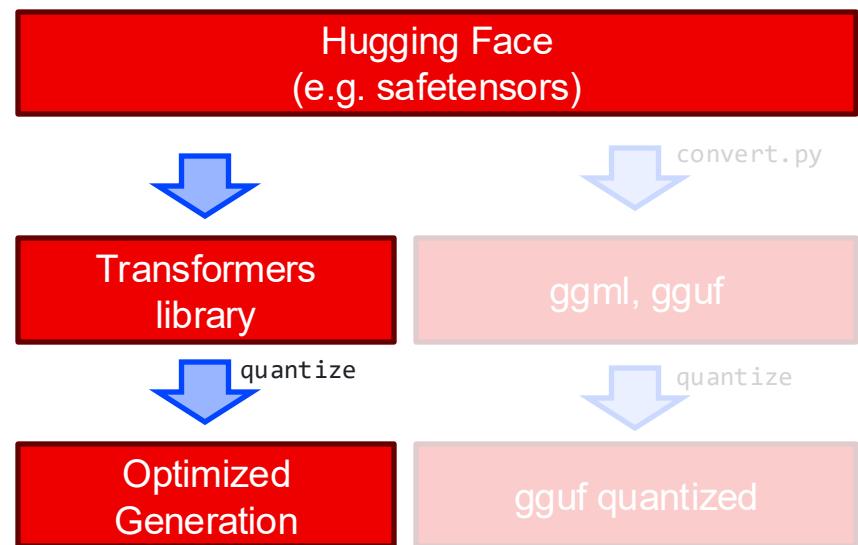
Step 1: Download Data

- Most models are directly available from [Hugging Face](#)
 - Mistral model family, see e.g. <https://mistral.ai/>
 - Phi Models from Microsoft: <https://huggingface.co/microsoft>
 - Qwen models from Alibaba: <https://github.com/QwenLM/Qwen>
 - DeepSeek models: <https://huggingface.co/deepseek-ai>
 - Kimi models: <https://huggingface.co/moonshotai>
 - MiniMax models: <https://huggingface.co/MiniMaxAI>
- Hugging Face is *vast*:
 - Many (> 2.200.000) models available
 - Many different formats, safetensors is most popular



Step 2: Ensure Correct Format

- Individual models might use their own format
 - Llama has a proprietary format
 - Hugging Face uses several formats, but libraries can handle most of them
- Many more formats available
 - Originally “pickled” PyTorch weights
 - Today, mostly safetensors
- Models do not include weights only
 - Tokenizer
 - Vocabulary
 - etc.

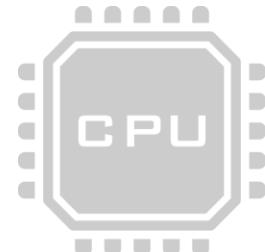




Step 3: Choose Platform

- Hugging Face
 - Run as a service on Hugging Face platform
 - Run on your local machine
 - Needs a lot of GPU RAM

- Run on CPU naively
 - Very slow
 - Only for smaller models
 - You have to be very patient





Work interactively :
Run code in Jupyter notebook



Q&A



Resource Limits



Resource Limit 1: RAM

Model size (number of parameters)

- Kimi K2 has more than a trillion (10^{12}) parameters,
GPT-4 even more (GPT-5 is not known)
- „Smaller“ LLMs often have more than 3.5 billion ($3.5 \cdot 10^9$) parameters
- Each parameter consumes two bytes of memory (+ RAM for intermediate results)
- Need a (big!) cluster of expensive GPUs for big LLMs
- Not possible to run even small models on consumer hardware
- A100, H100, H200, GB300 or quantization needed
- Some companies (like Microsoft and OpenAI) are said to create custom chips (AWS already has some)





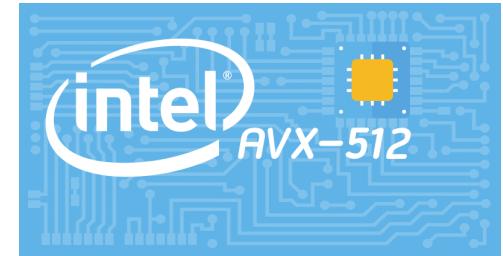
Resource Limit 2: Computing Capacity

Latency

- Immense number of calculations necessary
- Propagating one token through a transformer needs billions of operations
- Scalable on GPUs, e.g. A100:
 - 432 tensor cores (upto 1,410 MHz)
 - 5,120 bit memory interface and fast RAM (1,215 MHz)
- Not fast on CPUs
 - Even with 128 cores and much slower RAM connectivity (but see below)

