

Intelligente Informationssysteme

Conversational AI

Dominik Neumann

Bock 2 – Conversational AI

Design Patterns

Conversational AI

Simulate a conversation with the feeling of having a conversation with a human.

- conversational memory
- dialogue generation
- Examples: ChatGPT, Anthropic Claude, Perplexity.AI

Retrieval Augmented Generation

Knowledge Retrieval and understanding is key

- Access to contextual data
- Retrieval and augmentation strategies

CoPilot

Assists a human in his work.

Key differentiator for becoming a CoPilot is understanding of the environment in which the user works.

- access to tools and data,
- reasoning and planning capabilities,
- and specialized profiles
- Examples: GitHub CoPilot

Multi Agent Problem Solver

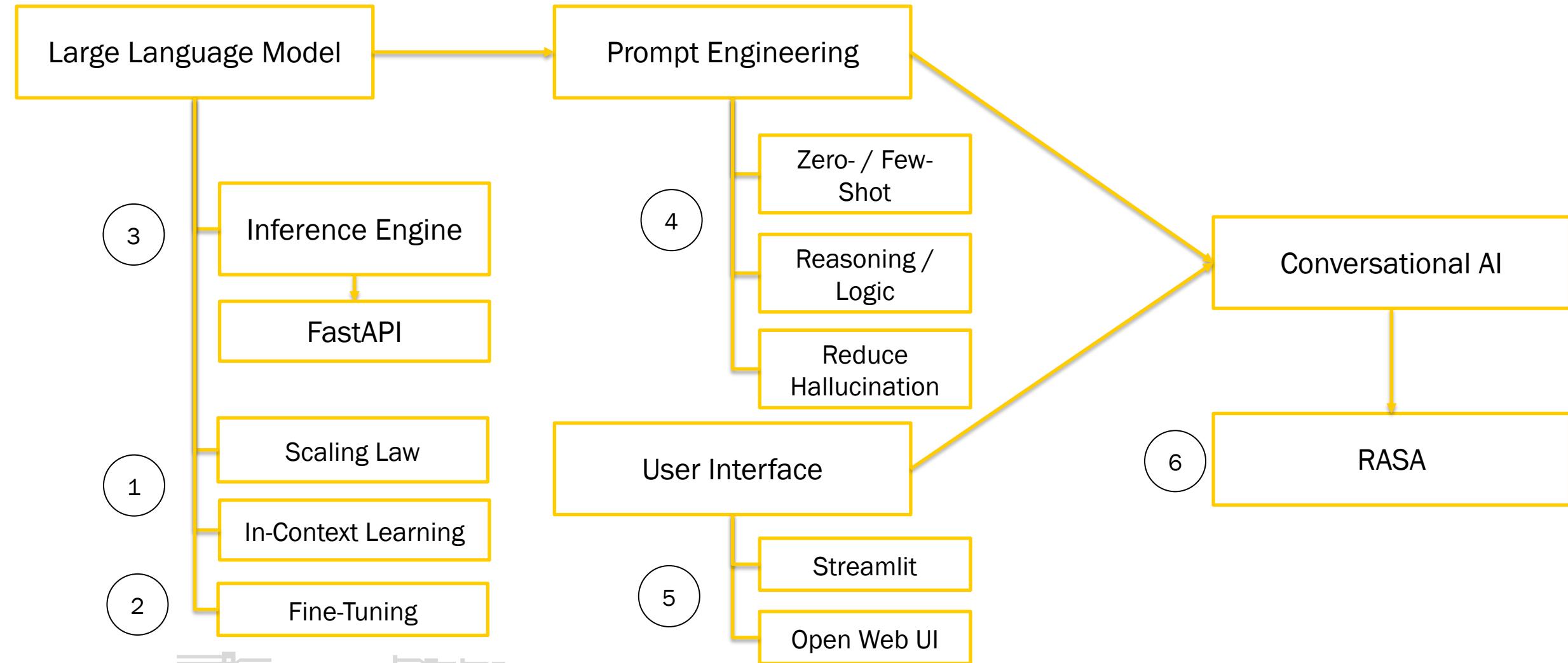
Multiple agents collaborate to solve a problem. Each agent

- has access to its own set of tools and
- can assume a very specific role while reasoning
- is planning (and executing) its actions.
- Example: Devin (<https://preview.devin.ai>)



Simulate a conversation with the feeling of having a conversation with a human.

Bock 2 – Conversational AI

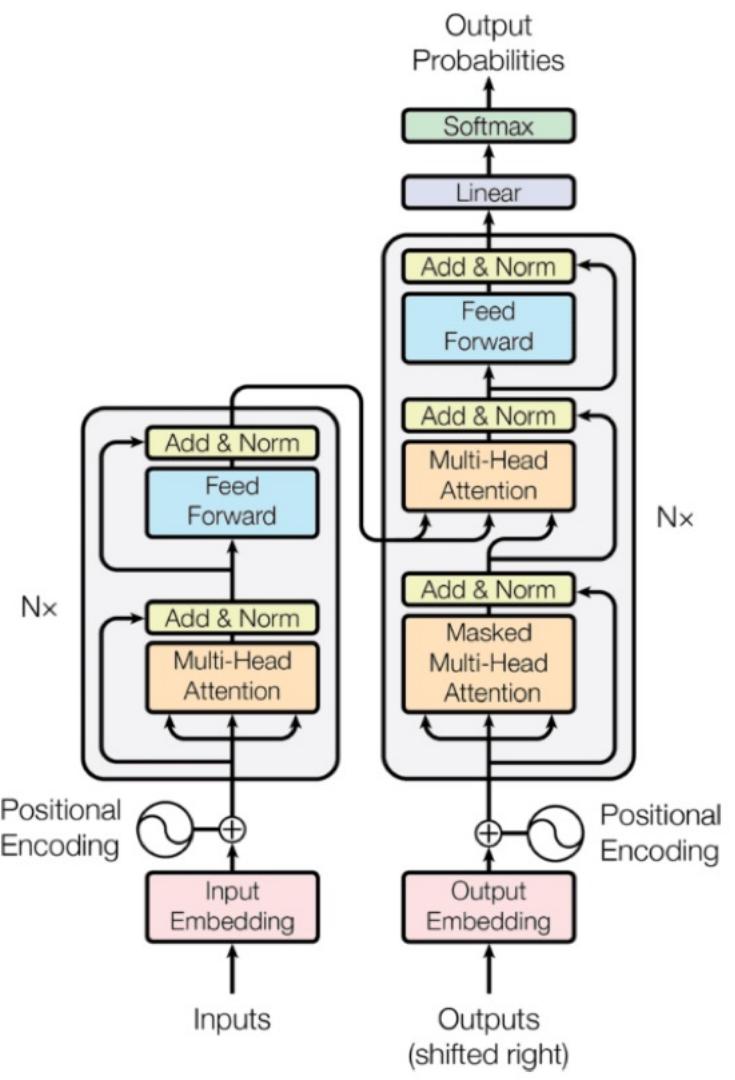


01

Scaling Laws Emerging Probabilities In-Context Learning

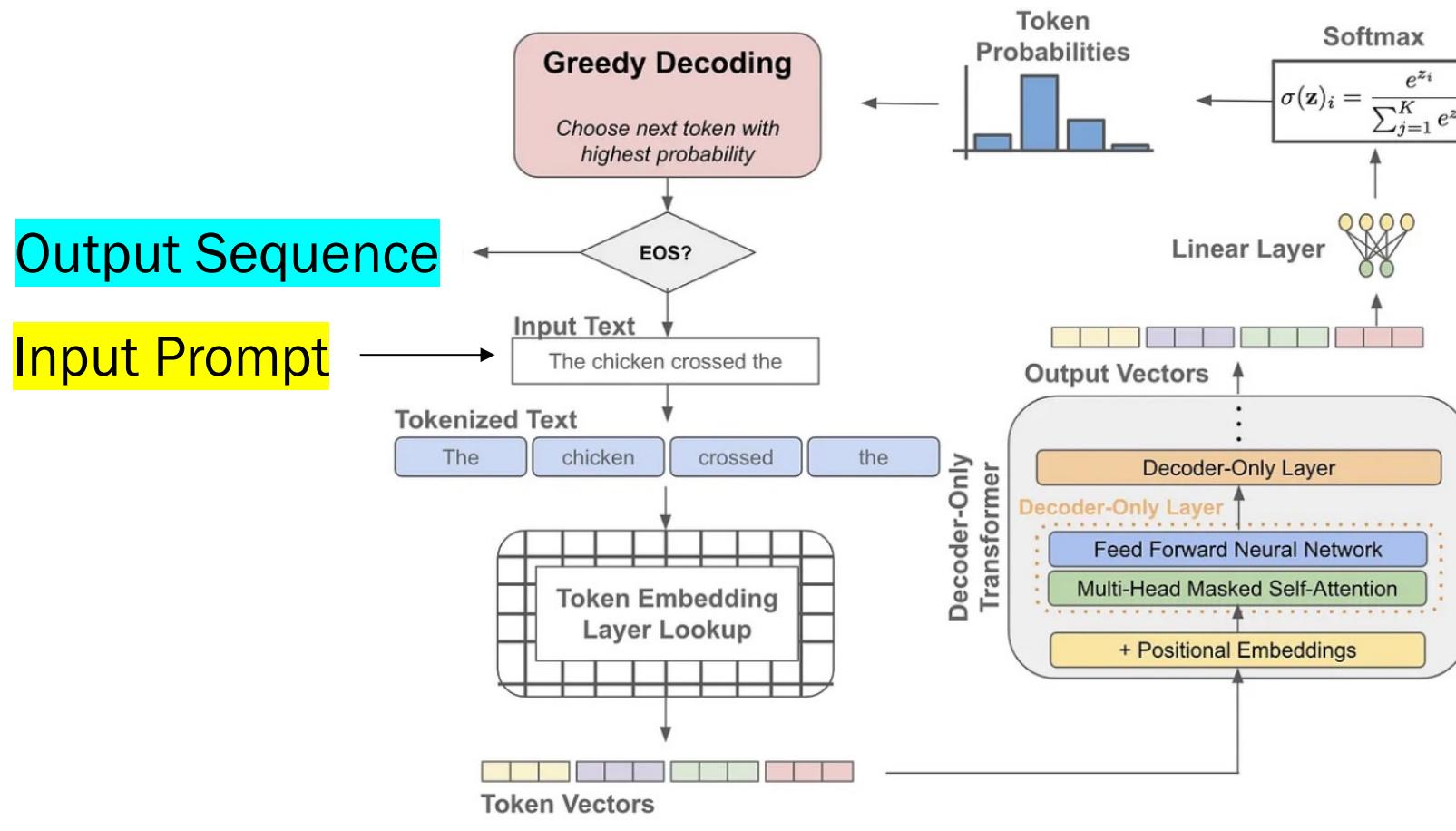


Recap: Large Language Models



The heart of Transformer is the (self)-attention mechanism, which can capture long-term contextual information much more effectively using GPUs than the recurrence and convolution mechanisms.

Recap: Large Language Models



Most LLMs: Decoder-only architectures

See: https://medium.com/@amansinghalmi_33304/temperature-langs-b41d75870510

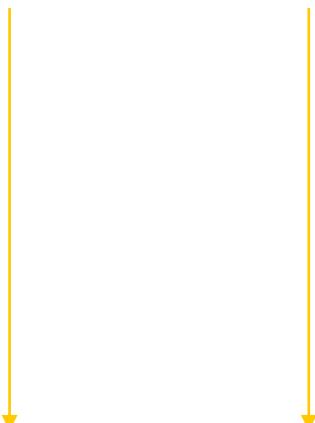


The large number of parameters (of the model)
makes a language model a large language model.

Understand Memory Consumption

Memory Needed for Inference

- 1 Parameter = 4 bytes (32-bit float)



- 1B parameters = 4×10^9 bytes = 4GB

Memory Needed for Training

	Bytes per parameter
Model Parameters	4 bytes
Adam optimizer (2 states)	+ 8 bytes
Gradients	+ 4 bytes
Activations & temporary memory	+ 8 bytes
TOTAL	24 bytes

- 1B parameters = 24×10^9 bytes = 24GB



Memory needed to train larger models

1B param model

24 GB



175B param model
(like GPT 3)

$175 * 24 = 4,200$ GB
@32-bit full precision

500B param model
(like PaLM)

$500 * 24 = 12,000$ GB
@32-bit full precision

Scaling Laws

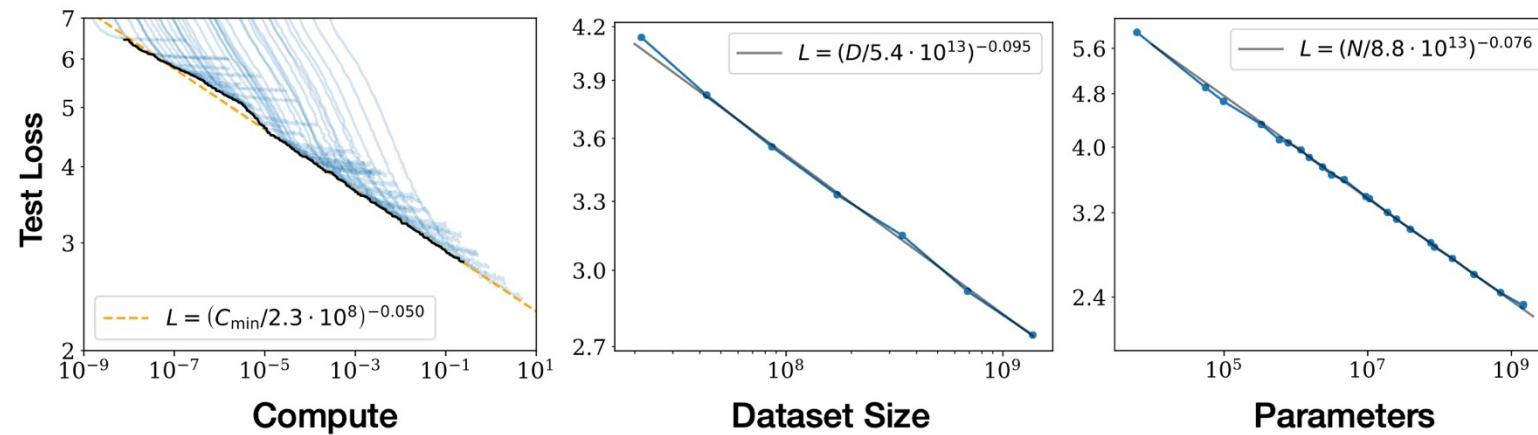
Pre-Training Costs

Model	Parameter Size	Data Scale	GPUs Cost	Training Time
GPT-3 (Brown et al., 2020)	175B	300B tokens	-	-
GPT-NeoX-20B (Black et al., 2022)	20B	825GB corpus	96 A100-40G	-
OPT (Zhang et al., 2022a)	175B	180B tokens	992 A100-80G	-
BLOOM (Scao et al., 2023)	176B	366B tokens	384 A100-80G	105 days
GLM (Zeng et al., 2023)	130B	400B tokens	786 A100-40G	60 days
LLaMA (Touvron et al., 2023a)	65B	1.4T tokens	2048 A100-80G	21 days
LLaMA-2 (Touvron et al., 2023b)	70B	2T tokens	A100-80G	71,680 GPU days
Gopher (Rae et al., 2022)	280B	300B tokens	1024 A100	13.4 days
LaMDA (Thoppilan et al., 2022)	137B	768B tokens	1024 TPU-v3	57.7 days
GLaM (Du et al., 2022)	1200B	280B tokens	1024 TPU-v4	574 hours
PanGu- α (Zeng et al., 2021)	13B	1.1TB corpus	2048 Ascend 910	-
PanGu- \sum (Ren et al., 2023b)	1085B	329B tokens	512 Ascend 910	100 days
PaLM (Chowdhery et al., 2022)	540B	780B tokens	6144 TPU-v4	-
PaLM-2 (Anil et al., 2023)	-	3.6T tokens	TPUv4	-
WeLM (Su et al., 2023a)	10B	300B tokens	128 A100-40G	24 days
Flan-PaLM (Chung et al., 2022)	540B	-	512 TPU-v4	37 hours
AlexaTM (Soltan et al., 2022)	20B	1.3 tokens	128 A100	120 days
Codegeex (Zheng et al., 2023)	13B	850 tokens	1536 Ascend 910	60 days
MPT-7B (Team, 2023)	7B	1T tokens	-	-

From: Efficient Large Language Models: A Survey <https://openreview.net/pdf?id=bsCCJHb08A>

Model Performance depends strongly on scale, weakly on model shape (architecture).
Scale consists of three factors:

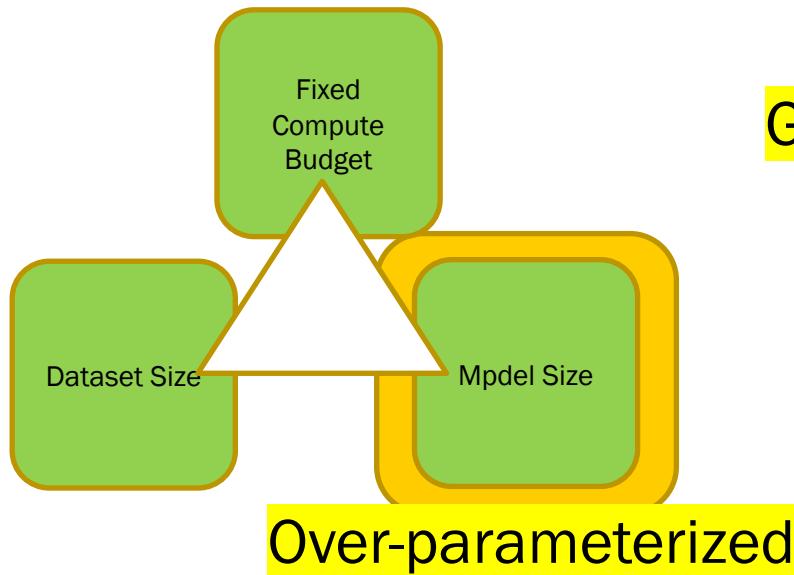
1. Number of model parameters
2. Size of dataset (tokens)
3. Amount of compute used for training (amount of backpropagation steps to adjust the parameters)



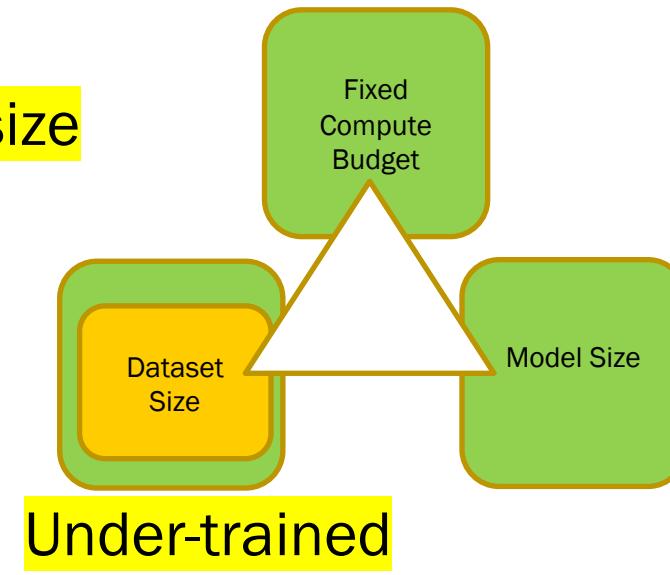
<https://arxiv.org/abs/2001.08361>

The quest for ideal balance between training dataset size, model size, and compute budget continues to shape the development of large language models.

