UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Large Language Models

Lifecycle of LLMs

Rodrygo Santos
Anisio Lacerda

ChatGPT 5

Describe a large language model in one sentence for a 2-year-old.

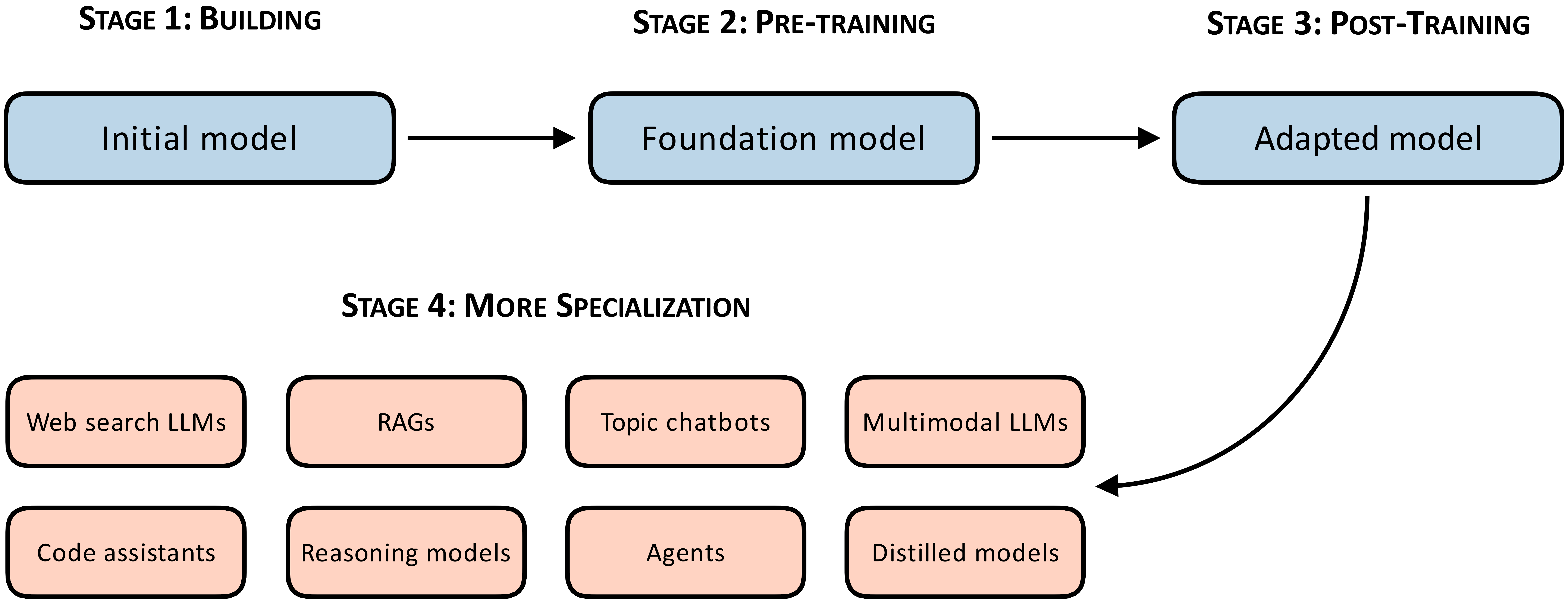
A large language model is like a very smart talking toy that listens to your words and makes up answers or stories, just like playing pretend with you.

Ask anything

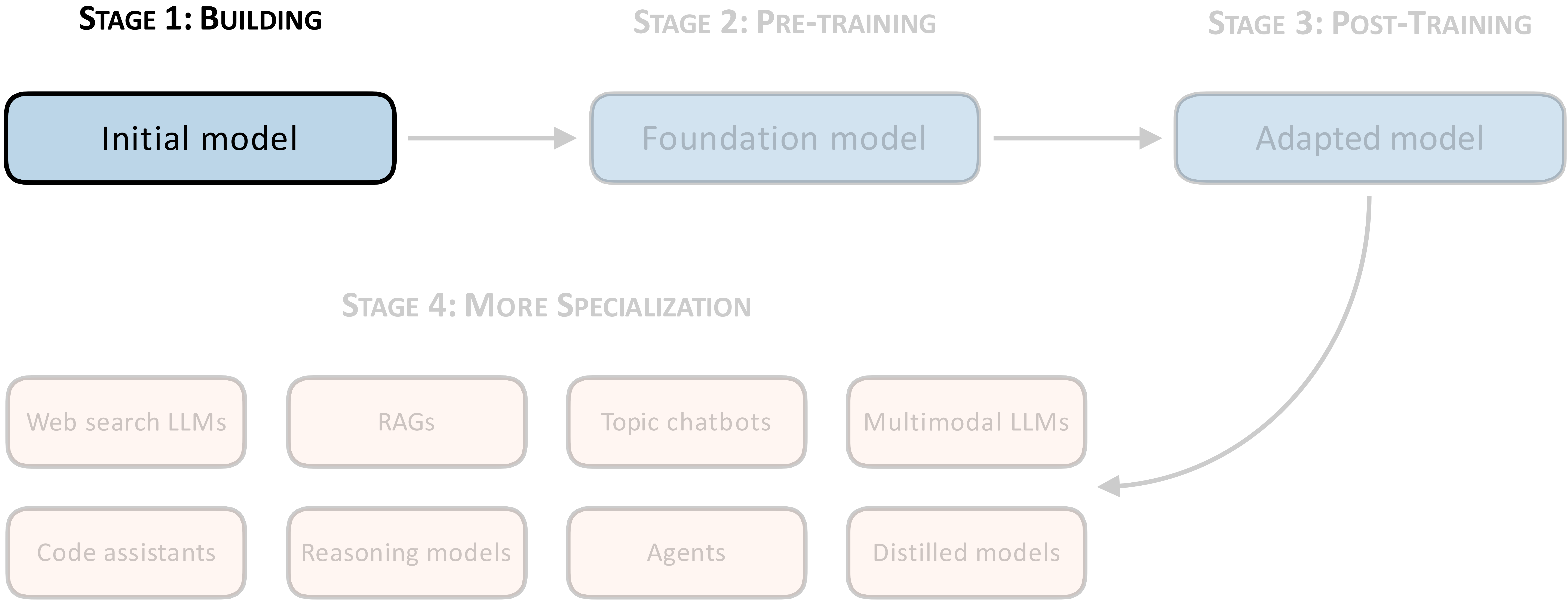
ChatGPT can make mistakes. Check important info.

```
$ ollama run gpt-oss:20b
>>> Describe a large language model in one sentence for a 2-year-old.
Thinking...
We need to describe a large language model in one sentence for a
2-year-old. Must be simple, child-friendly, probably uses familiar
words. 2-year-old can understand basic objects. So maybe: "It’s a
big talking friend that knows a lot of words and can help you with
stories and questions." But keep it simple: "It's like a
super-smart talking book that can answer questions and tell
stories." Use very simple language. "It’s a big brain in a computer
that talks and helps you." Use easy terms: "a big talking helper in
a computer." Let's craft.
...done thinking.

It’s a big talking helper that lives in a computer and can answer
questions and tell you stories.

>>> Send a message (/? for help)
```