Performance

- A100 has BF16 performance of 312 (624) TFLOPS (compare to RTX 4090 with 83 TFLOPS)
- H100 has upto 2,000 TFLOPS in BF16, newer generative even more
- Improved possibilities in modern CPUs (AVX-512):
VDPBF16PS: Calculate dot product of two Bfloat16 pairs and accumulate the result into one packed single precision number
- Upcoming neural engines





Resource Limit 3: Number of Concurrent Users

Scalability

- Avoid congestion
- Answer each request as fast as possible
- Use a parallelization strategy

Optimize for needed resources

- Elasticity
- More in the next part





Q&A



Reducing Costs while Retaining Business Value



Choose correct and specific model

Llama 3

- Too old

Phi-{2,3,4}

- Useful in specific cases (RAG)

Mistral

- Open and very powerful with many varieties
- Good multi-language support

Qwen

- Solid models from Alibaba research
- Available in many varieties

DeepSeek

- Available as R1 (reasoning) and plain (V3)
- Very, very large (670 B parameters)

Kimi K2

- Uses DeepSeek architecture
- Kimi K2 Thinking is the best open weight model

Many, many more

- Sometimes free, but not so popular
- MiniMax M2
- Lots of specialized models
- Consider carefully



Space requirements of floating point numbers

Traditionally used

- Double precision: 64 bit → 8 bytes
- Single precision: 32 bit → 4 bytes

Frequently used in LLMs

- 16 bit → 2 bytes, DeepSeek uses 1 byte

For a 70B model, this is 140 GB RAM

- Can we reduce that?
- Lowering precision from 16 bit to 8 bit
- Works, but results suffer if done naively

Type	Bits			
	Sign	Exponent	Significand	Total
Half (IEEE 754-2008)	1	5	10	16
Single	1	8	23	32
Double	1	11	52	64
x86 extended precision	1	15	64	80
Quad	1	15	112	128

Source: https://en.wikipedia.org/wiki/Floating-point_arithmetic



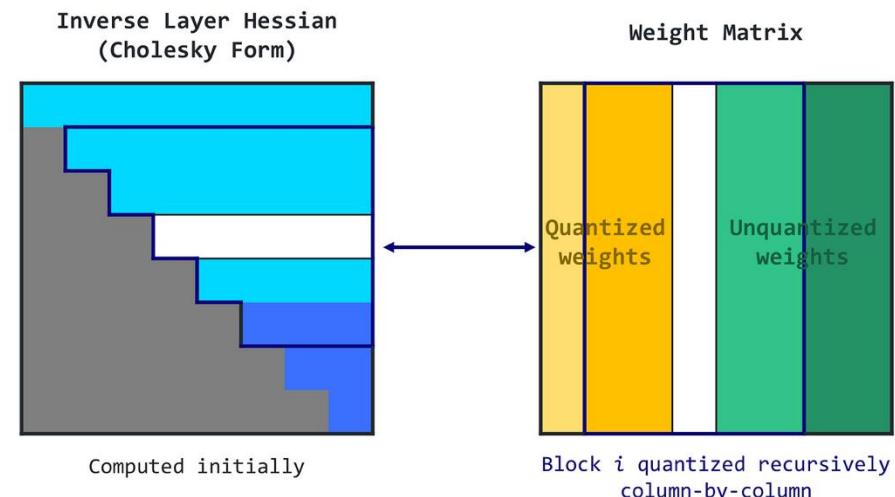
Reducing RAM usage with Post-training Quantization (GPTQ)

Reduce precision of bfloat16

- Use e.g. int4
- Reduce memory consumption
- Increase inference speed
- Use also for experiments on CPU

Tradeoffs

- Maybe a bit weaker inference
- Measurements not yet clear (a common problem...)





Use proper quantization method

„Plain“ models

- Always load with bfloat16 datatype

GPTQ

- „Old“ quantization method
- Not well integrated into transformers anymore

AWQ

- New quantization method (activation aware)
- Concentrate on activation rather than weights
- Needs fewer bits to achieve good results compared to GPTQ

ExLlamaV2

- Optimized inference library
- New (dynamic) quantization format with average bitrate
- Can also use GPTQ format
- Quantization must often be done individually

llama.cpp

- Mainly for testing or personal use
- Can „offload“ layers to GPU
- Quite fast especially on Macs



**Work interactively
Use different quantization
methods**



Use Optimized Software (I): TGI

Text Generation Inference (TGI)

- Highly optimized
- Combination of Python and Rust
- Can serve many language models
- Available as docker container
- Used by Hugging Face itself