- Very large models may be over-parameterized and under-trained
- Smaller models trained on more data could perform as well as large models



Green: optimal size



„Autoregressive Transformers smoothly improve in performance as model size and compute budgets increase, following a power-law plus constant scaling law.

The optimal model size also depends on the compute budget through a power-law, with exponents that are nearly universal across all data domains.“

The cross-entropy loss has an information theoretic interpretation as $H(\text{True}, \text{Model}) = H(\text{True}) + DKL(\text{True} \parallel \text{Model})$ and the empirical scaling laws suggest a prediction for both the true data distribution's entropy and the KL divergence between the true and model distributions.

<https://arxiv.org/pdf/2010.14701>

Scaling Laws Further Reading

„Autoregressive Transformers smoothly improve in performance as model size and compute budgets increase, following a power-law plus constant scaling law.“

The optimal model size also depends on the compute budget through a power-law, with exponents that are nearly universal across all data domains.“

The cross-entropy loss has an information theoretic interpretation as $H(\text{True}, \text{Model}) = H(\text{True}) + D_{\text{KL}}(\text{True} \parallel \text{Model})$ and the empirical scaling laws suggest a prediction for both the true data distribution's entropy and the KL divergence between the true and model distributions.

<https://arxiv.org/pdf/2010.14701>

Scaling Laws

Further Reading

Log law: $\log(a/b) = \log(a) - \log(b)$

Cross-Entropy $H(P, Q) = - \sum P(x) * \log(Q(x))$

is used in classification tasks where the goal is to minimize the cross entropy between the predicted probability distribution and the true distribution.

$$\begin{aligned} D_{KL}(P || Q) &= \sum P(x) * \log(P(x) / Q(x)) = \sum P(x) * [\log(P(x)) - \log(Q(x))] = \sum P(x) * \log(P(x)) - \sum P(x) * \log(Q(x)) \\ &= \sum P(x) * \log(P(x)) + H(P, Q) \end{aligned}$$

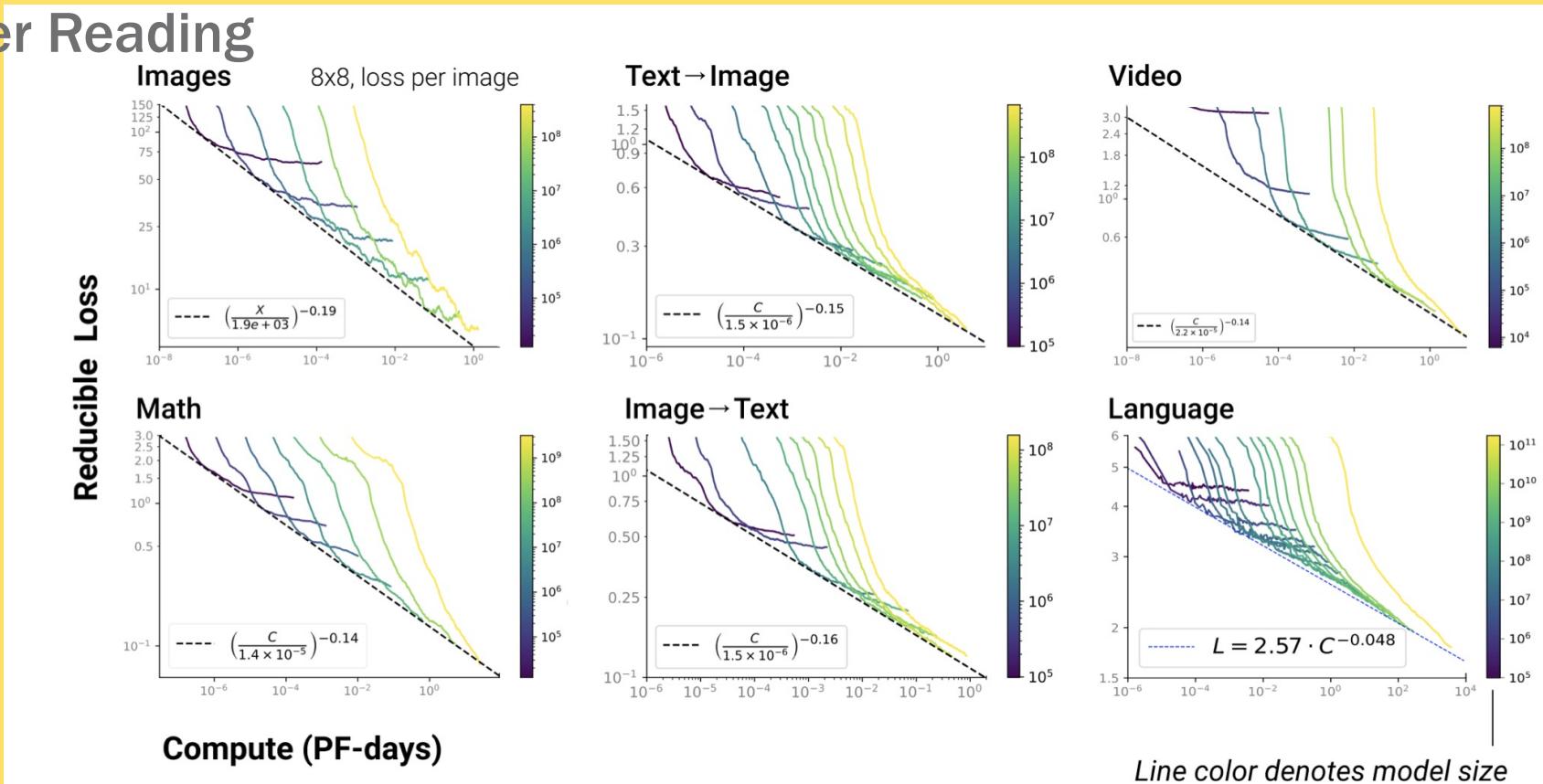
KL divergence $D_{KL}(P || Q)$, also known as relative entropy, measures the difference between two probability distributions P and Q

$$\begin{aligned} \rightarrow H(P, Q) &= - \sum P(x) * \log(P(x)) + D_{KL}(P || Q) \\ \rightarrow H(P, Q) &= H(P) + D_{KL}(P || Q) \end{aligned}$$

<https://medium.com/@mrthinger/kl-divergence-vs-cross-entropy-exploring-the-differences-and-use-cases-3f3dee58c452>

Scaling Laws

Further Reading



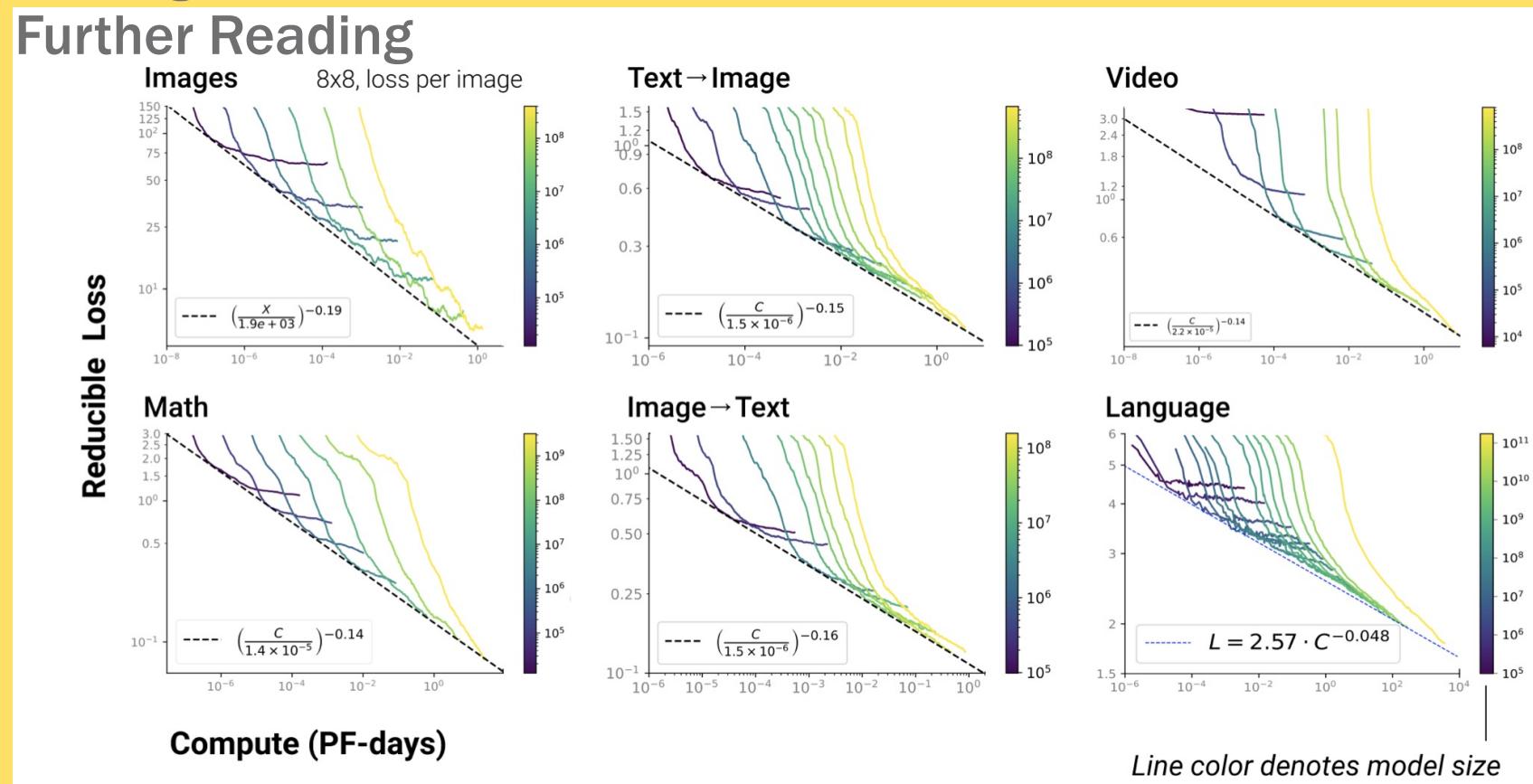
By studying many different model sizes N , compute budgets C , or dataset sizes D , we demonstrate that the scaling relation for the loss

$$L(x) = L_\infty + \left(\frac{x_0}{x}\right)^{\alpha_x} \quad (1.1)$$

<https://arxiv.org/pdf/2010.14701>

Scaling Laws

Further Reading



Information theoretic interpretation of Cross-Entropy: Irreducible Loss estimates the entropy of the true data distribution,

$$\begin{aligned} L_\infty &\approx S(\text{True}) \\ \left(\frac{x_0}{x}\right)^{\alpha_x} &\approx D_{\text{KL}}(\text{True} \parallel \text{Model}) \end{aligned}$$

“Irreducible Loss”
“Reducible Loss” (1.2)

Reducible Loss is an estimate of the KL divergence between the true and model distributions

<https://arxiv.org/pdf/2010.14701>

As the models grow **emergent abilities** arise.

- “**Emergence is when quantitative changes in a system result in qualitative changes in behavior.**” (rooted back to a 1972 essay called “More Is Different” by Nobel prize-winning physicist Philip Anderson)

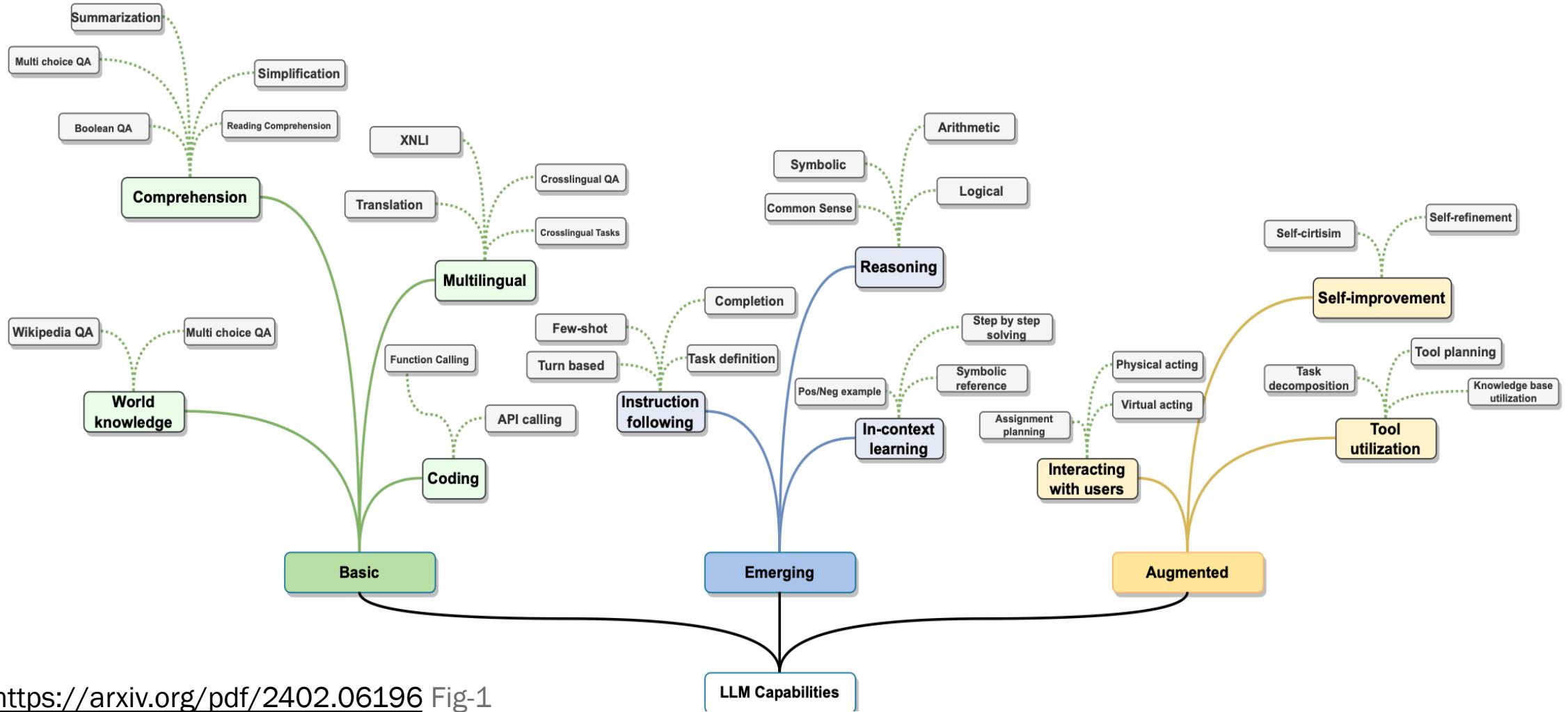
Large language models can exhibit unpredictable ("emergent") jumps in capability as they are scaled up. Emergence refers to the capabilities of LLMs that appear suddenly and unpredictably as model size, computational power, and training data scale up.