Developing an LLM



Developing an LLM

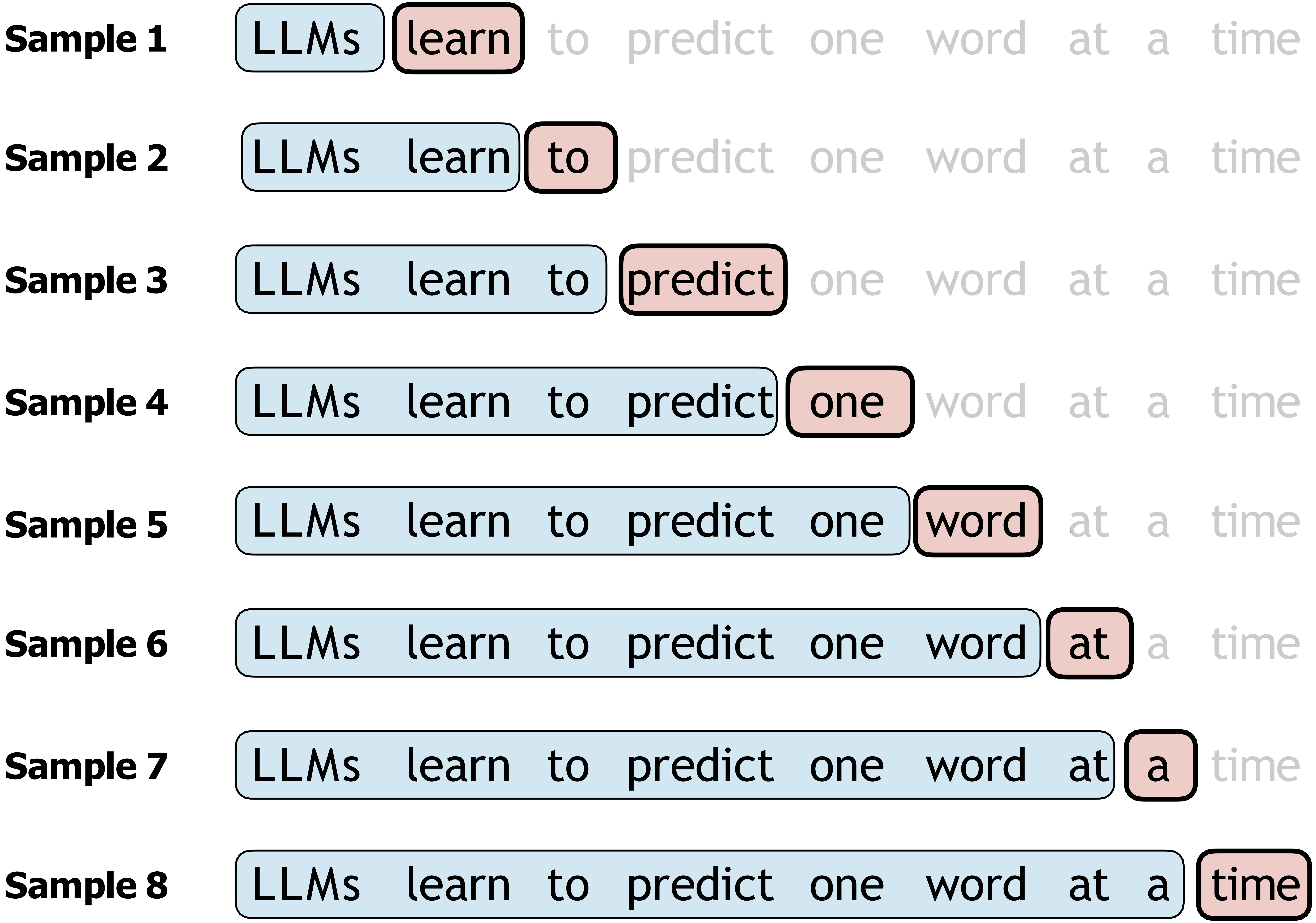


The model is simply (pre)trained to predict the next word

Sample: LLMs learn to predict one word at a time

Input x Target y

The LLM can't access words past the target


Sample 1
LLMs learn to predict one word at a time
Sample 2
LLMs learn to predict one word at a time
Sample 3
LLMs learn to predict one word at a time
Sample 4
LLMs learn to predict one word at a time
Sample 5
LLMs learn to predict one word at a time
Sample 6
LLMs learn to predict one word at a time
Sample 7
LLMs learn to predict one word at a time
Sample 8
LLMs learn to predict one word at a time

Batching

Sample text

"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade … "

Tensor containing the inputs

```
x = tensor([[ "In",      "the",      "heart", "of"   ],
            [ "the",     "city",     "stood", "the"  ],
            [ "old",     "library",  ",",     "a"    ],
            [ …                                      ]])
```

(Common input lengths are >1024)

Batching

Sample text

"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade … "

Tensor containing the inputs

```
x = tensor([[ "In",      “the",      "heart", "of"   ],
            [ "the",     "city",     "stood", "the"  ],
            [ "old",     "library",  ",",     "a"    ],
            [ …                                      ]])
```

(Common input lengths are >1024)

Batching

Sample text

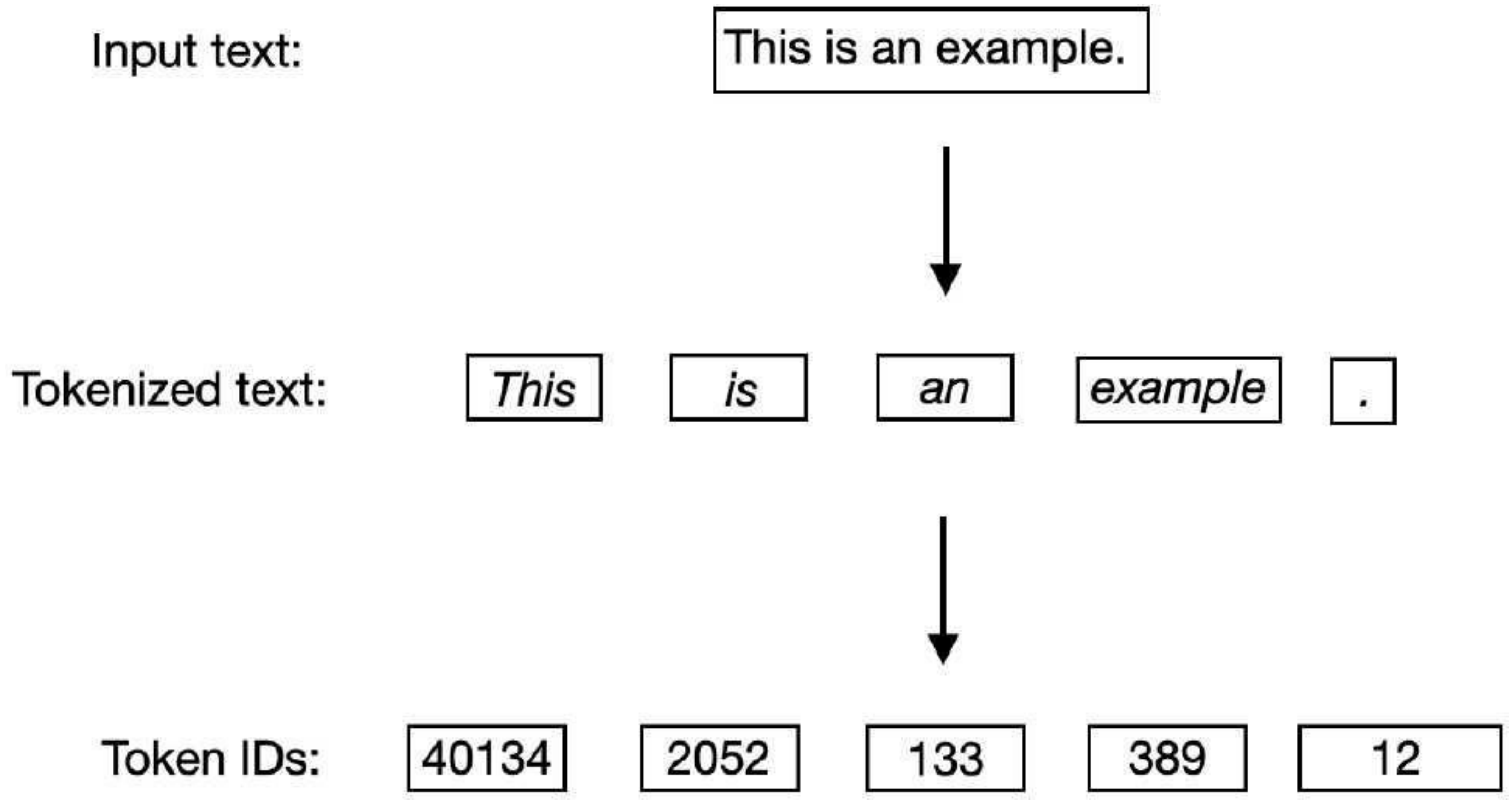
"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade … "

Tensor containing the inputs

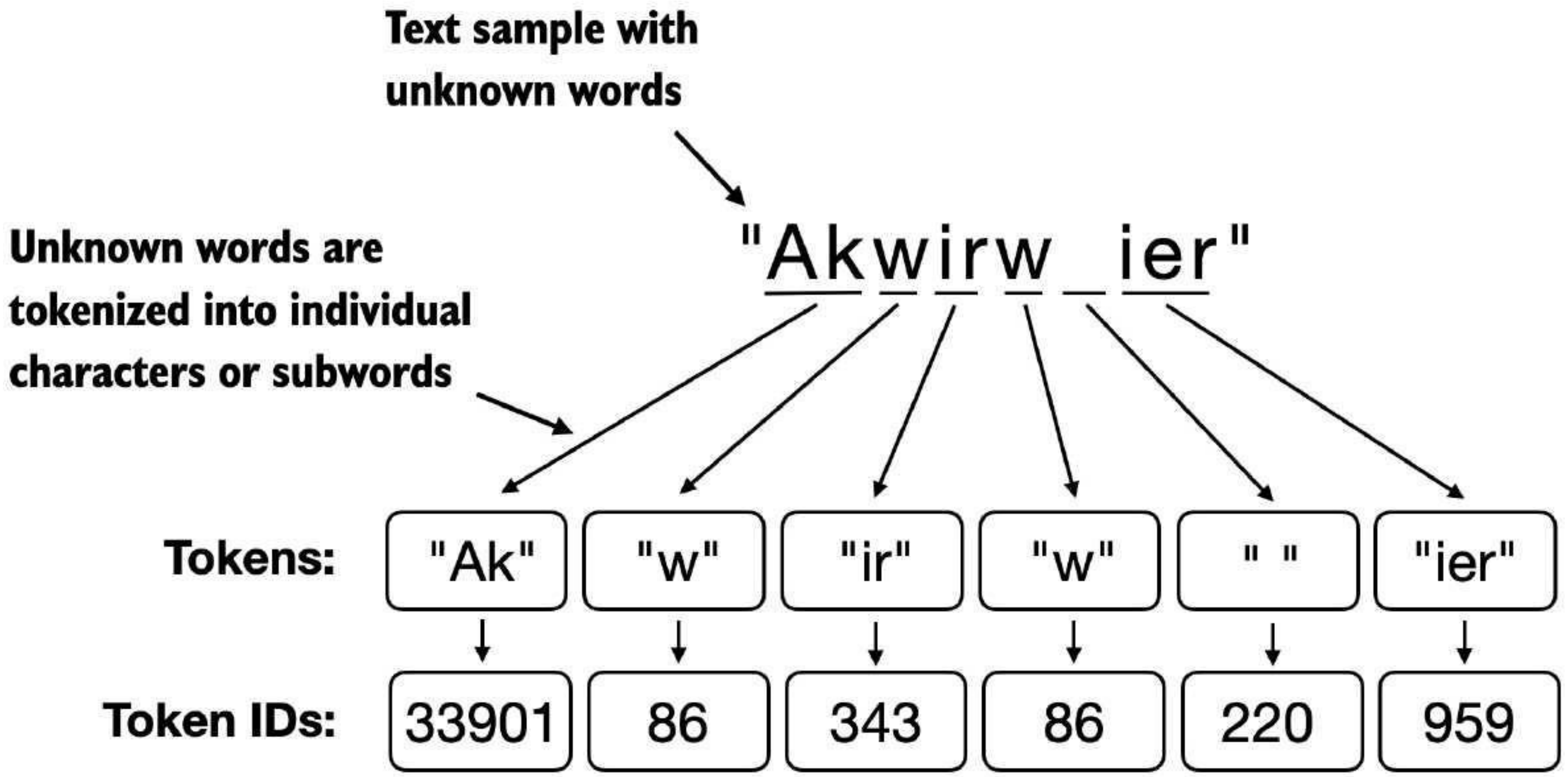
```
x = tensor([[  "In",       “the",      "heart",  "of"     ],
            [  "the",      "city",     "stood",  "the"    ],
            [  "old",      "library",  ",",      "a"      ],
            [  …                                          ]])
```

(Common input lengths are >1024)

There’s one more thing: tokenization



Subword-level tokenization



GPT-3 was trained on 0.5T tokens

Dataset	Quantity (tokens)	Weight in Training Mix	Epochs Elapsed when Training for 300B Tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Language Models are Few-Shot Learners (2020), https://arxiv.org/abs/2005.14165

Llama 1 was trained on 1.4T tokens

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

LLaMA: Open and Efficient Foundation Language Models (2023), https://arxiv.org/abs/2302.13971

Llama 2 was trained on 2T tokens

“Our training corpus includes a new mix of data from publicly available sources, which does not include data from Meta’s products or services. We made an effort to remove data from certain sites known to contain a high volume of personal information about private individuals. We trained on 2 trillion tokens of data as this provides a good performance–cost trade-off, up-sampling the most factual sources in an effort to increase knowledge and dampen hallucinations.”

Llama 2: Open Foundation and Fine-Tuned Chat Models (2023), https://arxiv.org/abs/2307.09288

Llama 3 was trained on 15T tokens

"To train the best language model, the curation of a large, high-quality training dataset is paramount. In line with our design principles, we invested heavily in pretraining data. Llama 3 is pretrained on over 15T tokens that were all collected from publicly available sources."

Introducing Meta Llama 3: The most capable openly available LLM to date (2024), https://ai.meta.com/blog/meta-llama-3/

Llama 4 was trained on 30T tokens

"Additionally, we focus on efficient model training by using FP8 precision, without sacrificing quality and ensuring high model FLOPs utilization—while pre-training our Llama 4 Behemoth model using FP8 and 32K GPUs, we achieved 390 TFLOPs/GPU. The overall data mixture for training consisted of more than 30 trillion tokens, which is more than double the Llama 3 pre-training mixture and includes diverse text, image, and video datasets."