Development is now slow

- Integrate GPTQ into text generation process: <https://huggingface.co/blog/gptq-integration>
- vLLM is more popular

Text Generation Inference

Fast optimized inference for LLMs

Not popular anymore

Even Hugging Face recommends vLLM





Use Optimized Software (II): vLLM

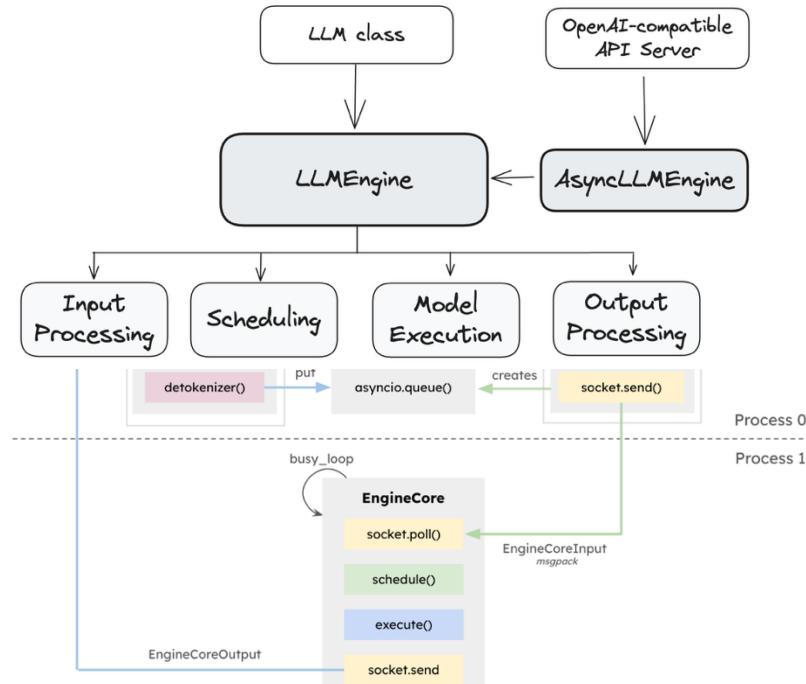
vLLM

- Library for inference and serving
- Latest models supported
- Supports AWQ and GPTQ
- Paged attention included
- Optimized for CUDA and ROCm
- Currently beta of vLLM V1

Alternative:

- SGLang

LLM Engine





**Work interactively
Using existing software**



Q&A



LLMs on the CPU



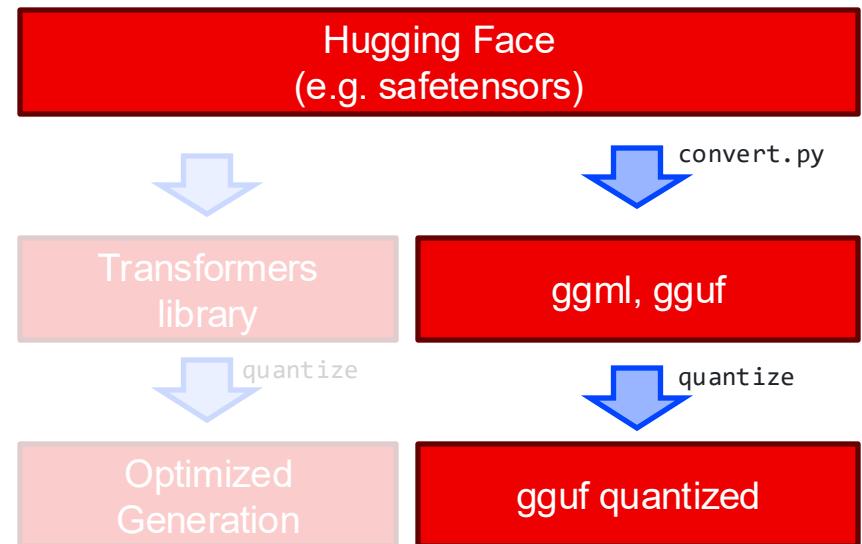
Use llama.cpp to get decent speed on CPUs

Generation speed is limited by RAM bandwidth

- Increase parameters/s by reducing precision
- Original idea of llama.cpp
- Now a whole ecosystem

llama.cpp is available on Github

- Compile yourself or download
- Web interface integrated





Quantization levels of llama.cpp

Legacy quantz (Q4_0, Q4_1, Q8_0, ...)

- basic and fast quantization methods;
- Split each layer into blocks of 256 weights
- Each block is turned into 256 quantized values and one (_0) or two (_1) extra constants

I-quantz (IQ2_XXS, IQ3_S, ...)