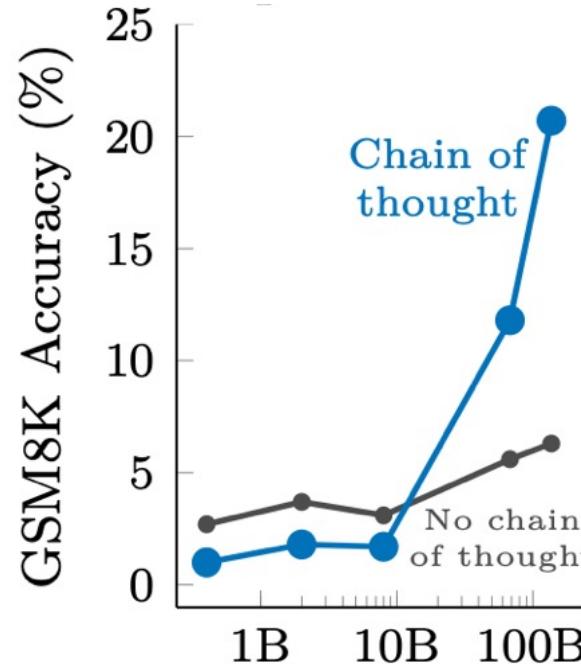
- “**An ability is emergent if it is not present in smaller models but is present in larger models.**” (definition from Emergent Abilities of Large Language Models in <https://arxiv.org/abs/2206.07682>)

<https://cset.georgetown.edu/article/emergent-abilities-in-large-language-models-an-explainer>

Scaling Laws

With growing model size more and more capabilities emerge



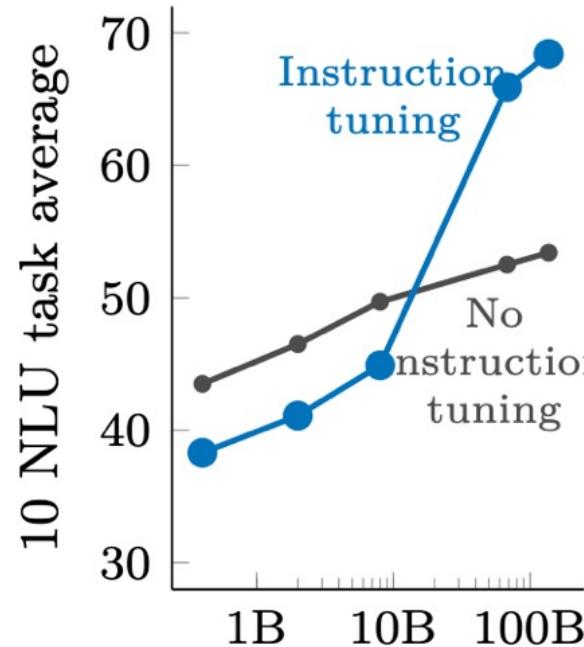


Multi-Step Reasoning,

where LLMs can solve a complex task by breaking down that task into intermediate reasoning steps as demonstrated in the chain-of-thought prompt

Chain-of-Thought prompting surpasses standard prompting without intermediate steps when scaled up to 100B parameter

<https://arxiv.org/abs/2206.07682>

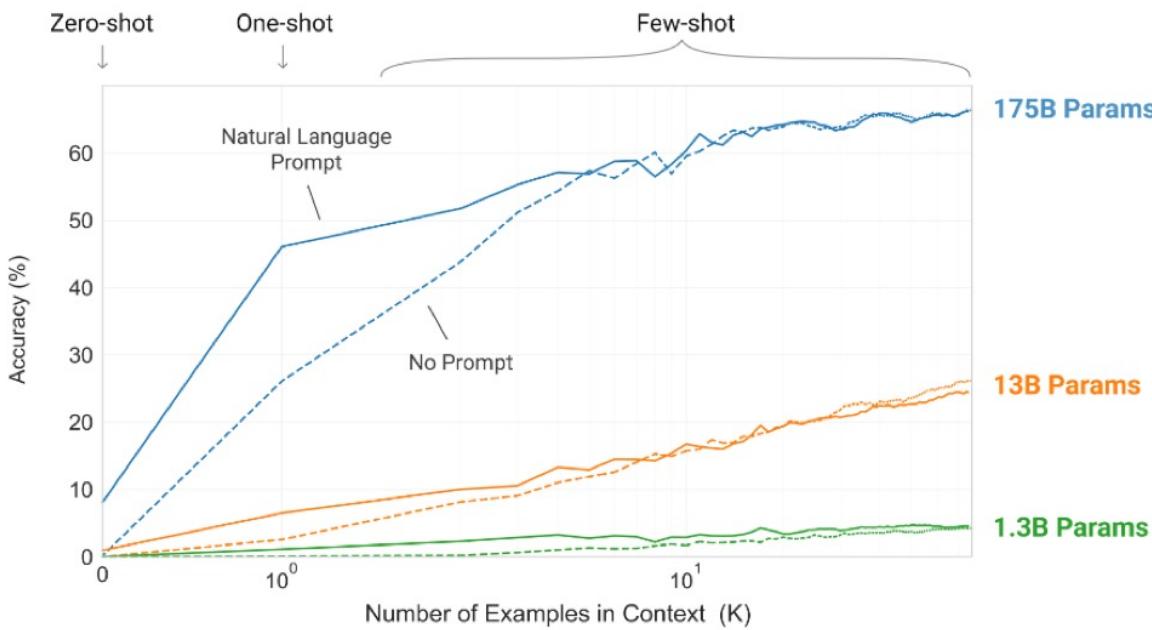


Instruction Following,

where LLMs, after instruction tuning, can follow the instructions for new types of tasks without using explicit examples,

By finetuning on a mixture of tasks phrased as instructions, language models have been shown to respond appropriately to instructions describing an unseen task (scaled up to 8B)

<https://arxiv.org/abs/2206.07682>



<https://arxiv.org/pdf/2402.06196.pdf> Fig-9

In-Context Learning,

where LLMs learn a new task from a small set of examples presented in the prompt at inference time.

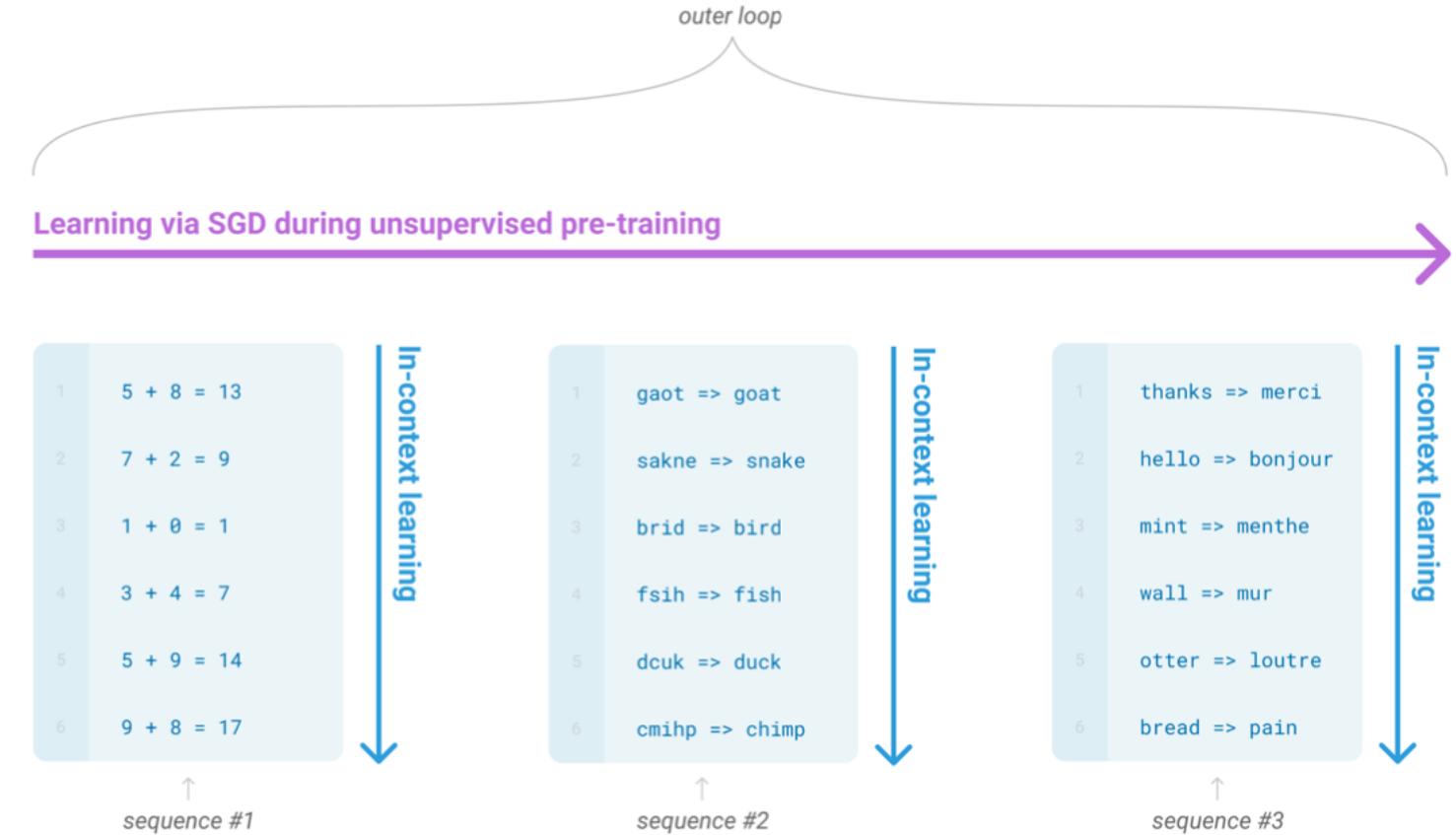
GPT-3 shows that larger models make increasingly efficient use of in-context information. It shows in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description.

In-Context Learning

During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities.

It then uses these abilities at inference time to rapidly adapt to or recognize the desired task.

We use the term “in-context learning” to describe the inner loop of this process, which occurs within the forward-pass upon each sequence.



In-Context Learning

Larger models make increasingly efficient use of in-context information

The three settings we explore for in-context learning

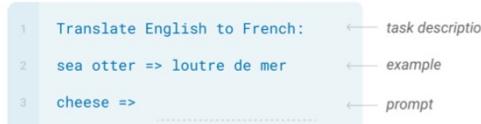
Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



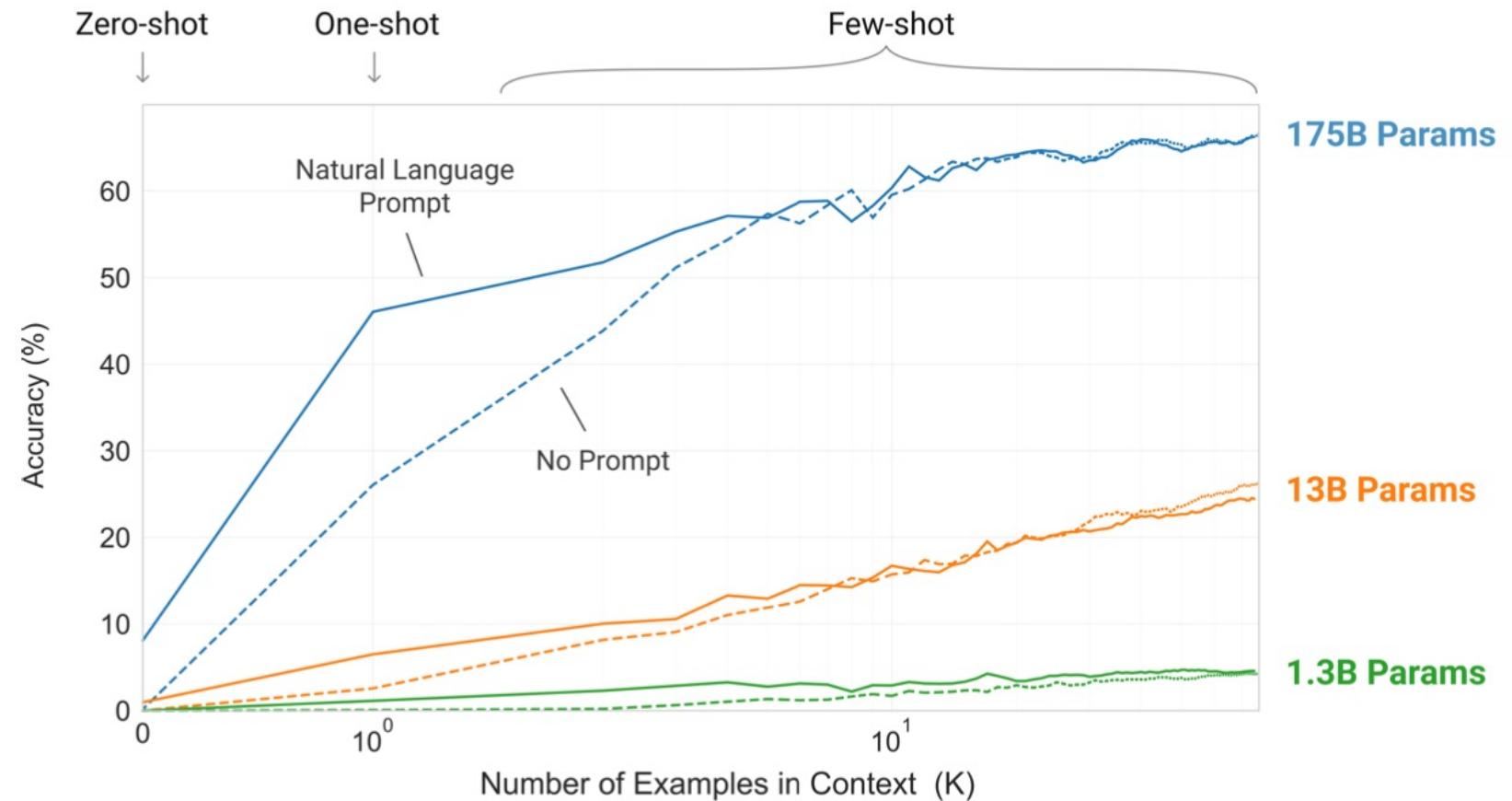
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



GPT-3 Language Models are Few-Shot Learners, <https://arxiv.org/abs/2005.14165>

02

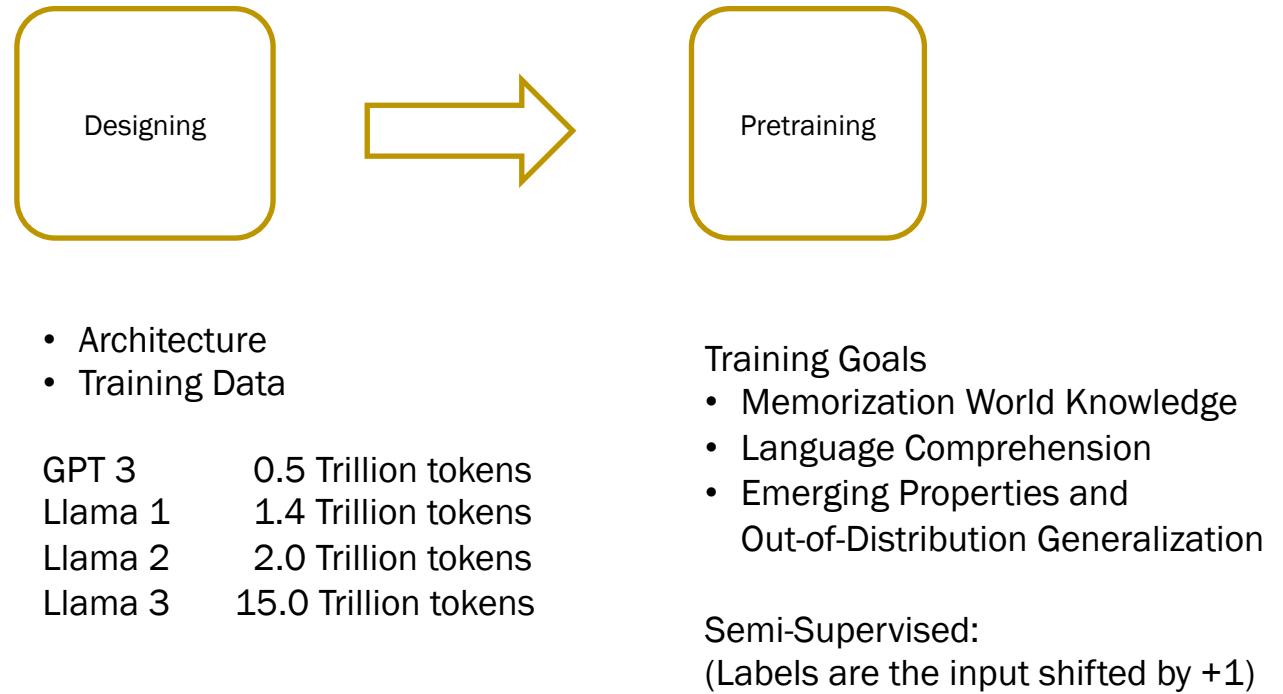
Building LLMs: From Pretraining to Posttraining