The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation (2025), https://ai.meta.com/blog/llama-4-multimodal-intelligence/

Quantity vs quality

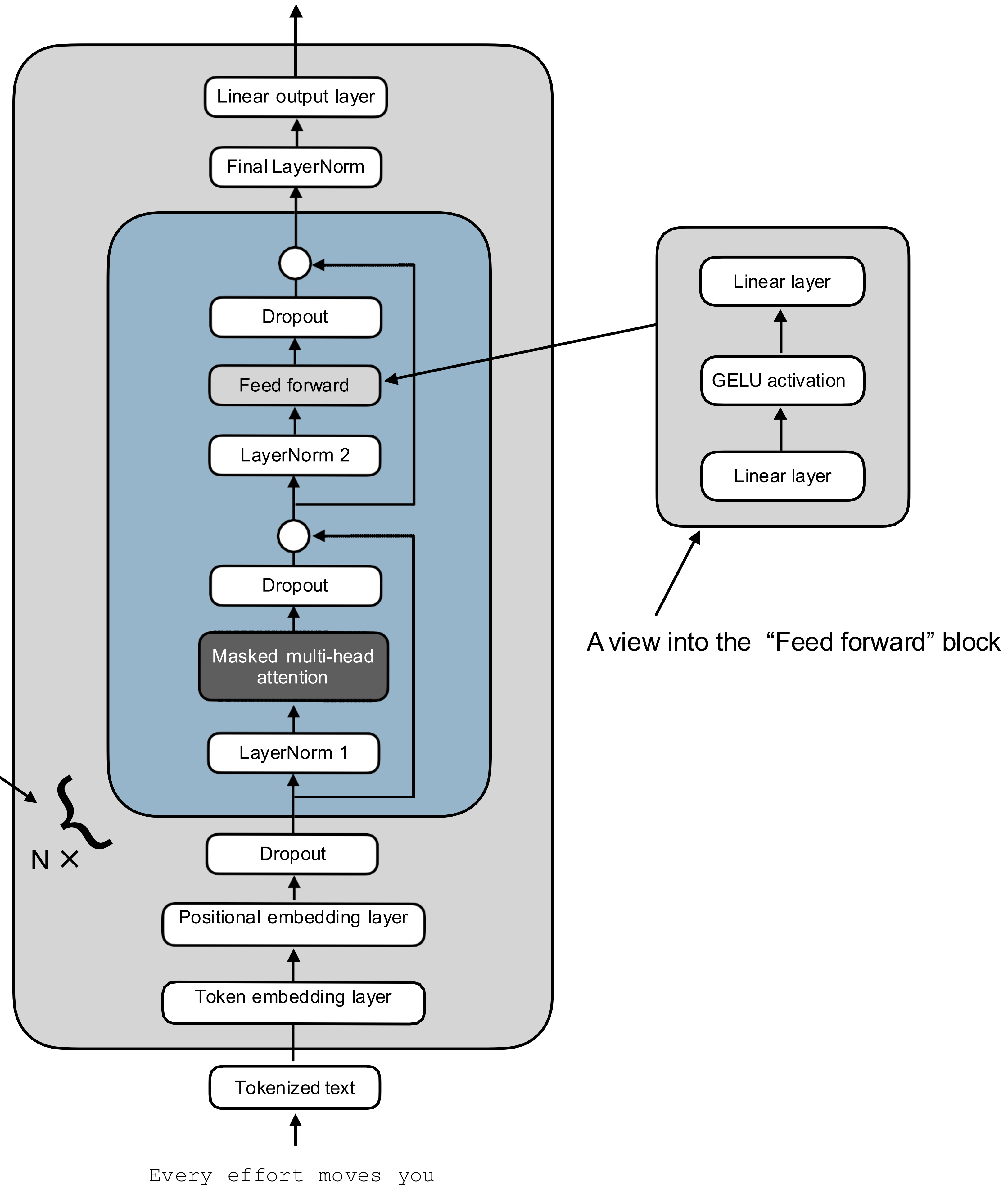
“we mainly focus on the quality of data for a given scale. We try to calibrate the training data to be closer to the “data optimal” regime for small models. In particular, we filter the publicly available web data to contain the correct level of “knowledge” and keep more web pages that could potentially improve the “reasoning ability” for the model. As an example, the result of a game in premier league in a particular day might be good training data for frontier models, but we need to remove such information to leave more model capacity for “reasoning” for the mini size models.

Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone (2024), https://arxiv.org/abs/2404.14219