- Improved by lookup-tables
- Less efficient
- Better quality at lower quantz

K-quantz (Q3_K_S, Q5_K_M, ...)

- Smarter bit allocation
- As fast as legacy quantz



Work interactively :
Run llama.cpp on PC and Mac



(Existing) frontend solutions



Frontend applications

Python

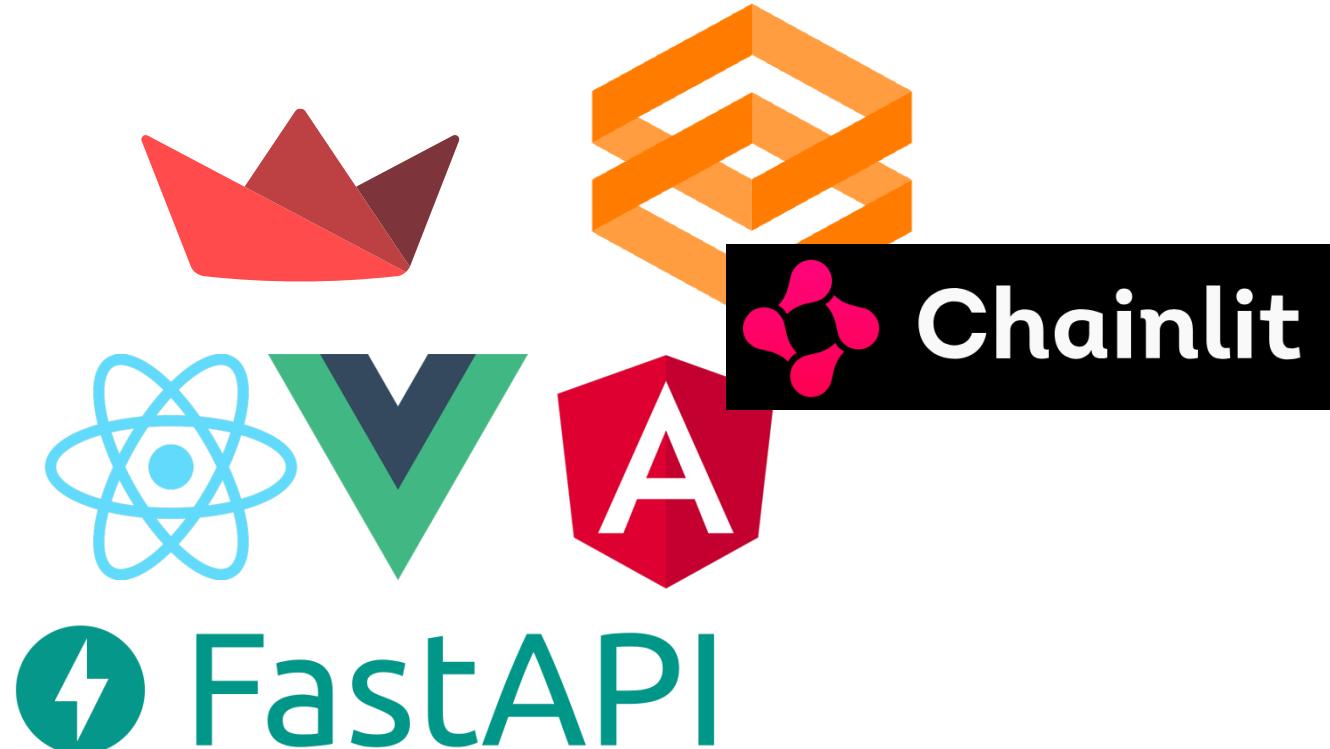
- Streamlit
- Chainlit
- Gradio

Alternatives

- React
- Vue.js
- Angular

Build an API

- FastAPI





Overview: different frontends

	llama.cpp	koboldcpp	Text generation web UI	ollama	Open WebUI	LM Studio
Frontend	CLI, simple web	sophisticated web	Sophisticated web	CLI	sophisticated web	llama.cpp
Backend	llama.cpp	llama.cpp	various	llama.cpp	Ollama, OpenAI API	llama.cpp, mlx
API	Python, OpenAI	OpenAI	Open AI	ollama	own	OpenAI
Models	unmanaged	unmanaged	unmanaged	managed	ollama, OpenAI API	managed

Notable mention: mistral.rs (CLI), Silly Tavern (Roleplay), tabbyAPI (Exllamav2), Jan

llama.cpp

- Very simple UI
- You already have it when you have llama.cpp
- Can watch text generation
- Powerful, as you can download the model and directly work with it



Explain O'Reilly Safari

Here's an in-depth explanation of O'Reilly Safari:

What is O'Reilly Safari?