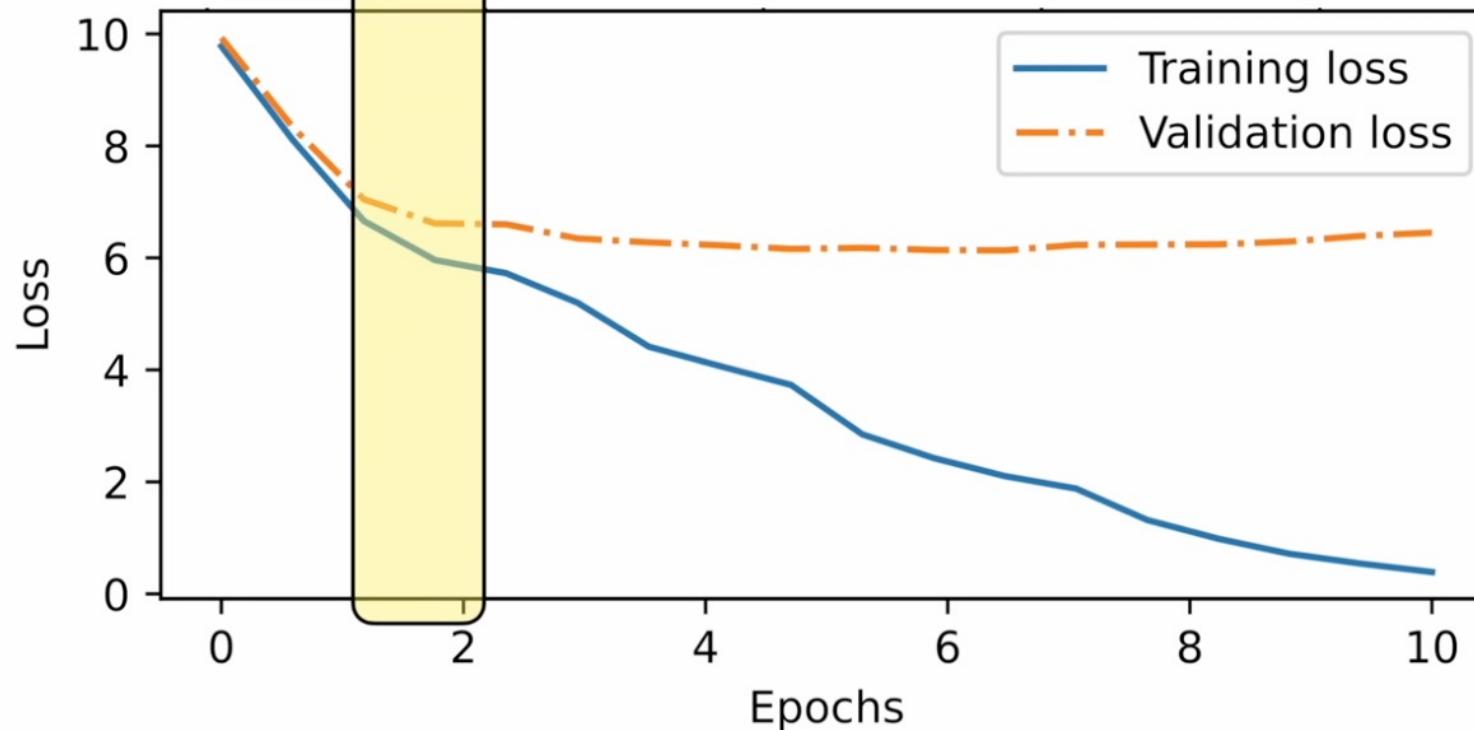
How Are LLMs built?

Six Steps of Building LLMs:

1. Data Cleaning
2. Tokenization
3. Positional Encoding
4. Model Pre-Training
5. Fine-tuning and/or
6. Instruction Tuning
7. Model Alignment



Training for ~1-2 epochs is usually a good sweet spot



Source: “Build a Large Language Model (From Scratch)”,
Sebastian Raschka, <https://www.manning.com/books/build-a-large-language-model-from-scratch>

Pure Language Modelling is NOT what you want
when you have an AI assistant!



PROMPT: „Explain the moon landing to a 6 years old in a few sentences.“

COMPLETION: Explain the moon landing to a 6 years old in a few sentences.

The first thing you need to do is to get the moon landing to a 6 years old in a few sentences.

The second thing you need to do is to get the moon landing to a 6 years old in a few sentences.

The third thing you need to do is to get the moon landing to a 6 years old in a few sentences.

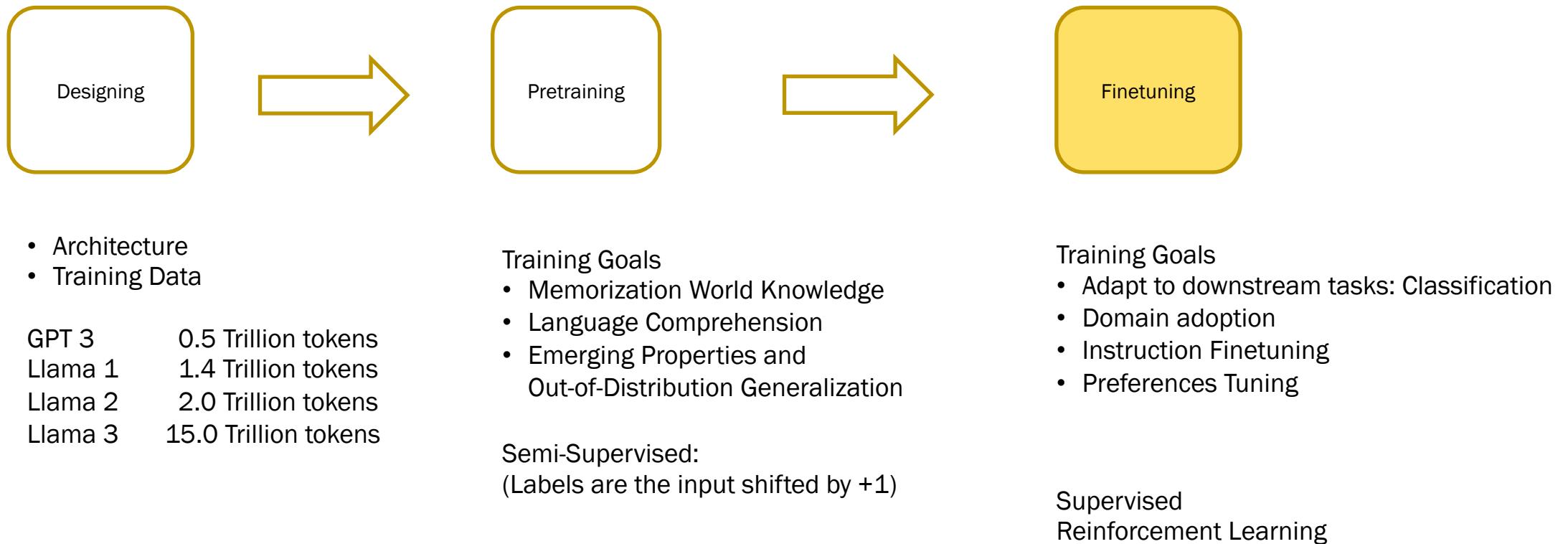
The fourth thing you need to do is to get the moon landing to a 6 years old in a few sentences.

....

No good answer for an assistant! How can we align the to question – answering?



From Pre-Training to Fine-Tuning



Source: “Build a Large Language Model (From Scratch)”,
Sebastian Raschka, <https://www.manning.com/books/build-a-large-language-model-from-scratch>

Six Steps of Building LLMs:

1. Data Cleaning
2. Tokenization
3. Positional Encoding
4. Model Pre-Training
5. Fine-tuning and/or
 - How LLMs are trained on custom datasets?
 - How LLMs are trained to better follow instructions?
6. Instruction Tuning
 - How LLMs are aligned to be used for chat?
7. Model Alignment

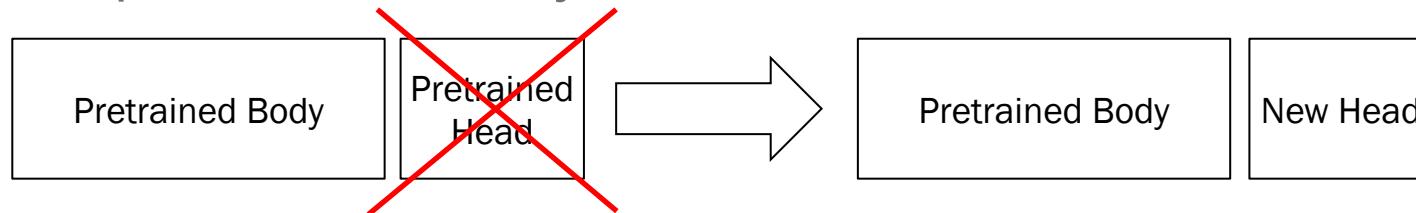
Fine-tune a LLM always starts with a specific intention: Why!

- Fine-Tuning for a specific task
 - Named Entity Recognition
 - Sentiment Analysis
 - Text Classification
- Fine-Tuning for domain adaption
- Fine-Tuning for better instruction following (Instruction Tuning)



There are two approaches for fine-tuning:

1. Chop off the final Layer and add a new one

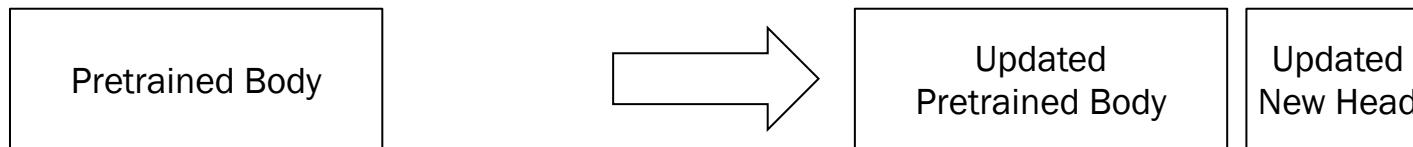


- chosen if the task the model was pretrained is different from the task the model should be finetuned
- remove the final layer and add a new one
- the new head is trained from scratch and the rest is kept frozen



There are two approaches for fine-tuning:

2. Finetune everything

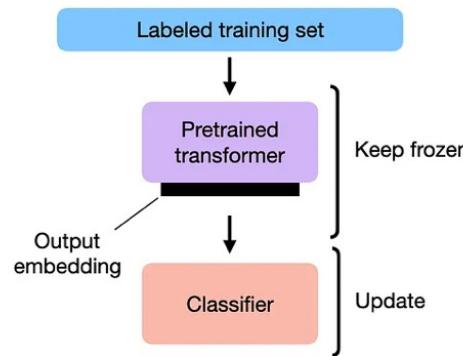


- all parameters of the pre-trained model are updated
- much more expensive
- usually done when the target task is similar to the task the model was pretrained

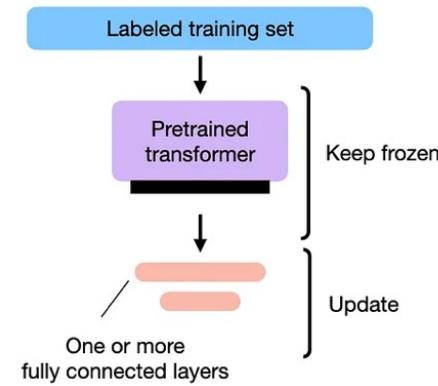


Fine Tuning - Do we need to finetune all layers?

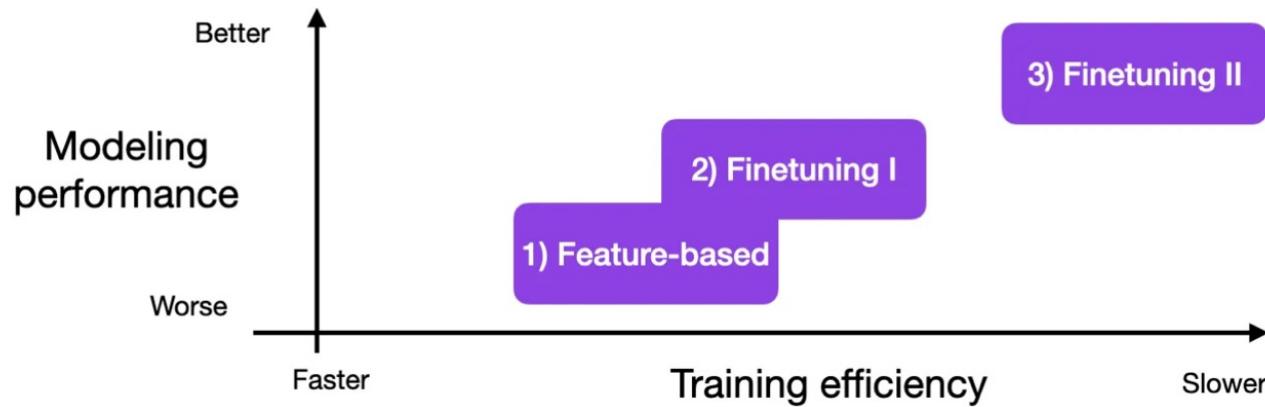
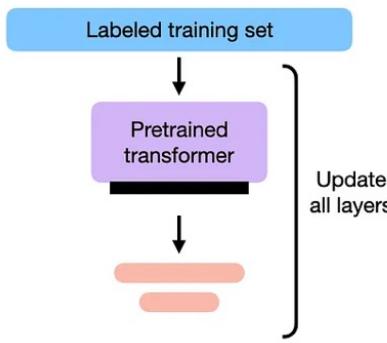
1) FEATURE-BASED APPROACH



2) FINETUNING I



3) FINETUNING II



Source: <https://magazine.sebastianraschka.com/p/finetuning-large-language-models>

Fine-tuning BERT for text classification



Fine-tuning GPT2 for domain adaption

02_GPT2_domain_q&a_adoption.ipynb

02_GPT2_domain_adoption.ipynb



Idea: finetune the LLM with language modeling of the desired answer.

Next word Prediction

Supervised

This was the big jump from GPT3 to ChatGPT

Alpaca 2023: <https://crfm.stanford.edu/2023/03/13/alpaca.html>

OpenAssistant 2023: <https://arxiv.org/pdf/2304.07327>

LIMA: <https://arxiv.org/pdf/2305.11206> Knowledge is in the pretrained LLM. You need very little data to SFT. Just learns the FORMAT of desired answers.



Instruction Tuning

1. Start with Instruction Dataset

```
{  
    "instruction": "Rewrite the following sentence using passive voice.",  
    "input": "The team achieved great results.",  
    "output": "Great results were achieved by the team."  
},
```

2. Apply prompt style template

Goes into the LLM

Expected output from LLM

Apply prompt style template (for example, Alpaca-style)

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:

Rewrite the following sentence using passive voice.

Input:

The team achieved great results.

Response:

Great results were achieved by the team.

Pass to LLM for supervised instruction finetuning

LLM

Source: <https://magazine.sebastianraschka.com/p/finetuning-large-language-models>

Preference Tuning

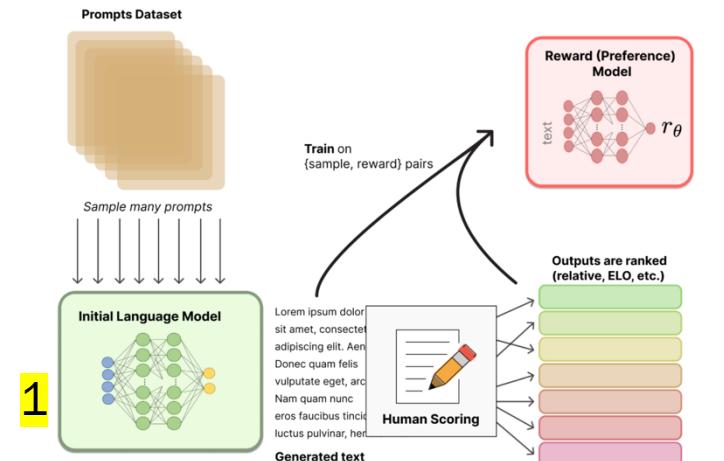
1. Start with a pre-trained or fine-tuned LLM
2. Generate a reward model (RM, also referred to as a preference model) calibrated with human preferences:

Input = “Some Text”

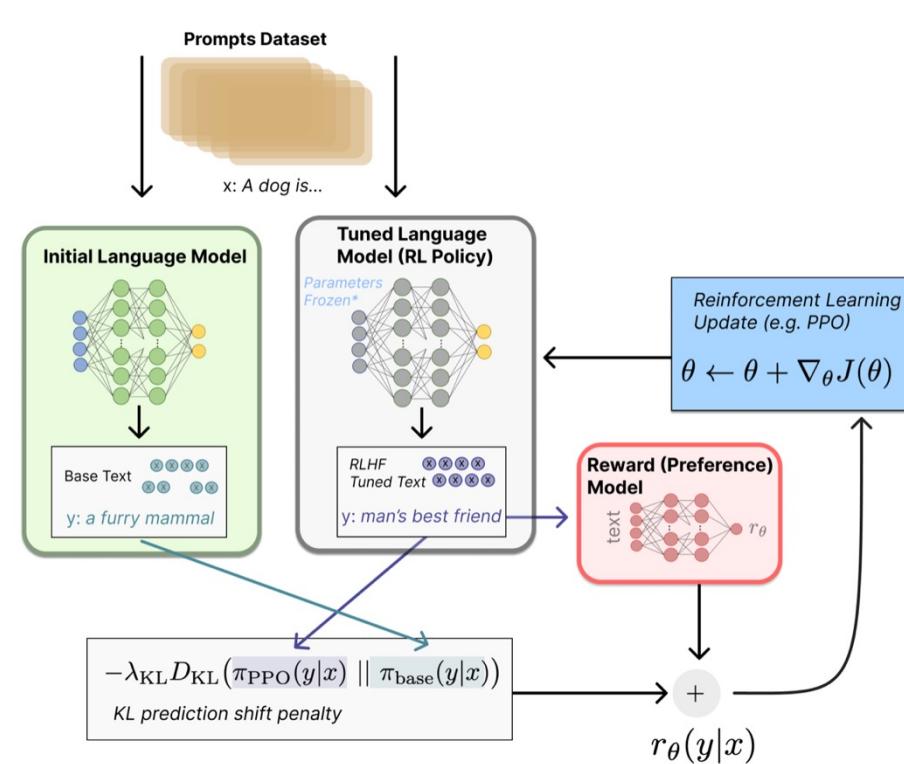
Output = scalar reward representing the human preference