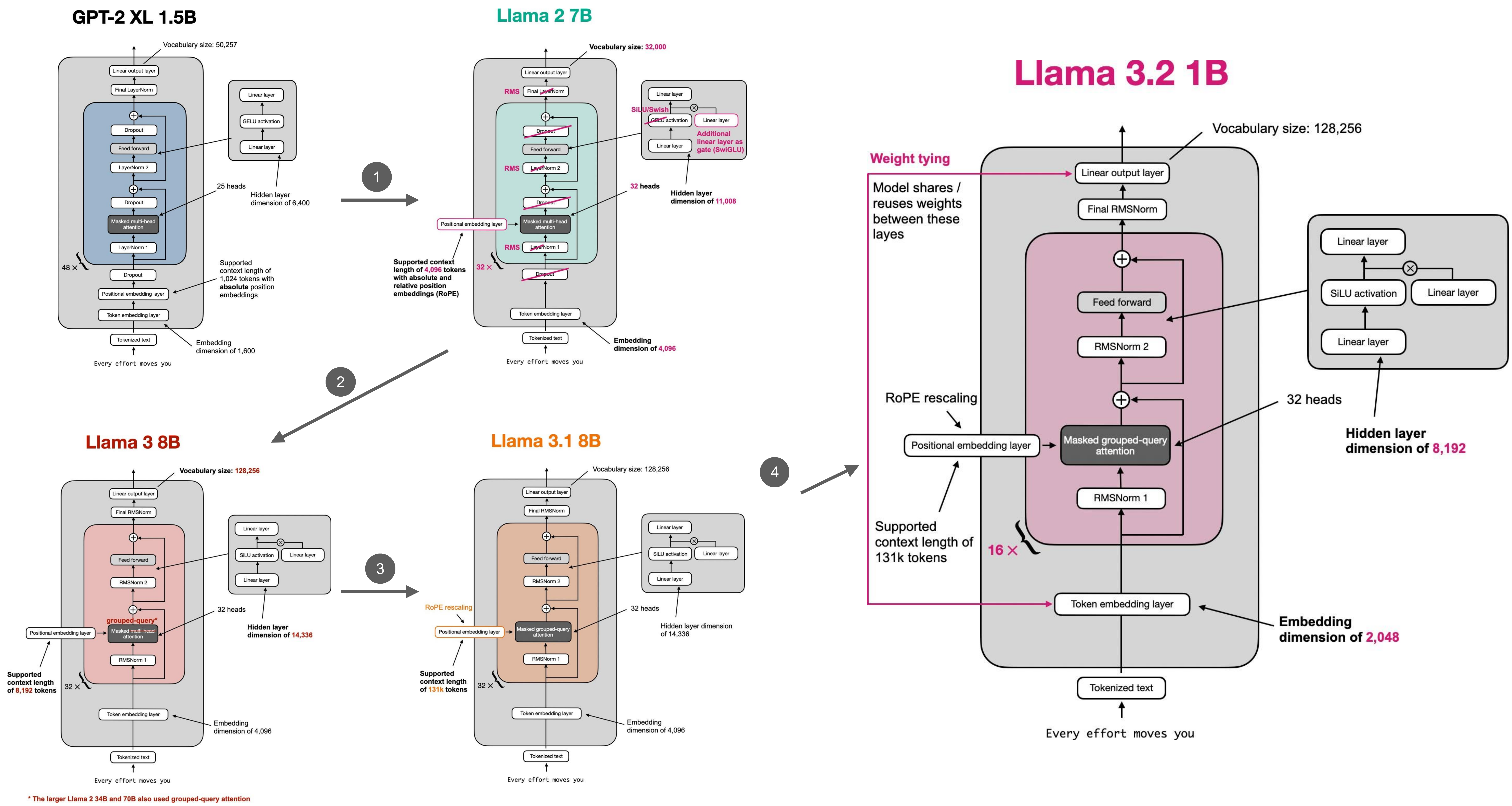
LLM architecture

Original GPT architecture (2018)

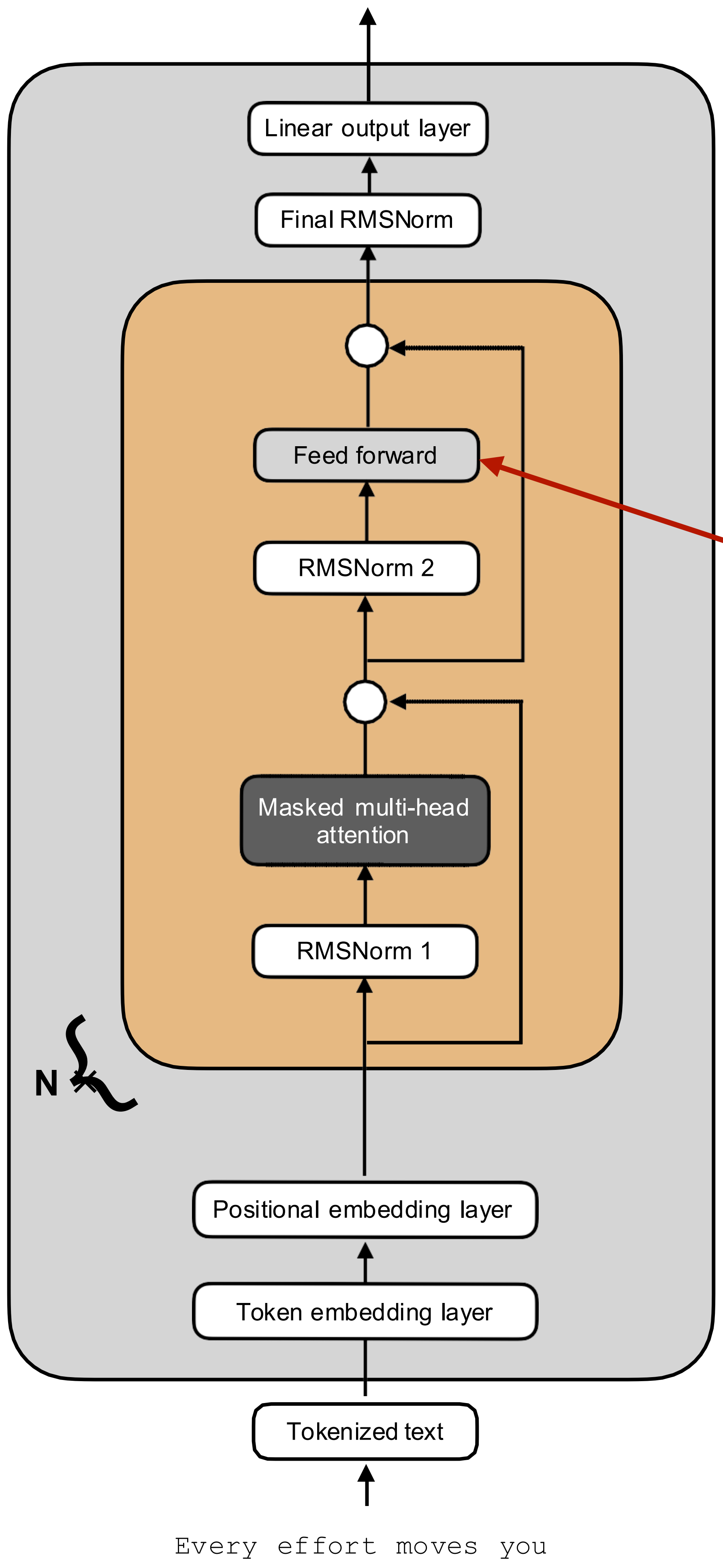
Repeat this transformer block N times



Not “that much” has changed

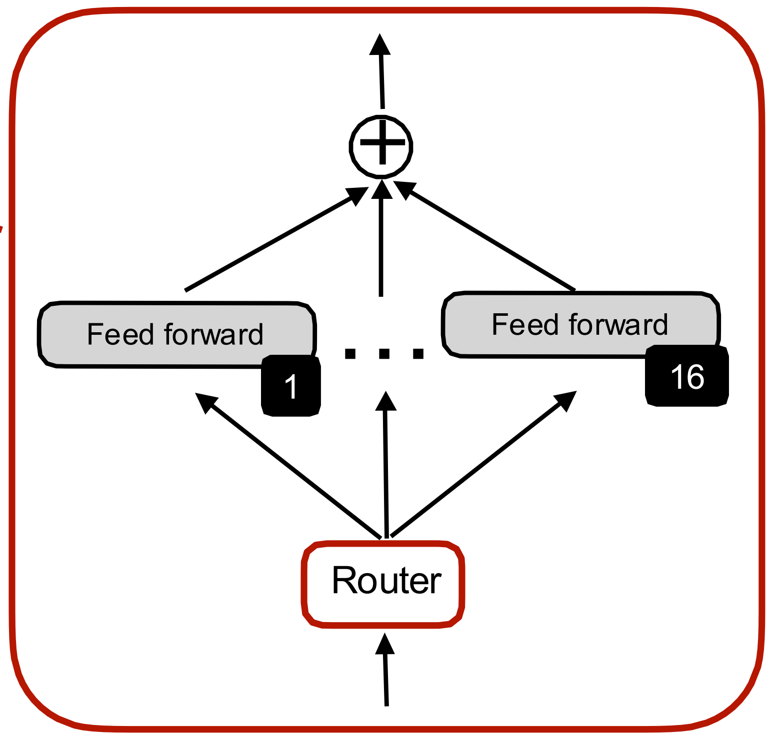


https://github.com/rasbt/LLMs-from-scratch/tree/main/ch05/07_gpt_to_llama



Llama 4 Scout (Apr 2025)

replaces the feedforward module by 16 experts:

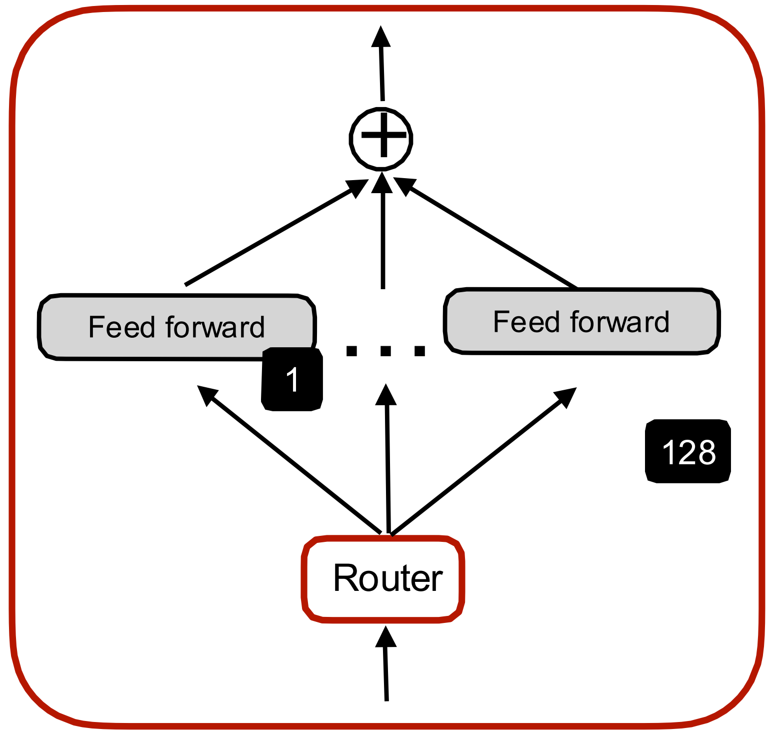


Resource savings:

- Model size is **109B**
- but only **2 experts?** are utilized at a time
- only **17B** parameters are active at a time

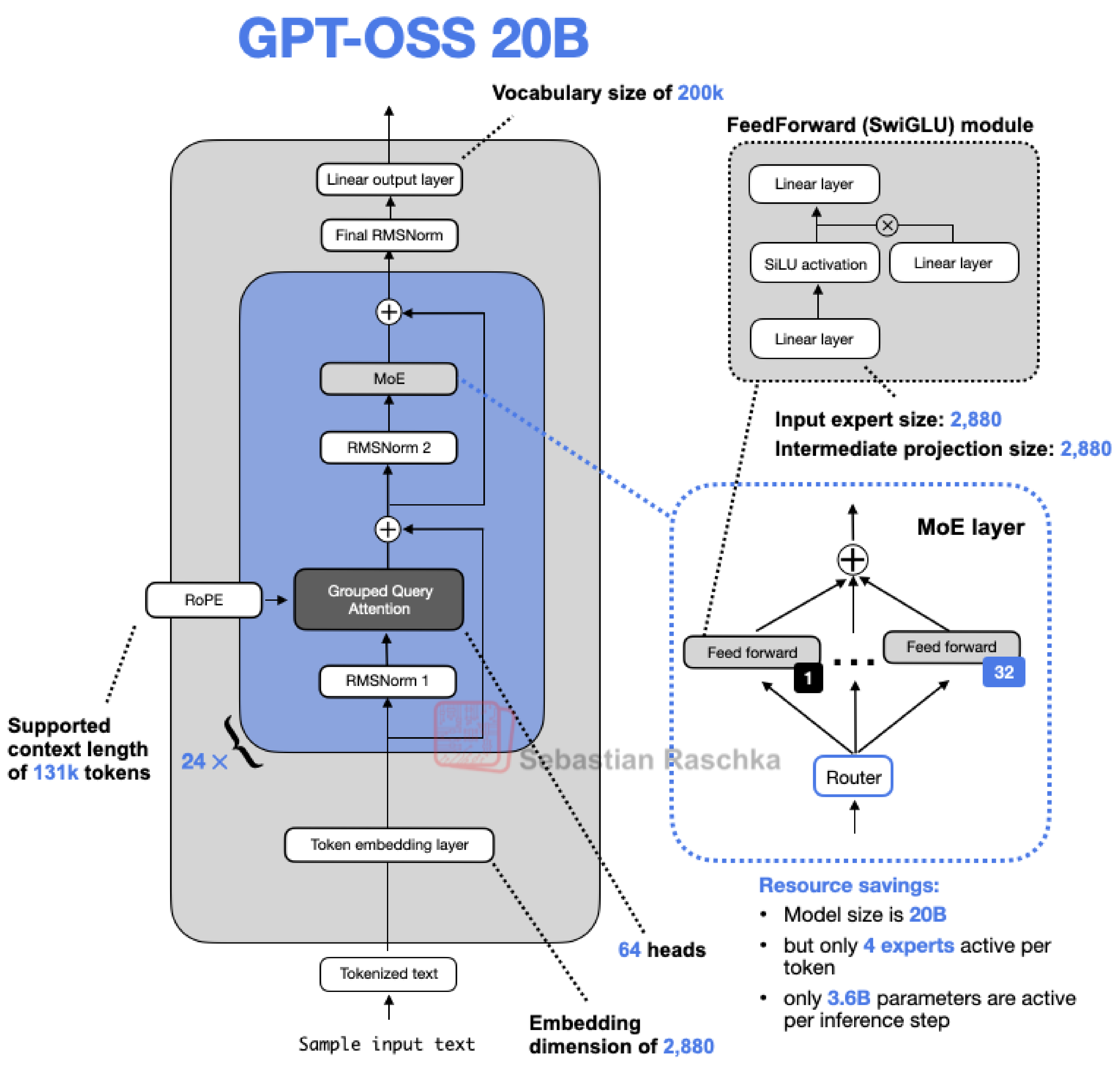
Llama 4 Maverick (Apr 2025)