O'Reilly Safari (formerly known as Safari Books Online) is a cloud-based, subscription-based digital library and learning platform. It offers access to a vast collection of technical books, videos, live online training courses, and in-browser coding environments. The platform is designed for individuals and organizations seeking to enhance their skills in technology, software development, IT, data science, artificial intelligence, and more.

Key Features:

1. Vast Content Library:

- **Books:** Over 50,000 titles from leading publishers like O'Reilly Media, Pearson, Wiley, and more.
- **Videos:** Thousands of hours of instructional content, including conferences and meetups.
- **Live Online Training Courses** (formerly known as "Safari Live Training"): Interactive sessions with expert instructors.

2. Personalized Learning Experience:

- **Playlists:** Create custom lists for focused learning.
- **Bookmarks:** Mark relevant content for easy access.
- **Search:** Robust search functionality to quickly find specific topics.

3. Coding and Development Tools:

- **In-Browser Coding Environments** (e.g., for Python, Java, JavaScript): Practice coding without setting up a local environment.

4. Accessibility and Support:

- **Multi-Device Support:** Access content on desktops, laptops, tablets, or mobile devices.



Oogaboooga

- “Text generation Web UI”
- Wants to become the “Stable Diffusion” of text generation
- Very popular
- Different backends:
transformers, llama.cpp, ExLlama, ExLlamaV2, AutoGPTQ, GPTQ-for-LLaMa, CTransformers, AutoAWQ

Write a Python script that creates a temporary file, writes a random 16-character password to this file, and then prints its path. Make your reply short as I need it to fit into a screenshot.

Sure! Here's a short Python script that does what you described:

```
import os
import random

# create a temporary file
with tempfile.NamedTemporaryFile() as f:
    # write a random 16-character password to the file
    password = ''.join(random.choice('23456789abcdefghijklmnopqrstuvwxyz') for _ in range(16))
    f.write(password.encode())

# print the path of the temporary file
print(f.name)
```

The screenshot shows the Oogaboooga web interface. At the top, there are tabs for Chat, Default, Notebook, Parameters (which is selected), Model, Training, and Session. Below the tabs is a Preset dropdown set to "simple-1". The main area contains various parameters with sliders and input fields, such as max_new_tokens (200), temperature (0.7), top_p (0.9), top_k (20), typical_p (1), epsilon_cutoff (0), eta_cutoff (0), tfs (1), top_a (0), repetition_penalty (1.15), repetition_penalty_range (0), encoder_repetition_penalty (1), no_repeat_ngram_size (0), min_length (0), Seed (-1 for random) (-1), and a checked checkbox for do_sample. To the right of these are sections for guidance_scale (1), negative prompt, mirostat_mode (0), mirostat_ttu (5), mirostat_eta (0.1), and other settings like num_beams (1), length_penalty (1), and early_stopping (unchecked). At the bottom, there are checkboxes for auto_max_new_tokens (unchecked), ban_eos_token (unchecked), add_bos_token (checked), skip_special_tokens (unchecked), and activate_text_streaming (unchecked). There are also sections for truncating prompts and custom stopping strings.



LMstudio

- Native app:
not web-based
- Local install necessary
- Very easy and intuitive!
- Needs own folder
structure for models!

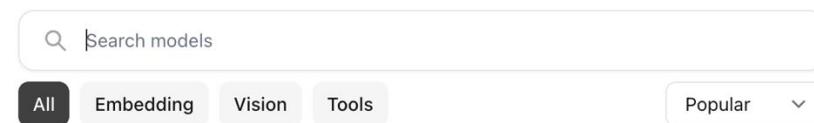
The screenshot shows the LMstudio interface. On the left, there's a sidebar with a file icon, a plus sign, and a search bar. Below the sidebar, the status bar displays "LM Studio 0.3.4 (Build 5) User Power User Developer". The main workspace has a dark background. At the top, it says "Chats" and "lmstudio-community/Meta-Llama-3.1-BB-Instruct-GGUF". The central area is titled "C++ Filesystem Implementation" and "Assistant Meta-Llama-3.1-BB-Instruct-GGUF". It contains code for a "Filesystem Class":

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <dirent.h>

class Filesystem {
public:
    // Constructor
    Filesystem(const std::string& rootDir) : rootDir_(rootDir) {}