3. Use the RM for Fine-tuning with Reinforcement Learning

- Probability Distribution of initial Model and tuned Model should be similar
- Use RM to penalize the tuned Model if it produces not preferred answers



2

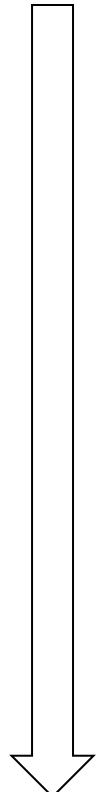


Pretraining from scratch → Expensive, almost never necessary

Continued pretraining → Add new knowledge

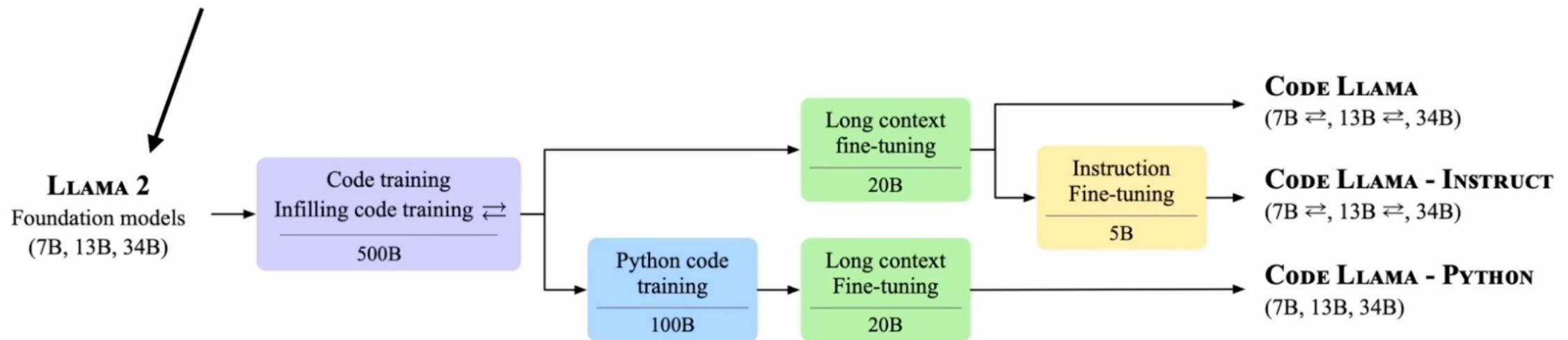
Finetuning → Special usecase, follow instructions

Preference finetuning → Improve helpfulness+safety if developing a chatbot



Example: Llama 2

Pretraining (from scratch)



Code Llama: Open Foundation Models for Code, <https://arxiv.org/abs/2308.12950>

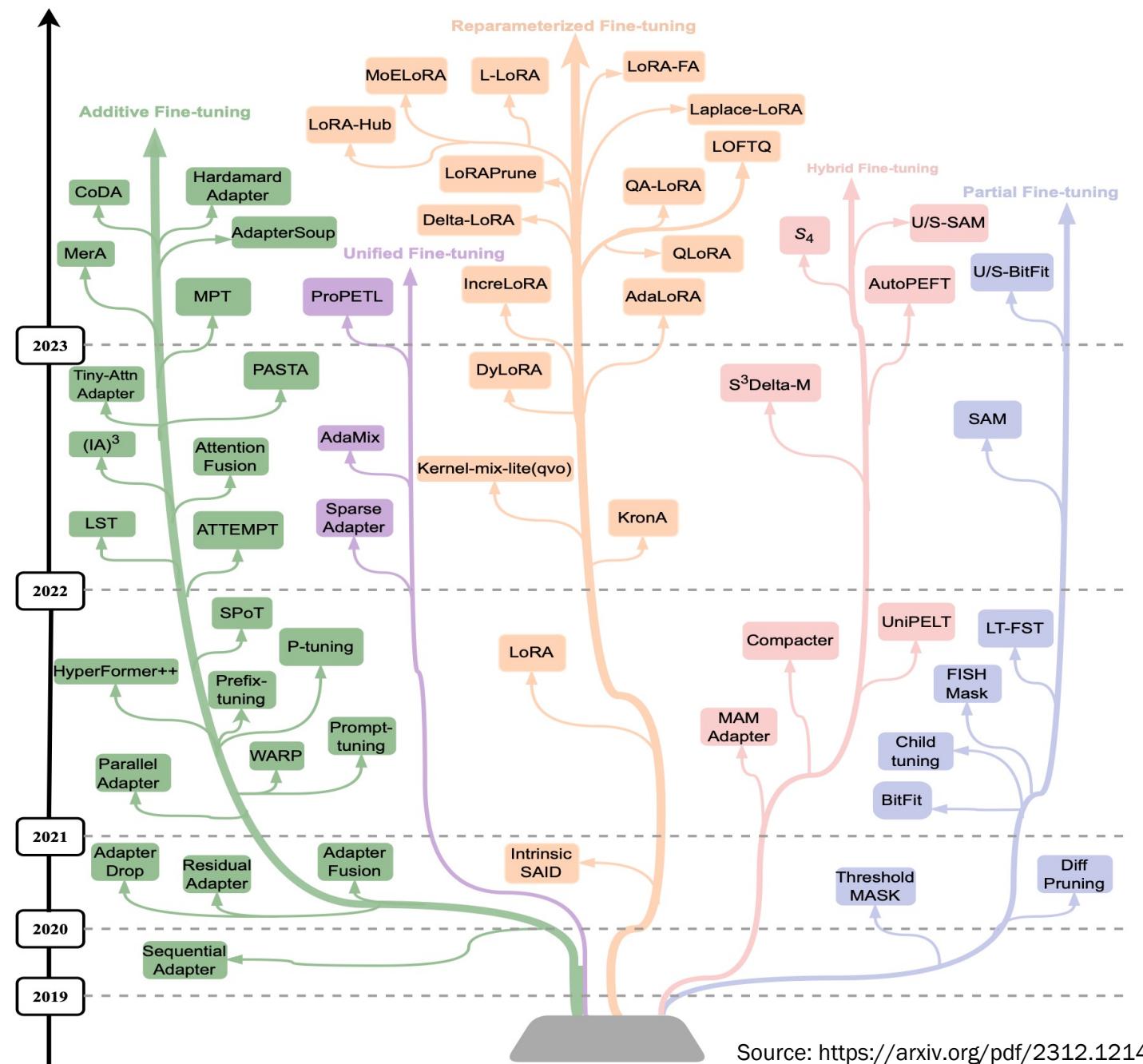
PEFT

Parameter Efficient Fine-Tuning Methods

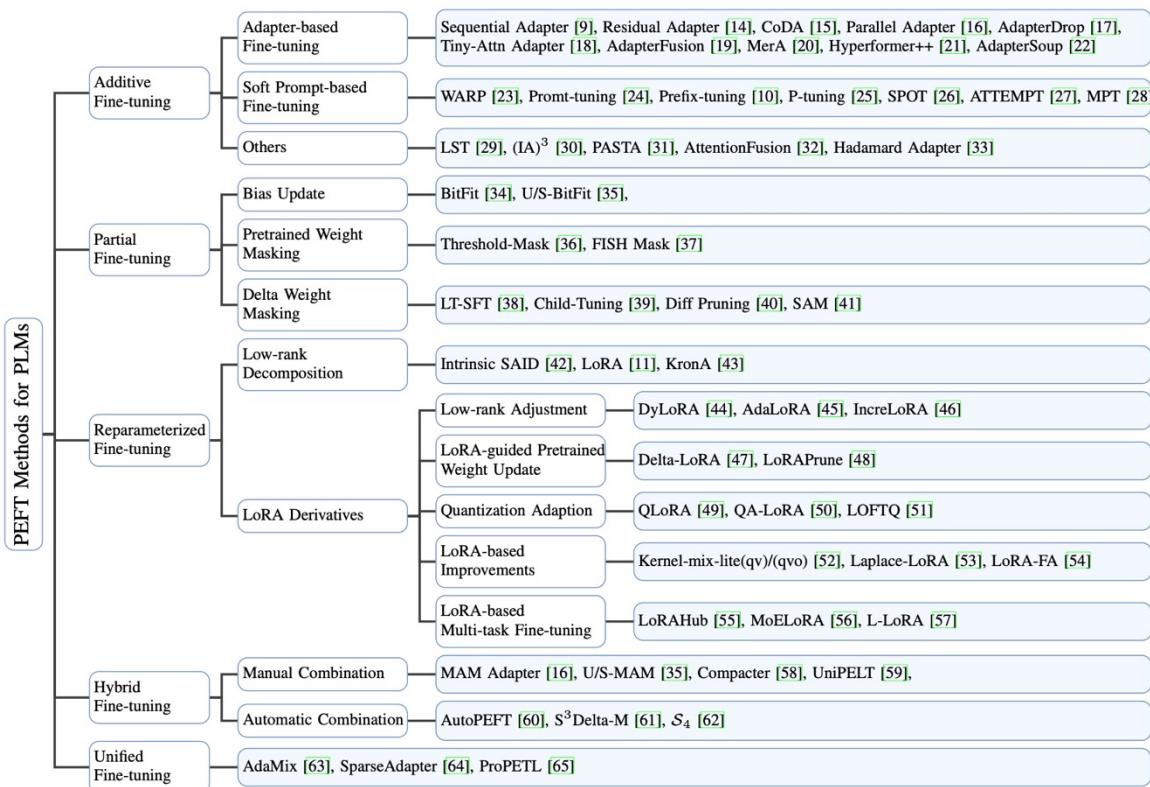
The evolutionary development of PEFT methods in recent years.

The tree shows scientific papers concerning fine tuning.

Methods on the same branch have some common features. The vertical position of the methods shows the timeline of their release dates of the papers.



Taxonomy of Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models.



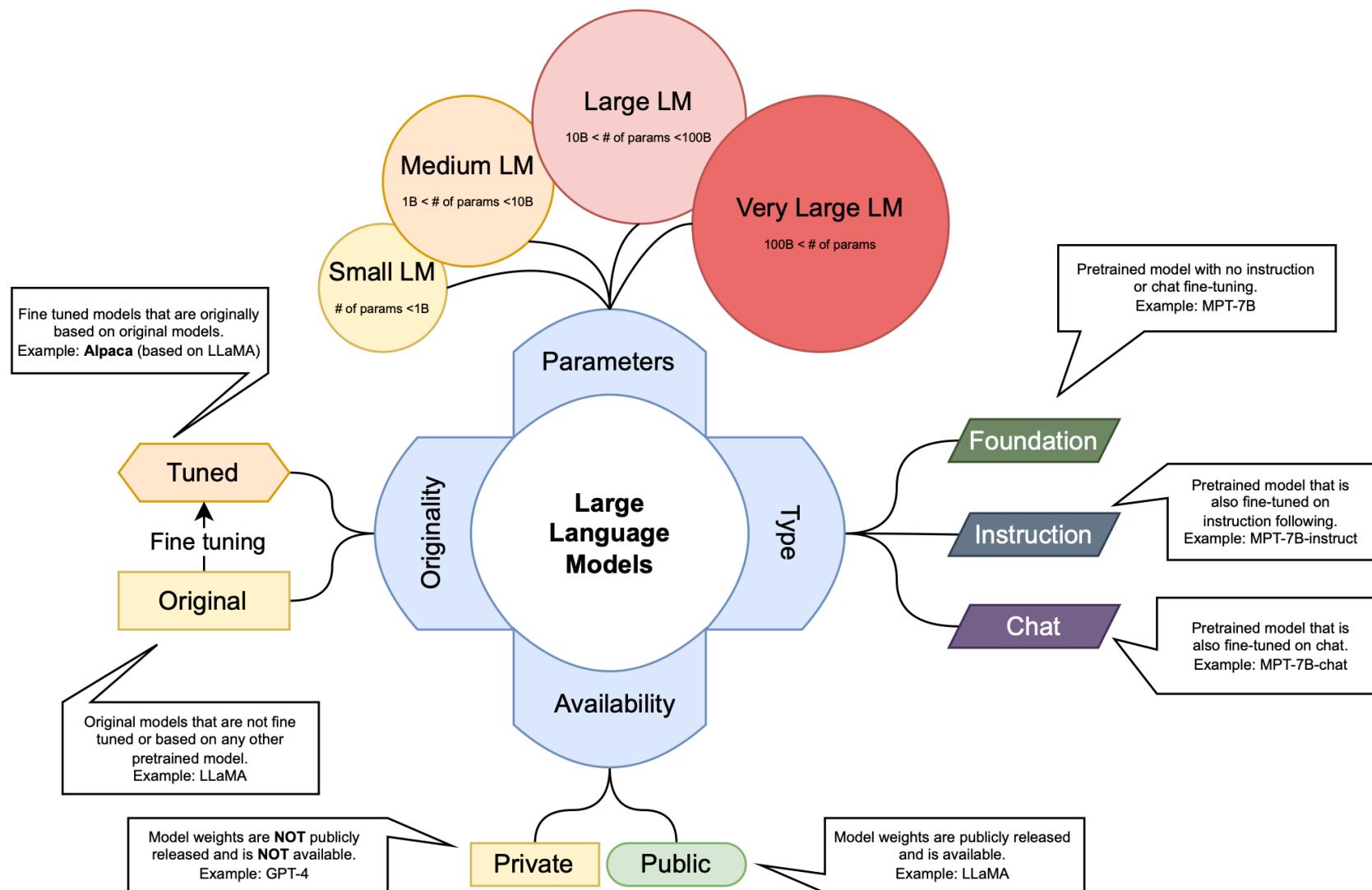
Parameter Efficient Fine-Tuning (PEFT) offers an effective solution by **reducing the number of fine-tuning parameters** and memory usage while achieving comparable performance to full fine-tuning.

PEFT can be classified into additive, partial, reparameterized, hybrid, and unified fine-tuning methods.

Most PEFT methods primarily focus on the self-attention layer and feed-forward network layer, allowing models like encoder-based, encoder-decoder-based, and decoder-based to leverage relevant techniques for parameters reduction.

Source: <https://arxiv.org/pdf/2312.12148.pdf>, Fig. 2

Summary



<https://arxiv.org/pdf/2402.06196.pdf> Fig-43

03

Model Evaluation



Frameworks:

- Stanford HELM - Holistic Evaluation of Language Models: <https://crfm-helm.readthedocs.io/en/stable/>
- Language Model Evaluation Harness: <https://github.com/EleutherAI/lm-evaluation-harness>

Leaderboards:

- Hugging Face Open LLM Leaderboard - a public leaderboard comparing open access large language models:
https://huggingface.co/spaces/open_llm-leaderboard/open_llm_leaderboard
(the Open LLM Leaderboard is actually just a wrapper running the open-source benchmarking library Eleuther AI LM Evaluation Harness created by the EleutherAI non-profit AI research lab)
- Stanford HELM – Leaderboards: <https://crfm.stanford.edu/helm/>



Benchmarks: Massive Multitask Language Understanding (MMLU)

Massive Multitask Language Understanding (MMLU) is a benchmark that can be used to test and assess

- general world knowledge and
 - problem-solving abilities
- of language models.

The benchmark covers a total of 57 different subject areas from the fields of STEM (science, technology, engineering and mathematics), humanities and social sciences, history and others. The difficulty level of the questions varies and ranges from basic knowledge to advanced, professional expert knowledge.

MMLU can be used to evaluate models in zero-shot and few-shot settings.

<https://github.com/hendrycks/test>

<https://crfm.stanford.edu/2024/05/01/helm-mmlu.html>

MMLU Leaderboard: <https://crfm.stanford.edu/helm/mmlu/latest/>



Benchmarks: Massive Multitask Language Understanding (MMLU)

How good is the model in answering multiple choice questions?

MMLU

MMLU = Measuring Massive Multitask Language Understanding (2020), <https://arxiv.org/abs/2009.03300>

Multiple-choice questions from diverse subjects

```
input = ("Which character is known for saying,  
        'To be, or not to be, that is the question'?  
        Options:  
        A) Macbeth, B) Othello,  
        C) Hamlet, D) King Lear.")  
  
model_answer = model(input)  
  
correct_answer = "C) Hamlet"  
  
score += model_answer == correct_answer  
  
# total_score = score / num_examples * 100%
```



Benchmarks: Other Evaluation Metrics

https://github.com/EleutherAI/lm-evaluation-harness/blob/main/Lm_eval/tasks/leaderboard/README.md

- BBH (3-shots, multichoice)
- GPQA (0-shot, multichoice)
- mmlu-pro (5-shots, multichoice)
- Musr (0-shot, multichoice)
- ifeval (0-shot, generative)
- Math-lvl-5 (4-shots, generative, minerva version)



Comparing a giving LLM against GPT-4

https://tatsu-lab.github.io/alpaca_eval/

AlpacaEval Leaderboard

An Automatic Evaluator for Instruction-following Language Models

Length-controlled (LC) win rates alleviate length biases of GPT-4, but it may favor models finetuned on its outputs.