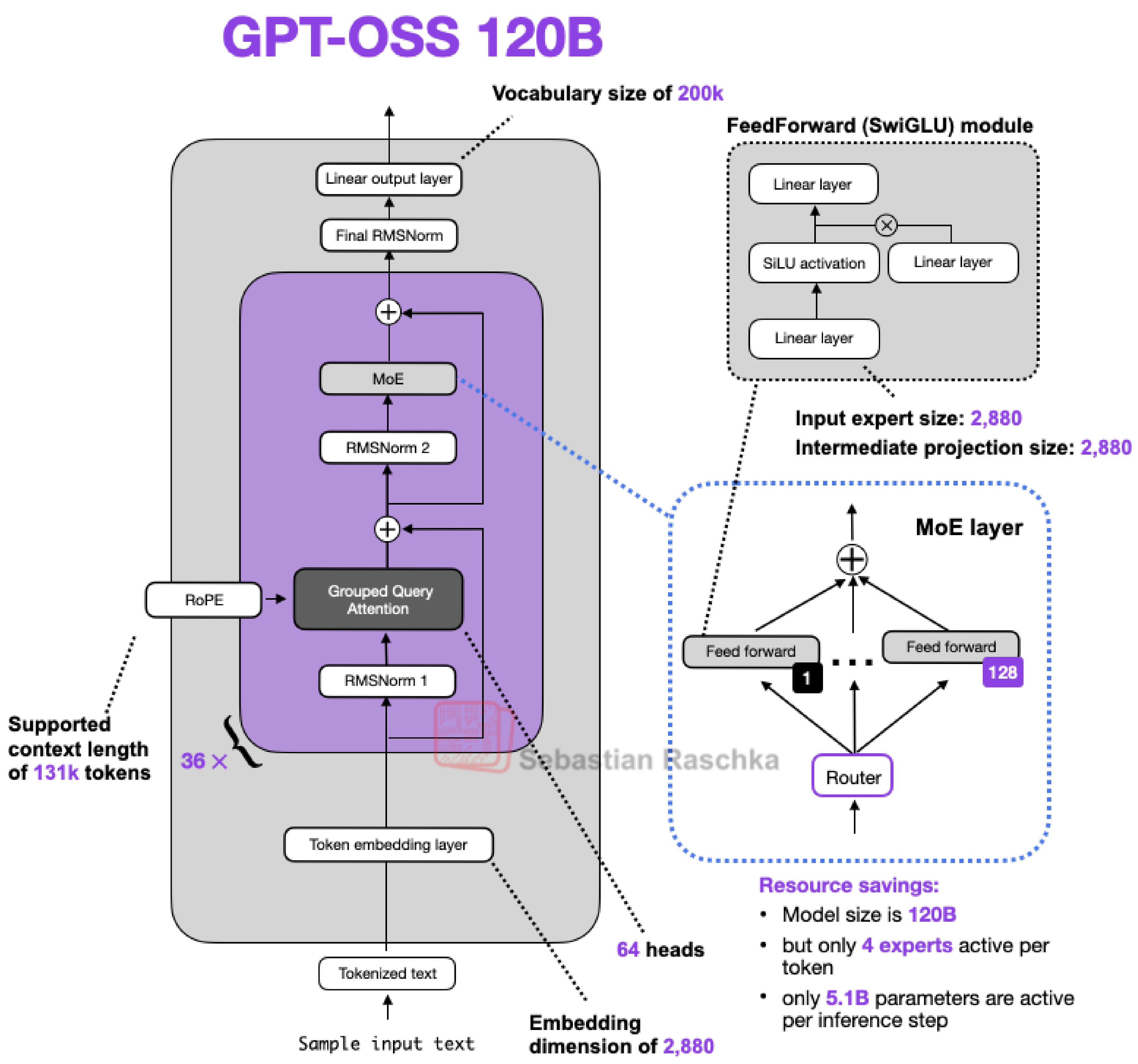
replaces the feedforward module by 128 experts:



Resource savings:

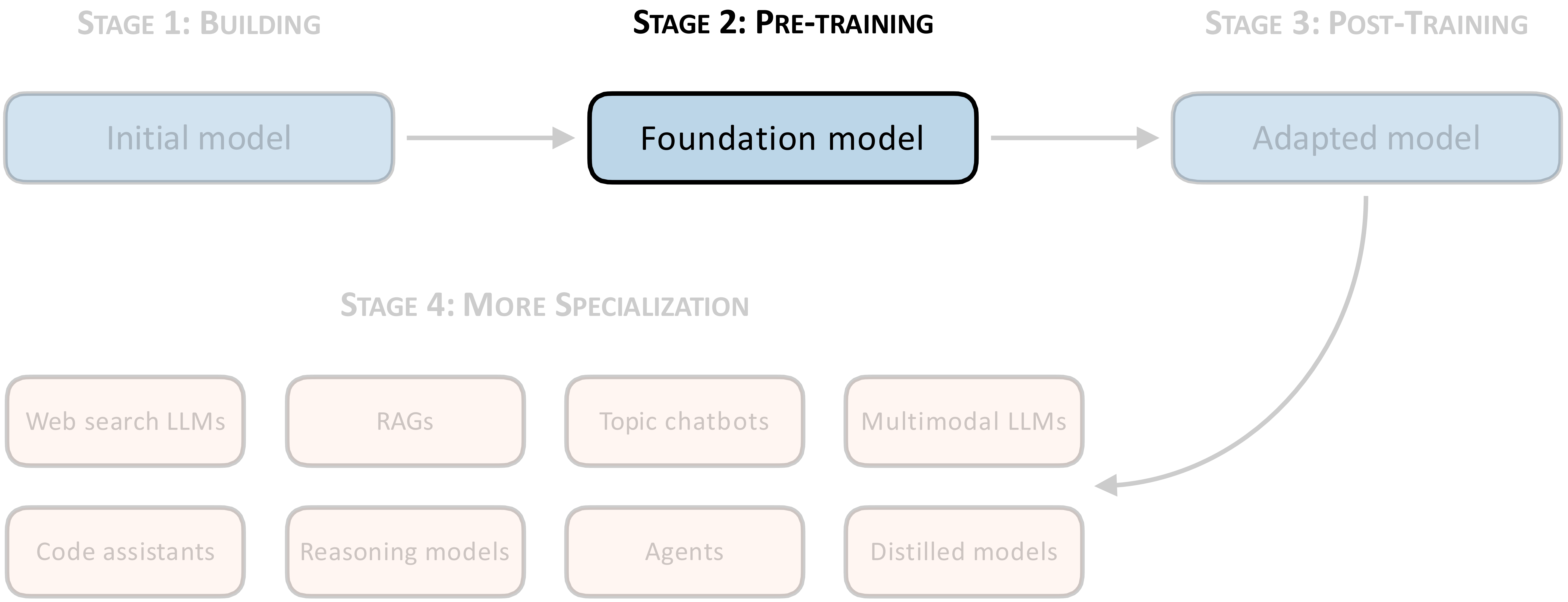
- Model size is **400B**
- but only **2 experts?** are utilized at a time
- only **17B** parameters are active at a time


GPT-OSS 20B
Vocabulary size of 200k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Grouped Query Attention
RMSNorm 1
24 ×
Supported context length of 131k tokens
Token embedding layer
Tokenized text
Sample input text
64 heads
Embedding dimension of 2,880
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Input expert size: 2,880
Intermediate projection size: 2,880
MoE layer
Feed forward
1
Feed forward
32
Router
Sebastian Raschka
Resource savings:
• Model size is 20B
• but only 4 experts active per token
• only 3.6B parameters are active per inference step