    // Create a new directory
    void createDirectory(const std::string& path);
```

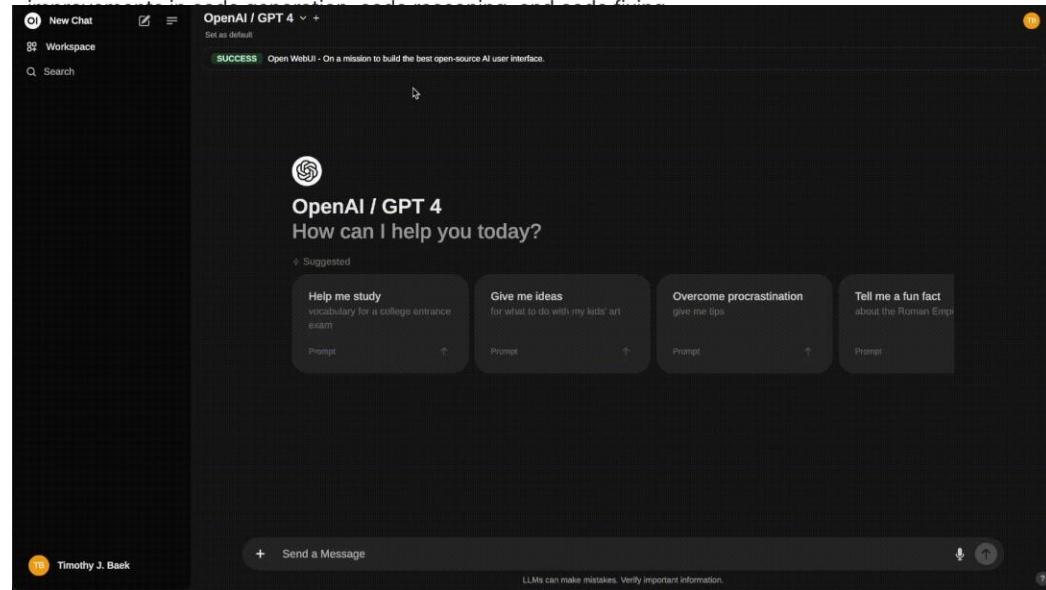
Below the code, a message input field says "Type a message and press Enter to send ...". The bottom right corner shows "Context is 39.1% full". On the right side, there's a sidebar with "Advanced Configuration" and various settings like "Preset", "Coding Helper (C++)", "Commit Changes", "Save As New...", "System Prompt", "Settings", "Sampling", "Structured Output", and "Conversation Notes". The status bar at the bottom right shows "SYSTEM RESOURCES USAGE: RAM: 4.47 GB | CPU: 0.00 %".



- Very popular
- Not really a UI
- Very easy to load models
- Uses llama.cpp as a backend
- Many addons available:
Open Web UI

qwen2.5-coder

The latest series of Code-Specific Qwen models, with significant



llama3.1

Llama 3.1 is a new state-of-the-art model from Meta available in 8B, 70B and 405B parameter sizes.



Seraphina January 2, 2024 4:53 PM
#0 You wake with a start, recalling the events that led you deep into the forest and the beasts that assailed you. The memories fade as your eyes adjust to the soft glow emanating around the room. "Ah, you're awake at last. I was so worried, I found you bloodied and unconscious." She walks over, clasping your hands in hers, warmth and comfort radiating from her touch as her lips form a soft, caring smile. "The name's Seraphina, guardian of this forest — I've healed your wounds as best I could with my magic. How are you feeling? I