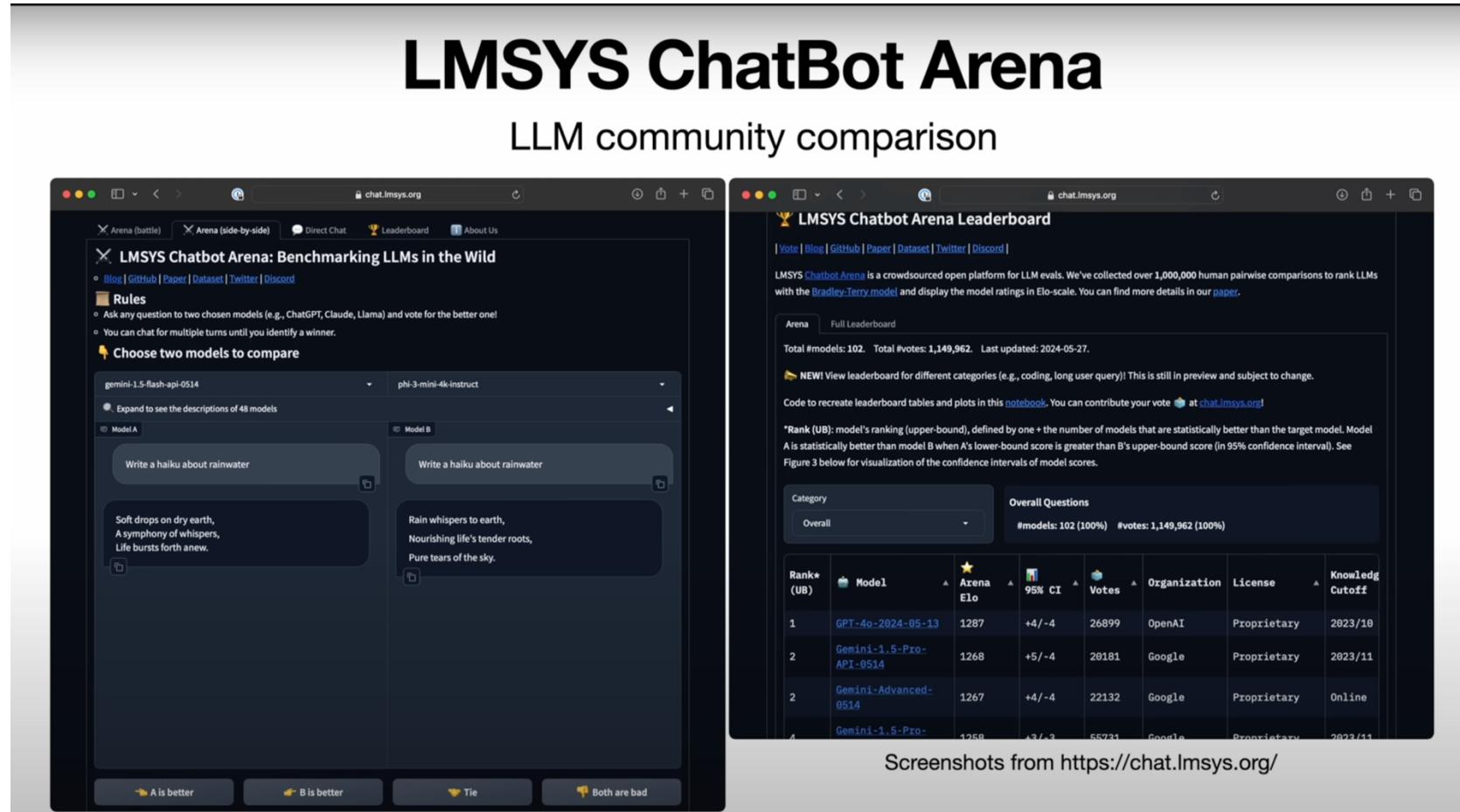
Version: AlpacaEval **AlpacaEval 2.0**

Filter: Community **Verified**

Baseline: GPT-4 Preview (11/06) | Auto-annotator: GPT-4 Preview (11/06)

Rank	Model Name	LC Win Rate	Win Rate
1	GPT-4 Omni (05/13)	57.5%	51.3%
2	GPT-4 Turbo (04/09)	55.0%	46.1%
3	Yi-Large Preview	51.9%	57.5%
4	GPT-4o Mini (07/18)	50.7%	44.7%
5	GPT-4 Preview (11/06)	50.0%	50.0%
6	Claude 3 Opus (02/29)	40.5%	29.1%
7	Llama 3.1 405B Instruct	39.3%	39.1%
8	GPT-4	38.1%	23.6%
9	Qwen2 72B Instruct	38.1%	29.9%
10	Llama 3.1 70B Instruct	38.1%	39.1%
11	Qwen1.5 72B Chat	36.6%	26.5%
12	GPT-4 (03/14)	35.3%	22.1%
13	Claude 3 Sonnet (02/29)	34.9%	25.6%
14	Llama 3 70B Instruct	34.4%	33.2%

<https://larena.ai/?leaderboard>



LMSYS ChatBot Arena

LLM community comparison

LMSYS Chatbot Arena: Benchmarking LLMs in the Wild

Rules

- Ask any question to two chosen models (e.g., ChatGPT, Claude, Llama) and vote for the better one!
- You can chat for multiple turns until you identify a winner.

Choose two models to compare

gemini-1.5-flash-api-0514 phi-3-mini-4k-instruct

Model A: Write a haiku about rainwater

Model B: Write a haiku about rainwater

Soft drops on dry earth,
A symphony of whispers,
Life bursts forth anew.

Rain whispers to earth,
Nourishing life's tender roots,
Pure tears of the sky.

Arena (battle) Arena (side-by-side) Direct Chat Leaderboard About Us

Vote | Blog | GitHub | Paper | Dataset | Twitter | Discord

LMSYS Chatbot Arena Leaderboard

Arena Full Leaderboard

Total #models: 102. Total #votes: 1,149,962. Last updated: 2024-05-27.

NEW! View leaderboard for different categories (e.g., coding, long user query)! This is still in preview and subject to change.

Code to recreate leaderboard tables and plots in this [notebook](#). You can contribute your vote [at chat.lmsys.org!](#)

*Rank (UB): model's ranking (upper-bound), defined by one + the number of models that are statistically better than the target model. Model A is statistically better than model B when A's lower-bound score is greater than B's upper-bound score (in 95% confidence interval). See Figure 3 below for visualization of the confidence intervals of model scores.

Category	Overall Questions
Overall	#models: 102 (100%) #votes: 1,149,962 (100%)

Rank* (UB)	Model	Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	GPT-4o-2024-05-13	1287	+4/-4	26899	OpenAI	Proprietary	2023/10
2	Gemini-1.5-Pro-API-0514	1268	+5/-4	20181	Google	Proprietary	2023/11
2	Gemini-Advanced-0514	1267	+4/-4	22132	Google	Proprietary	Online
4	Gemini-1.5-Pro-	1258	+3/-3	55731	Google	Proprietary	2023/11

Screenshots from <https://chat.lmsys.org/>

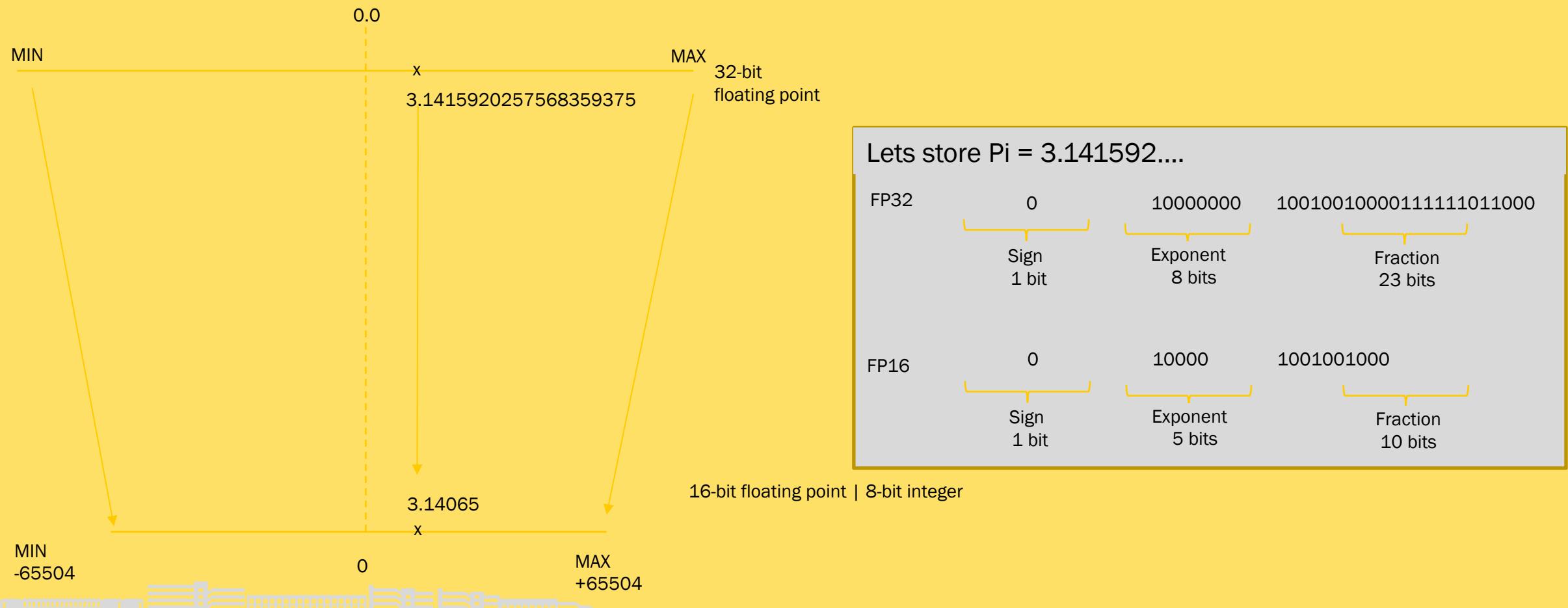


Quantization



Quantization

Quantization is a technique to reduce the computational and memory costs of running inference by representing the weights and activations with low-precision data types like 8-bit integer (int8) instead of the usual 32-bit floating point (float32). Quantization statistically projects the 32-bit floating point numbers into a lower precision space using scaling factors.



Quantization

	Bits	Exponent	Fraction	Memory needed to store one value
FP32	32	8	23	4 bytes
FP26	16	5	10	2 bytes
BFLOAT16	16	8	7	2 bytes
INT8	8		7	1 byte

Performing quantization to go from float32 to int8 is tricky. Only 256 values can be represented in int8, while float32 can represent a very wide range of values. The idea is to find the best way to project our range $[a, b]$ of float32 values to the int8 space.

For x (as float32) in $[a, b]$:
$$x = S * (x_q - Z)$$

$$x_q \text{ is the quantized int8 value associated to } x,$$

$$S \text{ and } Z \text{ are quantization parameters}$$

$$x_q = \text{round}(x/S + Z)$$

From: https://huggingface.co/docs/optimum/concept_guides/quantization

04

Inference Engine



Definition:

An inference engine interprets and evaluates the facts in the knowledge base in order to provide an answer. (<https://www.britannica.com/technology/inference-engine>)

In the field of artificial intelligence, an inference engine is a software component of an intelligent system that applies logical rules to the knowledge base to deduce new information. (https://en.wikipedia.org/wiki/Inference_engine)



Inference Engine

For us, an inference engine is a runtime software component, that provides inference services of a LLM like “text generation” via an API.

Typical Inference Engines are:

<https://ollama.com>



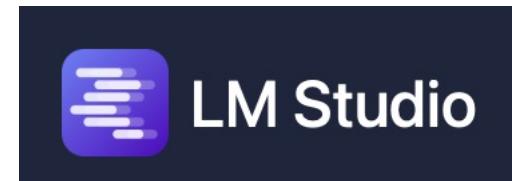
<https://www.nomic.ai/gpt4all>



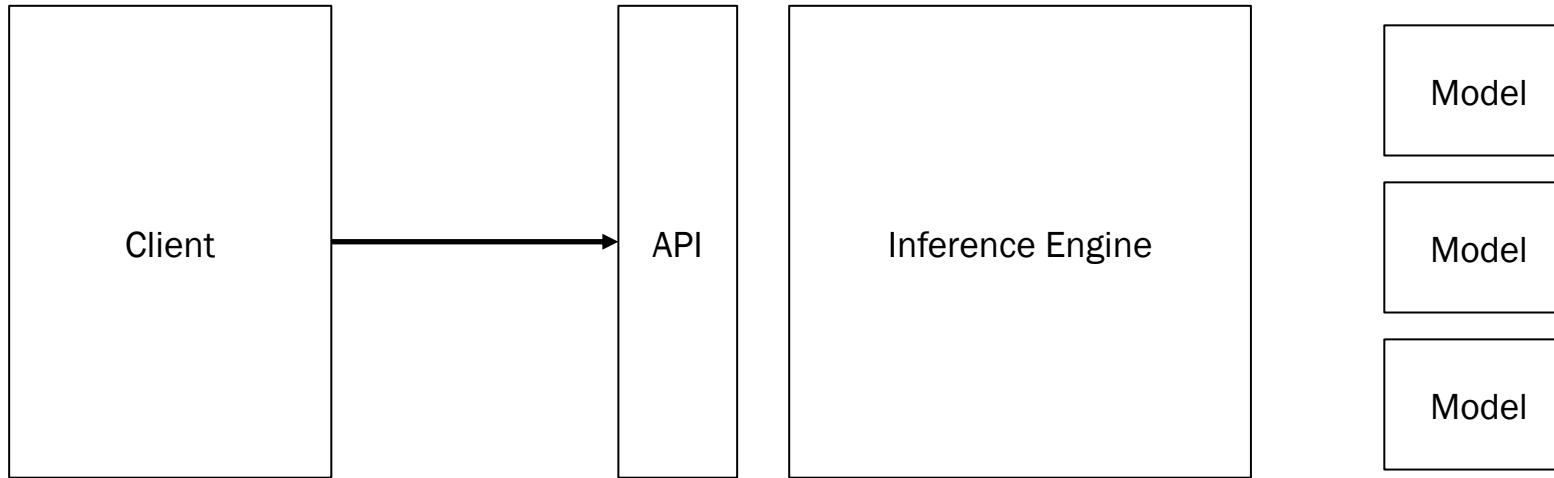
<https://docs.vllm.ai>



<https://lmstudio.ai>



Inference Engine



Inference Engine



Ollama <https://ollama.com>

- Runs as a service
- CLI 
- Python Client Library: pip install ollama
<https://github.com/ollama/ollama-python>
- Provides OpenAI API for text generation
<https://platform.openai.com/docs/guides/text-generation>

```
[base) done@MN-EX6661 ~ % ollama
Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve      Start ollama
  create     Create a model from a Modelfile
  show       Show information for a model
  run        Run a model
  stop       Stop a running model
  pull       Pull a model from a registry
  push       Push a model to a registry
  list       List models
  ps         List running models
  cp         Copy a model
  rm         Remove a model
  help       Help about any command

Flags:
  -h, --help            help for ollama
  -v, --version         Show version information
```

Inference Engine

Exercise:

- Install and start Ollama Server
- Use CLI to download and run LLama3.2
- Open VSCode and write some python scripts to connect to Ollama Server
 - via Ollama python library and
 - via OpenAI API (see <https://ollama.com/blog/openai-compatibility>)

05

User Interface



Exercise: What are the requirements for a useful chatbot?



Requirements

Streaming: A complete response from the LLM may take up to 10–20 seconds, while the first tokens are available in less than a second, so streaming responses should be a must.

Model Selection: Chatbot should provide the possibility to change the model.

Generation: Chatbot should provide the ability to set the generation parameters: temperature, top-k, top-p, beam-search, ...



Requirements

Chat: User should be able to start a new chat.