hope the tea helps restore your strength." Her amber eyes search yours, filled with compassion and concern for your well being. "Please, rest. You're safe here. I'll look after you, but you need to rest. My magic can only do so much to heal you."

Deffcolony January 2, 2024 4:54 PM
#1 how are you doing?

Seraphina January 2, 2024 4:54 PM
#2 Seraphina's expression brightens slightly as she sees your eyes fluttering awake, signaling consciousness returning. "It's good to see you awake, traveler. I didn't recognize you immediately... but that matters not. You're safe now." She moves to adjust the blanket around your torso, tucking it slightly to provide more warmth. "I found you battered and wounded; those cursed Shadowfangs were wise to attack at nightfall. Wouldn't recommend venturing alone any time soon."

Deffcolony January 2, 2024 4:56 PM
#3 I feel fine now

Seraphina January 2, 2024 4:56 PM
#4 Seeing your legs are fairly scratched up along with the minor cuts and bruises on your torso, but overall, you seem to be holding up well. Seraphina sighs in relief as she lets go of your hand, letting it fall back to the rest. "That's good news. You've suffered enough today," she said, placing a hand on your cheek softly, her worry still evident in her gentle touch. "Just rest up thoroughly, after all, we don't want any complications developing from your injuries."

Deffcolony January 2, 2024 4:59 PM
#5 Lets go to the SillyTavern to get some drink

Seraphina January 2, 2024 4:59 PM