Chat-History: Chatbot should save the chat history as short-term memory for each chat

Long-term memory: If user starts a new chat the old one should be saved as long-term memory



Requirements

Layout:

- Sidebar: Model Selection, Generation Parameters, Long-term Memory should be placed on sidebar.
- Chat Messages: questions of user and answers of assistents should be placed left - right



Chat Messages in Chat Completion API

<https://platform.openai.com/docs/guides/text-generation>

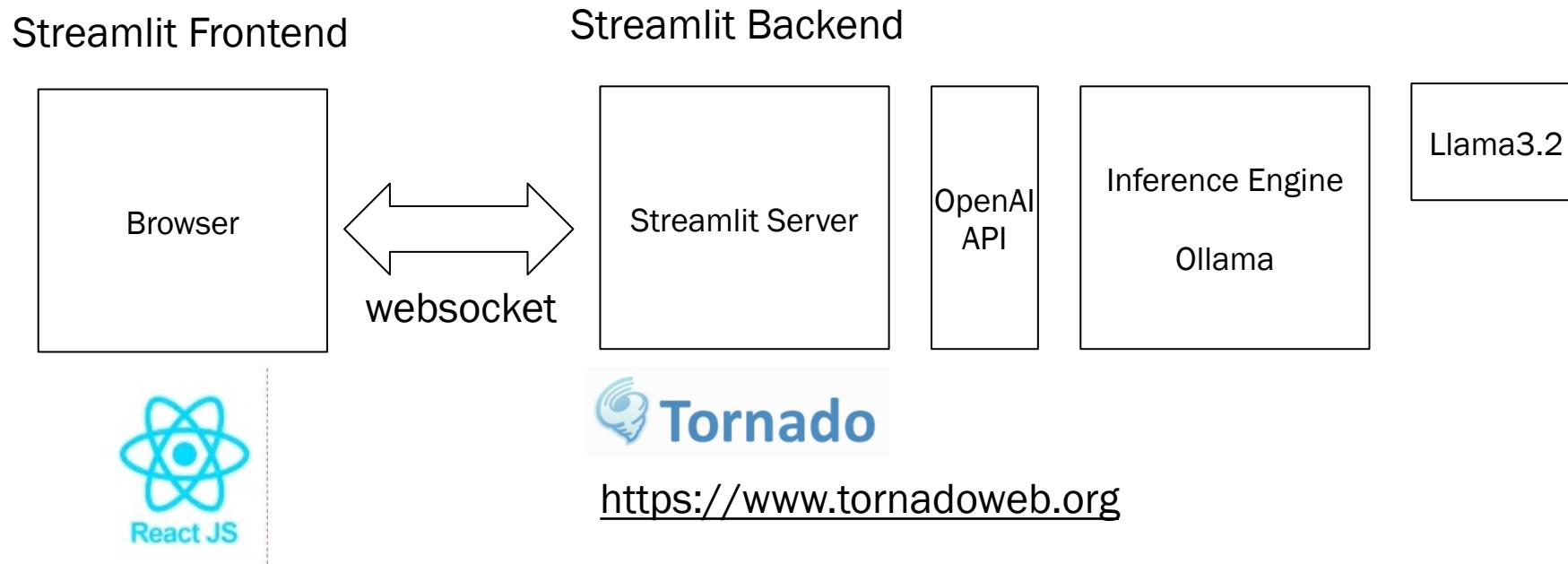
```
messages=[  
  { "role": "system",  
    "content": "You are a helpful assistant."},  
  { "role": "user",  
    "content": "Write a haiku about recursion in programming." }  
  { "role": "assistant",  
    "content": "some answer...." } ]
```



Streamlit - A faster way to build and share data apps



<https://streamlit.io>



Basic Concepts

Streaming: It is common for chatbots to implement websockets for streaming responses, and Streamlit uses websockets for everything by default.

Dataflow: Streamlit apps have a unique data flow: any time something must be updated on the screen, Streamlit reruns your entire Python script from top to bottom.

- modifying your app's source code.
- user interacts with widgets in the app



Basic Concepts

Streaming: It is common for chatbots to implement websockets for streaming responses, and Streamlit uses websockets for everything by default.

Dataflow: Streamlit apps have a unique data flow: any time something must be updated on the screen, Streamlit reruns your entire Python script from top to bottom.

- modifying your app's source code.
- user interacts with widgets in the app



Recap: Session Handling in Web Applications

Streamlit Stateless vs Stateful

<https://medium.com/accreddian/supercharging-your-streamlit-apps-unlocking-session-state-ccd44a0ae097>

Session Affinity in Kubernetes

<https://pauldally.medium.com/session-affinity-and-kubernetes-proceed-with-caution-8e66fd5deb05>

Share Streamlit Session State across all sessions

<https://github.com/whitphx/streamlit-server-state>

Streamlit on Kubernetes

<https://docs.streamlit.io/deploy/tutorials/kubernetes>

BUT: There is no way to share streamlit session state across pods:

<https://discuss.streamlit.io/t/keep-streamlit-app-running-even-if-pods-dies/40404>



Basic Concepts

Session Handling: When you deploy a Streamlit app, there is a single, Python backend (server). Different threads run to handle different users.

There is a session object `st.session_state` you can use to store your session state.

If you write anything to `st.session_state`, that information is only going to be available to the session in which it was written. (If a person opens multiple tabs in their browser to the same app, then each tab is going to be its own session.)



Exercice Build a Chatbot with Streamlit, Ollama Backend and OpenAI API



- open-source Python package to build production ready Conversational AI
- <https://github.com/Chainlit/chainlit>
- <https://docs.chainlit.io/get-started/overview>

Basic Concept:

- Whenever a user connects to your Chainlit app, a new chat session is created.
- A chat session goes through a life cycle of events, which you can respond to by defining hooks.



Exercice Build a Chatbot with Chainlit, Ollama Backend and OpenAI API

- Goto block2
- Create NEW environment:

```
conda create --name chainlit python=3.11
```

```
conda activate chainlit
```

```
pip install chainlit openai
```

```
mkdir chainlit
```

```
cd chainlit
```

```
chainlit init
```

← Initializes chainlit project

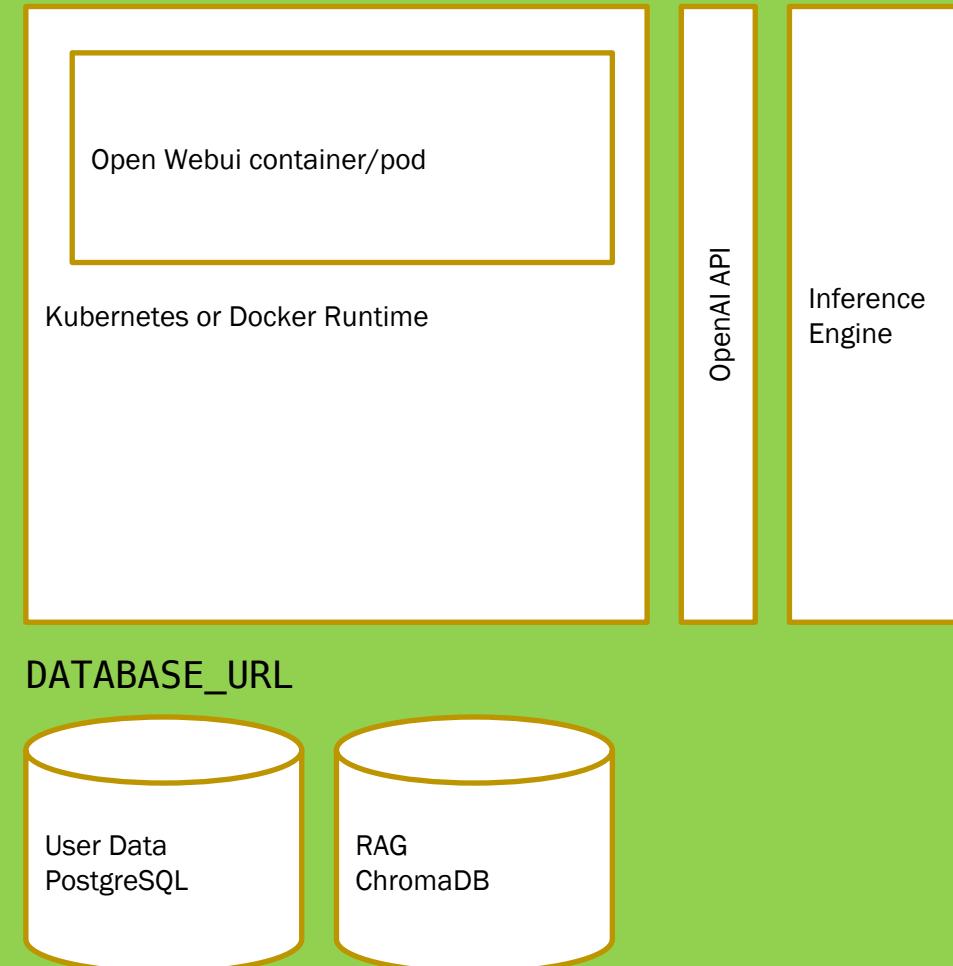
```
chainlit hello
```

← Starts hello demo



Open WebUI

Architecture



Open WebUI

```
git clone https://github.com/open-webui/open-webui.git
cd open-webui
docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v path-to-your-
data-on-host:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-
webui:main
```



Open WebUI

```
docker run
```

```
-d
```

```
-p 3000:8080
```



port mapping

```
--add-host=host.docker.internal:host-gateway
```

```
-v "/Users/done/Documents/Hochschule/WS 2024/WI-Intelligente-  
Informationssysteme/ws24/block2/data":/app/backend/data
```



mounting

```
--name open-webui
```



container name

```
--restart always
```

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



image



06

Prompt Engineering



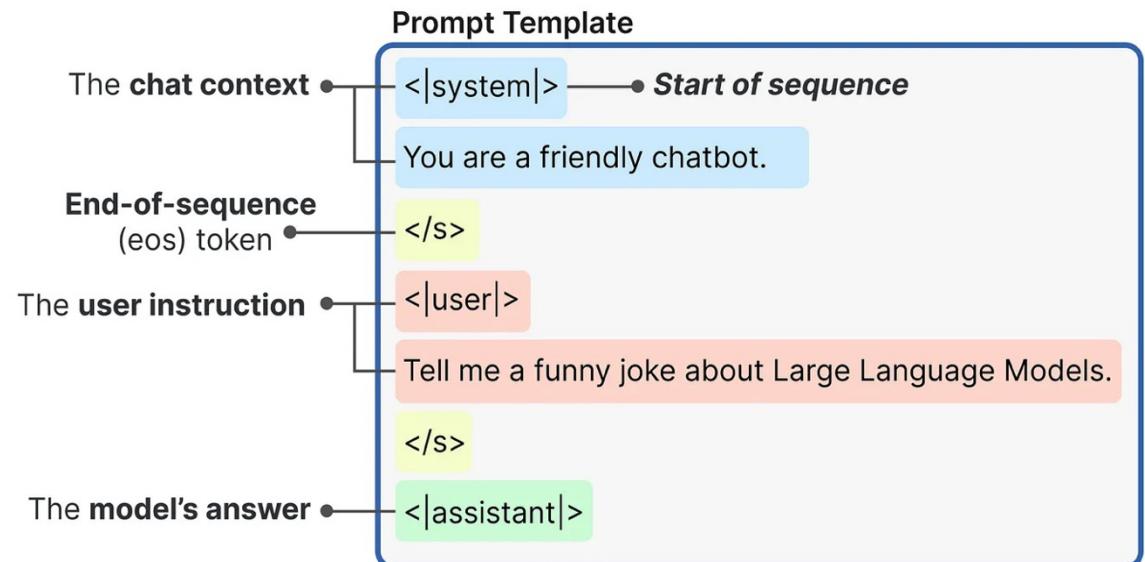
Prompt Engineering

What is a Prompt?

- A prompt is an input to a generative AI model, that is used to guide its output.
- Prompts may consist of text, image, sound, or other media.
- Prompts are often constructed via a **prompt template**:

A prompt template is a function that contains one or more variables which will be replaced by some media(usually text) to create a prompt. This prompt can then be considered to be an instance of the template.

A prompt template becomes a prompt when input is inserted into it.



Components of a Prompt

Prompt Component		Example
Directive	Many prompts issue a directive in the form of an instruction or question.	Write a poem about trees.
Output Formatting	Return out in a specific fomat: for example, JSON, CSVs or markdown	List all tree species as CSV.
Style Instructions	Style instructions are a type of output formatting used to modify the output stylistically rather than structural.	Write a clear and curt paragraph about llamas.
Role	A Role, also known as a persona, is a frequently discussed component that can improve writing and style text	Pretend you are a shepherd and write a limerick about llamas.
Additional Information (or context)	It is often necessary to include additional information to provide and set context.	



Prompt Engineering

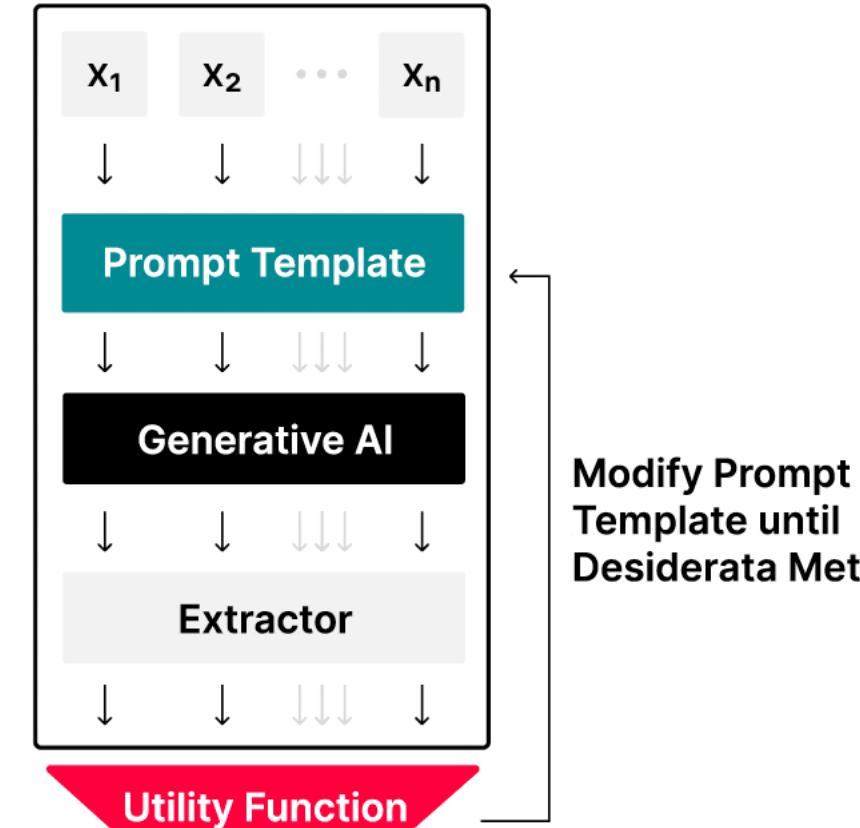
The Prompt Engineering Process consists of

three repeated steps:

1. performing inference on a dataset
2. evaluating performance and
3. modifying the prompt template.

Note that the extractor is used to extract a final response from the LLM output

Dataset Inference (i.e. entries $x_1 \dots x_n$)



Source: [PET] <https://arxiv.org/pdf/2406.06608.pdf> - Figure 1.4

In-Context Learning (ICL):

- ICL refers to the ability of GenAIs to learn skills and tasks by providing them with exemplars and or relevant instructions within the prompt, without the need for weight updates/retraining.
- These skills can be learned from exemplars and/or instructions.
- The word 'learn' is misleading. ICL can simply be task specification – the skills are not necessarily new, and can have already been included in the training data

Source: [PET] <https://arxiv.org/pdf/2406.06608.pdf> Figure 2.3

In-Context Learning (ICL):

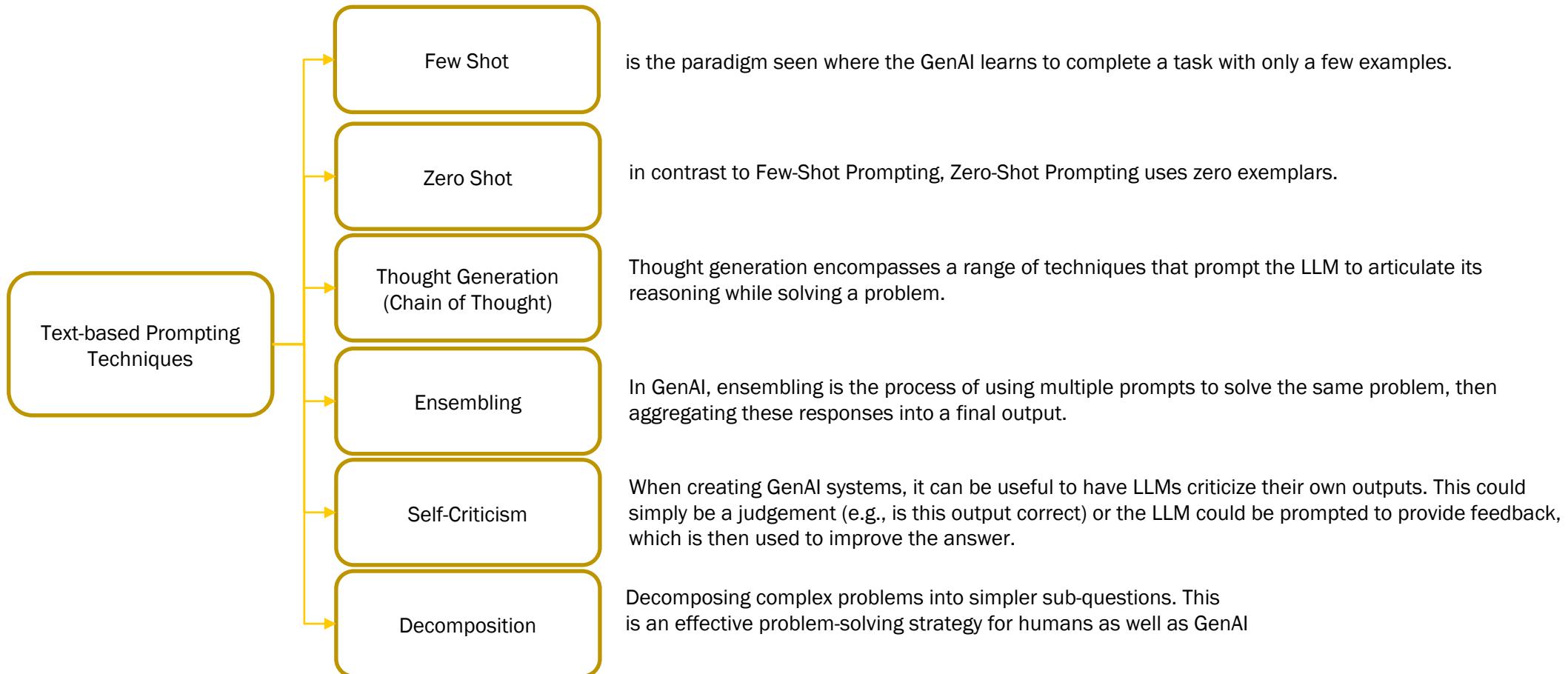
- Dos and Don'ts

1. Exemplar Quantity	2. Exemplar Ordering	3. Exemplar Label Distribution	4. Exemplar Label Quality	5. Exemplar Format	6. Exemplars Similarity
<p>Include as many exemplars as possible*</p> <p>Trees are beautiful: Positive I hate Pizza: Negative Squirrels are so cute: Positive YouTube Ads Suck: Negative I'm so excited:</p>	<p>Randomly order exemplars*</p> <p>I am so mad: Angry I love life: Happy I hate my boss: Angry Life is good: Happy I'm so excited:</p>	<p>Provide a balanced label distribution*</p> <p>I am so mad: Angry I love life: Happy I hate my boss: Angry Life is good: Happy I'm so excited:</p>	<p>Ensure exemplars are labeled correctly*</p> <p>I am so mad: Angry I love life: Happy I hate my boss: Angry Life is good: Happy I'm so excited:</p>	<p>Choose a common format*</p> <p>Im hyped!: Positive Im not very excited: Negative I'm so excited:</p>	<p>Select similar exemplars to the test instance*</p> <p>Im hyped!: Positive Im not very excited: Negative I'm so excited:</p>
<p>✓</p> <p>Trees are beautiful: Positive I'm so excited:</p>	<p>I love life: Happy Life is good: Happy I am so mad: Angry I hate my boss: Angry I'm so excited:</p>	<p>✗</p> <p>I am so mad: Angry People can be so dense: Angry I hate my boss: Angry Life is good: Happy I'm so excited:</p>	<p>I am so mad: Happy I love life: Angry I hate my boss: Angry Life is good: Happy I'm so excited:</p>	<p>Trees are nice==Positive YouTube Ads Suck==Negative I'm so excited==</p>	<p>Trees are beautiful: Positive YouTube Ads Suck: Negative I'm so excited:</p>