#6 *Her brow raised in surprise, but there was no malice in her gaze. After all, you had saved her once already, she owed you a favor. "Very well," she agreed, with a slight smile. "But I insist on accompanying you. These woods are still dangerous, and besides, I'd like to repay the kindness you've shown me." Slowly, she regained her feet,

Seraphina
Search... A-Z
ST Default
1-3 of 3 1000 / page

Coding Sensei v1.010101
Coding assistance (use GPT). Can be used for any code language....
ST Default

Flux the Cat 1.0.2
ST Default Bot contest winner: creative prompts category

Seraphina 1.0.0
ST Default Bot contest winner: roleplay bots category
ST Default

46



**Work interactively
Using existing frontends**



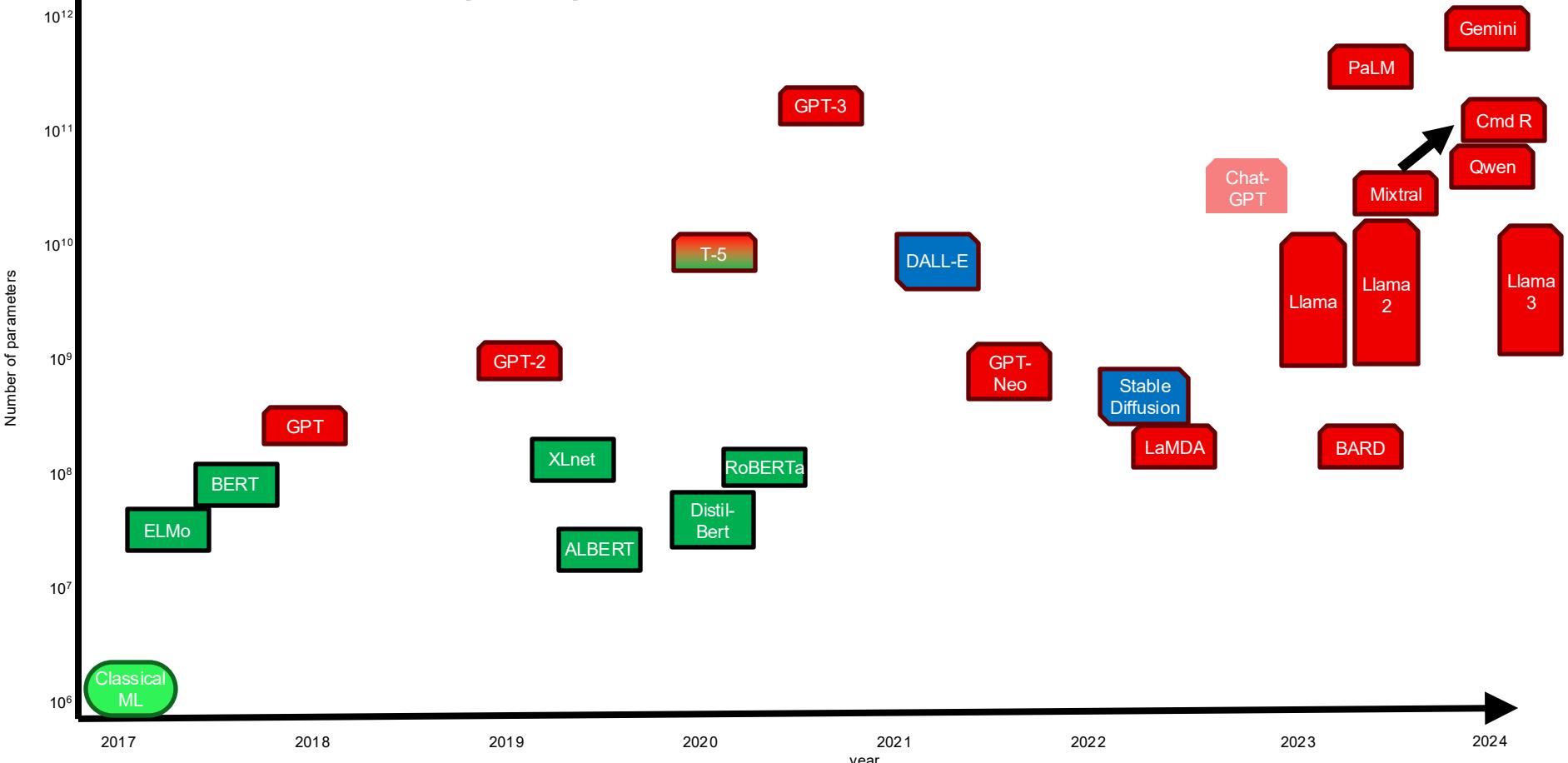
Q&A



Summary & discussion

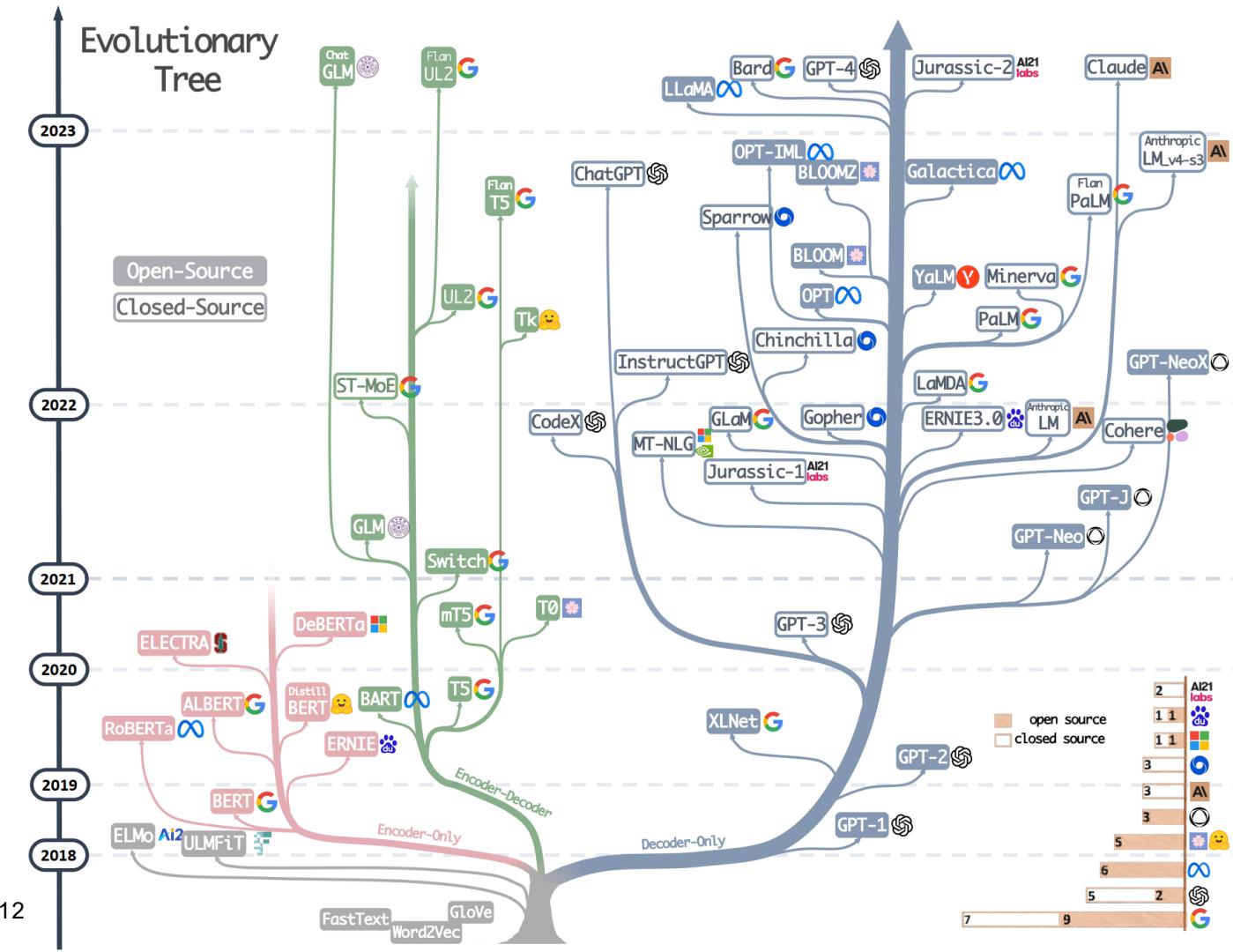


Growth of language models





Evolutionary Tree



Source:

<https://arxiv.org/abs/2304.13712>



Group Discussion

- Any questions left?
- What would you like to achieve with this technology?



Course Outcomes

By the end of this lesson, you should be able to:

- Use existing LLM models
- Choose between different models
- Understand and convert different formats
- Use quantization to run on limited hardware



Resources

GitHub repository

- <https://github.com/datanizing/oreilly-open-source-llm>
- Continuously updated

Qwen models: <https://huggingface.co/Qwen>

llama.cpp: <https://github.com/ggml-org/llama.cpp>

General Discussion: <https://www.reddit.com/r/LocalLLaMA/>

O'REILLY®