Source: [PET] <https://arxiv.org/pdf/2406.06608.pdf> Figure 2.3

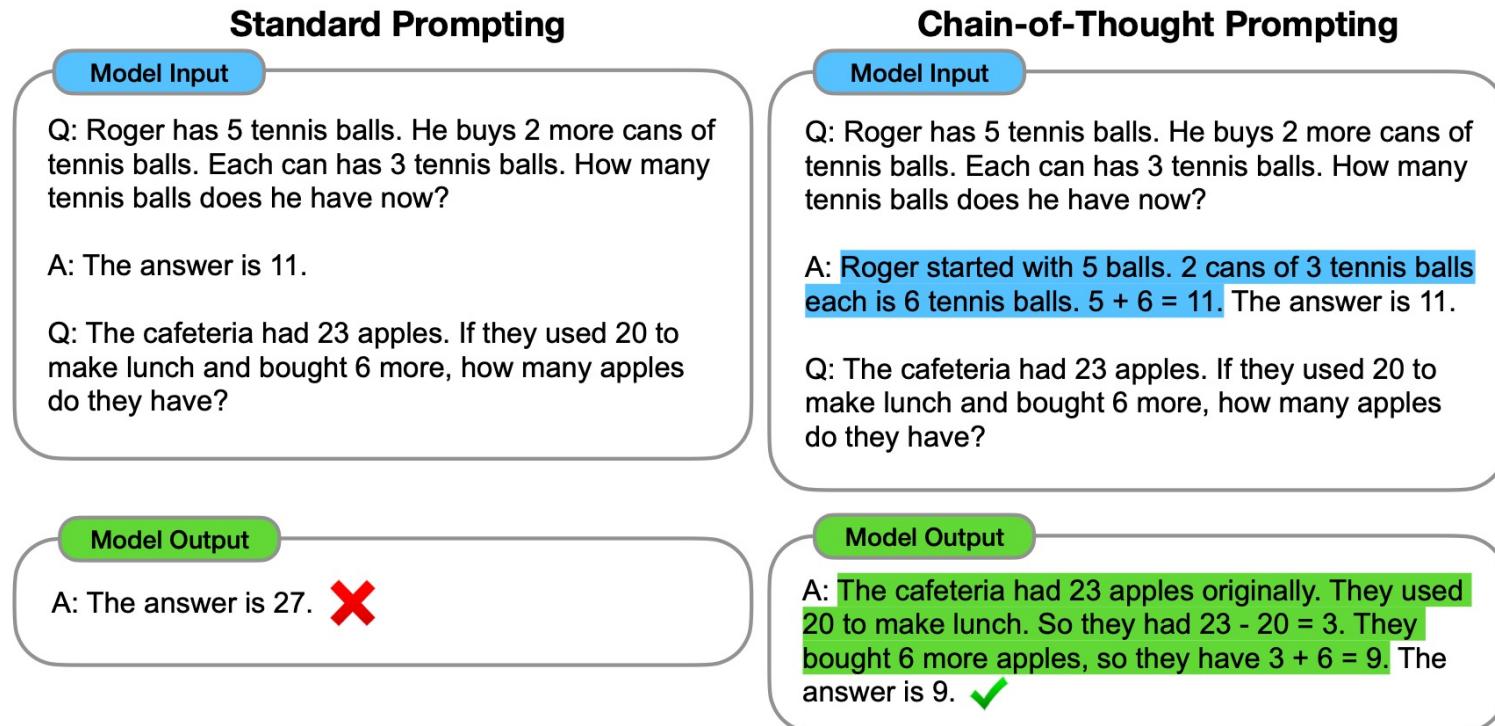


The 6 major Prompting Techniques:



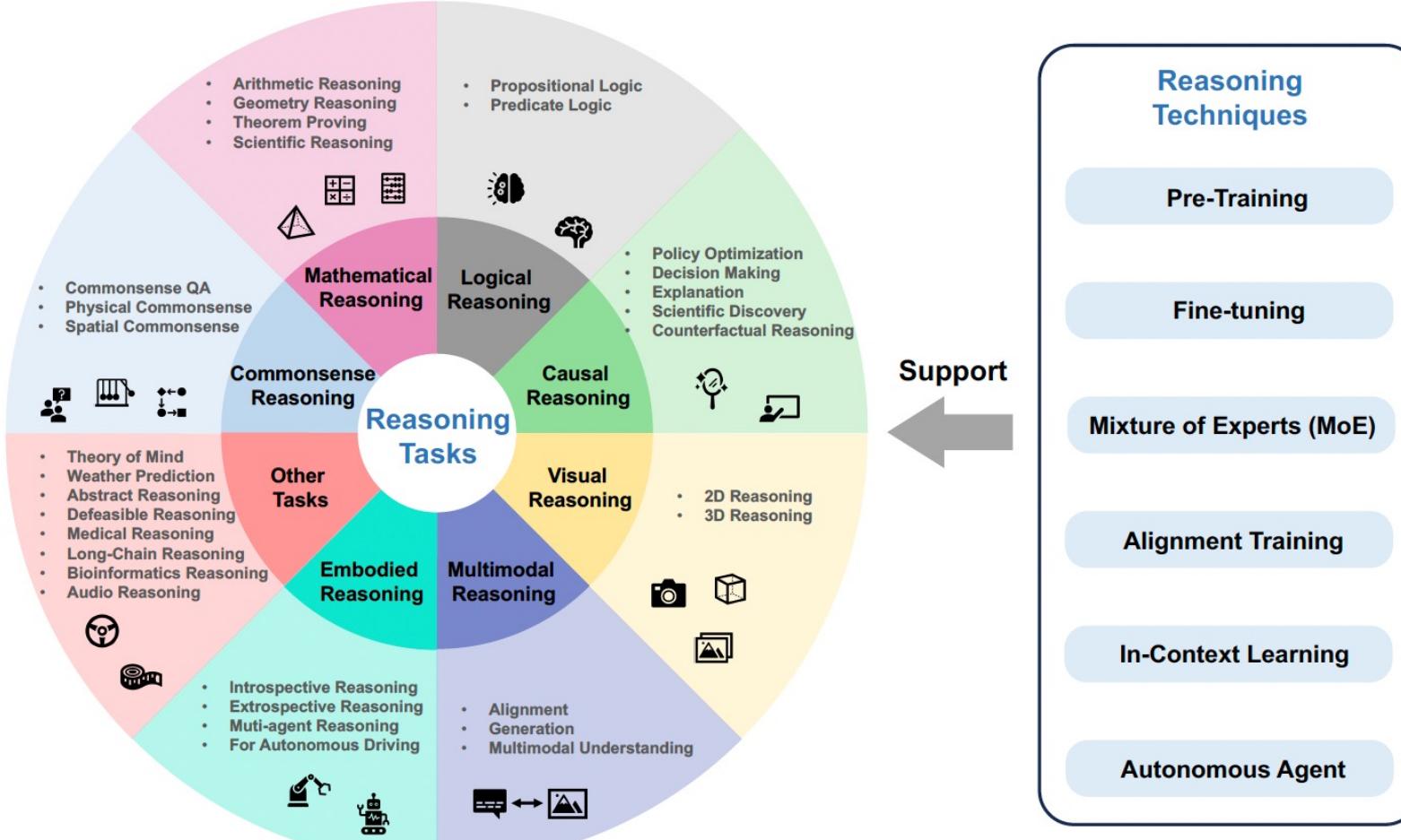
Chain of Thought - Reasoning

LLM can generate chains of thought if demonstrations of chain-of-thought reasoning are provided in the exemplars for few-shot prompting.



<https://arxiv.org/pdf/2201.11903>

There are many different Reasoning Tasks in LLM Usage



Source: A Survey of Reasoning with Foundation Models: <https://arxiv.org/pdf/2312.11562>

Reasoning encompasses cognitive processes and logical thinking to

- analyze information,
- make deductions,
- draw conclusions, and
- formulate coherent arguments.

i) Common Sense Reasoning

Context	Lee found the Northeast to be way too cold. Lee decided to move to Florida.
Question	How would you describe Lee?
Answers	a) happy b) likes cold weather c) likes the heat

ii) Math World Problem

Problem	A farmer has 3 types of fruits in his garden: apples, oranges, and pears. He has twice as many apples as oranges and three times as many pears as apples. If he has 24 oranges, how many pieces of fruit does he have in total?
Expression	$x = 24 \times 2 + 24 \times 3 \times 2 + 24$
Solution	216

Source: LARGE LANGUAGE MODELS AS ANALOGICAL REASONERS: <https://arxiv.org/pdf/2310.01714>

Reasoning can be observed in diverse domains, such as

- scientific inquiry,
- problem-solving,
- decision-making, and
- everyday discourse.

Its fundamental purpose is to enable individuals to connect pieces of information, evaluate relationships, and arrive at informed judgments or solutions.

iii) Types of Reasoning

	Example
Fact1	This animal is a robin.
Rule	All robins are birds.
Fact2	This animal is a bird.
Reasoning Type	Representation
Deduction	(Fact1 + Rule → Fact2)
Abduction	(Fact1 + Rule ← Fact2)
Induction	(Fact1 + Fact2 → Rule)

black text represents the given knowledge, while the red text represents the inferred knowledge

Further Reading



- [DLG] „Generative Deep Learning“, David Foster, 2023, O'Reilly
- [GAN] „NIPS 2016 Tutorial: Generative Adversarial Networks“, Ian Goodfellow 2027, <https://arxiv.org/pdf/1701.00160>
- [STA] „Stanford CS236, Deep Generative Models“, 2023, <https://deepgenerativemodels.github.io>
- [COR] „Cornell CS 6785: Deep Generative Models“, 2024, <https://kuleshov-group.github.io/dgm-website/>
- [REV] Roberto Gozalo-Brizuela, Eduardo C. Garrido-Merchán „ChatGPT is not all you need. A State of the Art Review of large Generative AI models“, <https://arxiv.org/pdf/2301.04655>
- [PET] „The Prompt Report: A Systematic Survey of Prompting Techniques“, 2024, <https://arxiv.org/pdf/2406.06608>
- [2020-01-23] „Scaling Laws for Neural Language Models“, <https://arxiv.org/pdf/2001.08361.pdf>
- [CT] „Scaling Laws and Emergent Properties“, Clément Thiriet, 30.04.2023, <https://cthiriet.com/blog/scaling-laws>

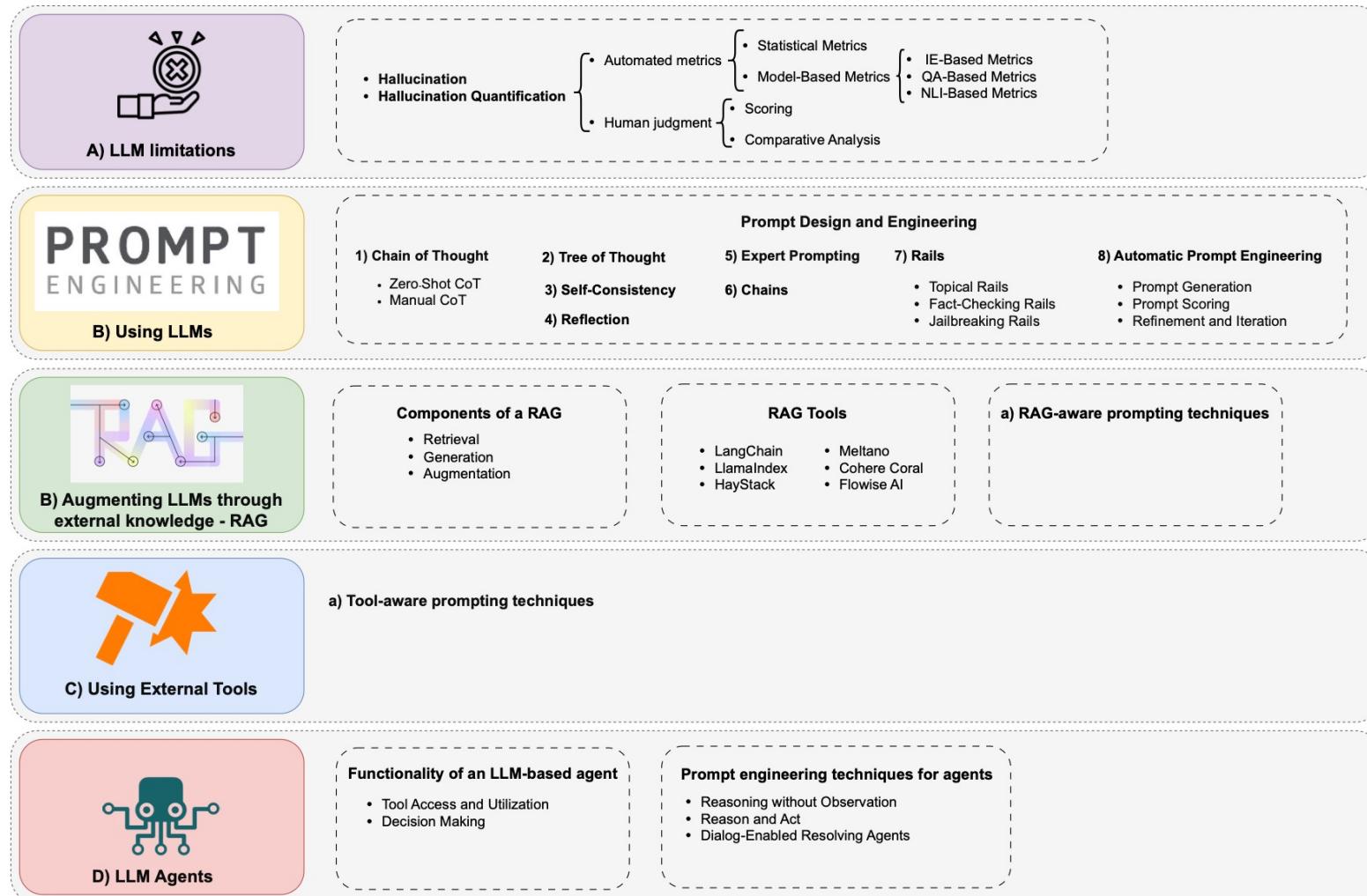


Dominik Neumann
Hochschule Reutlingen, Alteburgstraße 150, 72762 Reutlingen
www.reutlingen-university.de
T. +49 172 9861157
dominik.neumann@reutlingen-university.de
dominik.neumann@exxeta.com

Ausblick



LLM Augmentation



<https://arxiv.org/pdf/2402.06196.pdf> - Fig36

Why LLM Augmentation

LLM Challenges

Challenge	
Hallucination	Models always generate text as an extrapolation from the prompt you provided. Hallucination is the default. Sometimes the model generates text that is incorrect, nonsensical, or not real.
Attribution	Since LLMs are not databases or search engines, they would not cite where their response is based on. We have neither traceability nor explainability.
Staleness	Model contain incomplete, outdated, or wrong knowledge due to outdated training data.
Generalization	If we have use cases with non-publicly accessible knowledge, then the knowledge is not part of the chosen LLM. We want to retrain the LLM or enhance it with the knowledge.
Revision	How can incorrect knowledge or facts be replaced in an LLM?

Model Limitation

Despite the successes achieved through scaling, a language model is limited in its area of application.

1. Models are limited in what they know about the world.
2. Models are limited in kind of tasks they can solve
3. And, in addition, models are hard to adapt



How to deal with challenges