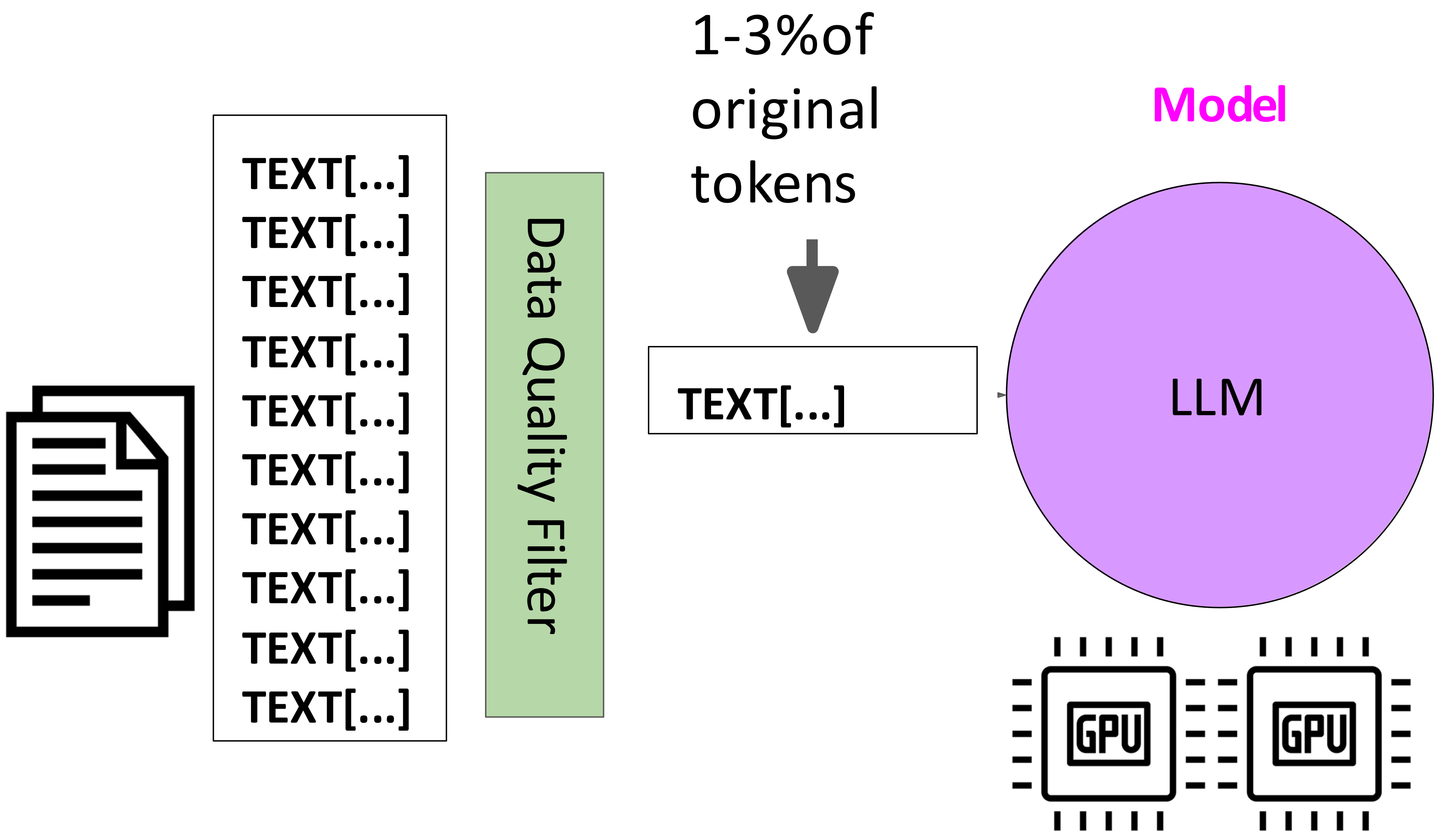

GPT-OSS 120B
Vocabulary size of 200k
Linear output layer
Final RMSNorm
MoE
RMSNorm 2
RoPE
Grouped Query Attention
RMSNorm 1
36 ×
Supported context length of 131k tokens
Token embedding layer
Tokenized text
Sample input text
64 heads
Embedding dimension of 2,880
FeedForward (SwiGLU) module
Linear layer
SiLU activation
Linear layer
Linear layer
Input expert size: 2,880
Intermediate projection size: 2,880
MoE layer
Feed forward
1
Feed forward
128
Router
Sebastian Raschka
Resource savings:
• Model size is 120B
• but only 4 experts active per token
• only 5.1B parameters are active per inference step

Takeaway: Not “that much” has changed

Developing an LLM



Pretraining at a high level

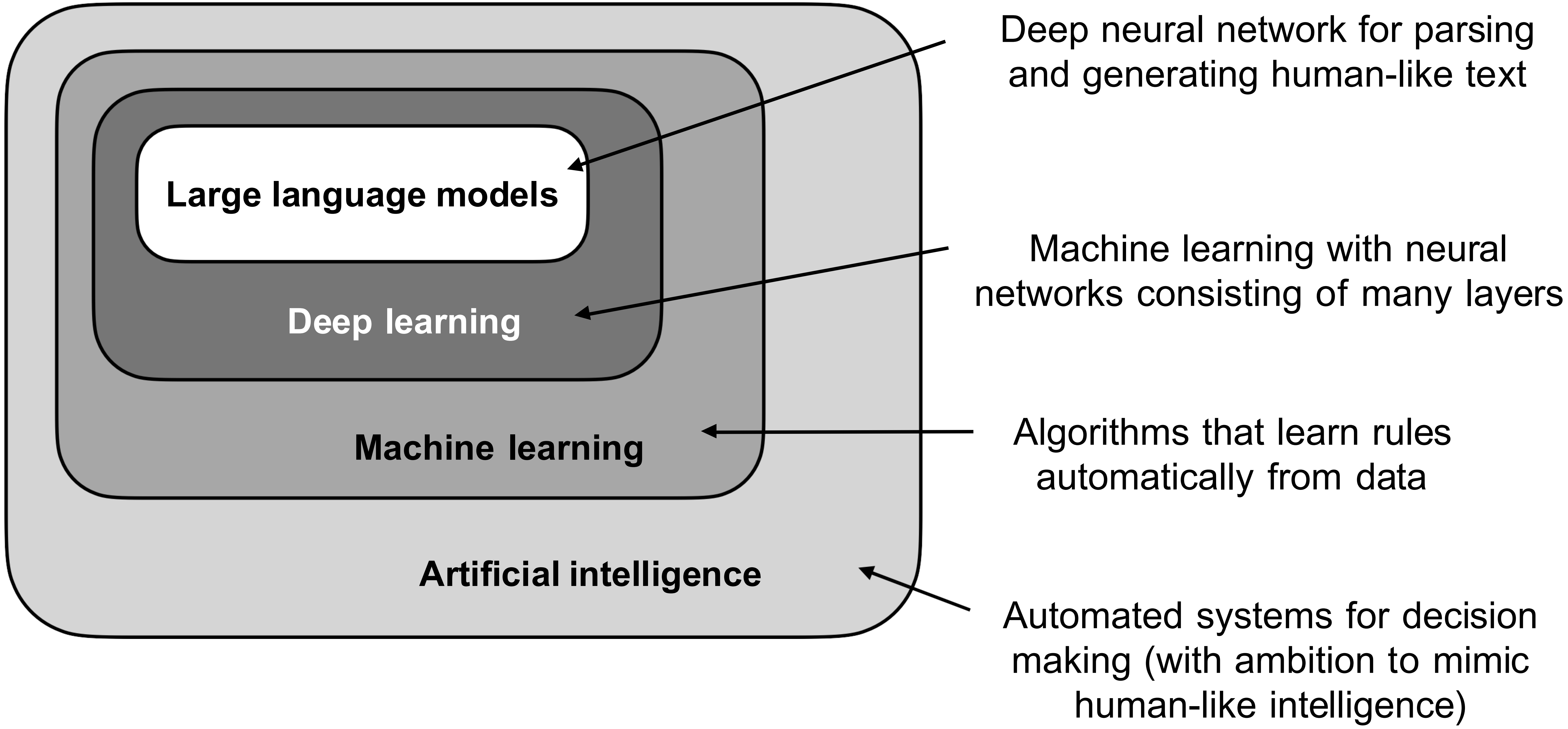


GB-TB-PB
of unstructured data

String	ID	Embedding
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

Vocabulary

LLMs are deep neural networks



Pretty standard deep learning training loop

Labels are the inputs shifted by +1

Sample text

"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade … "

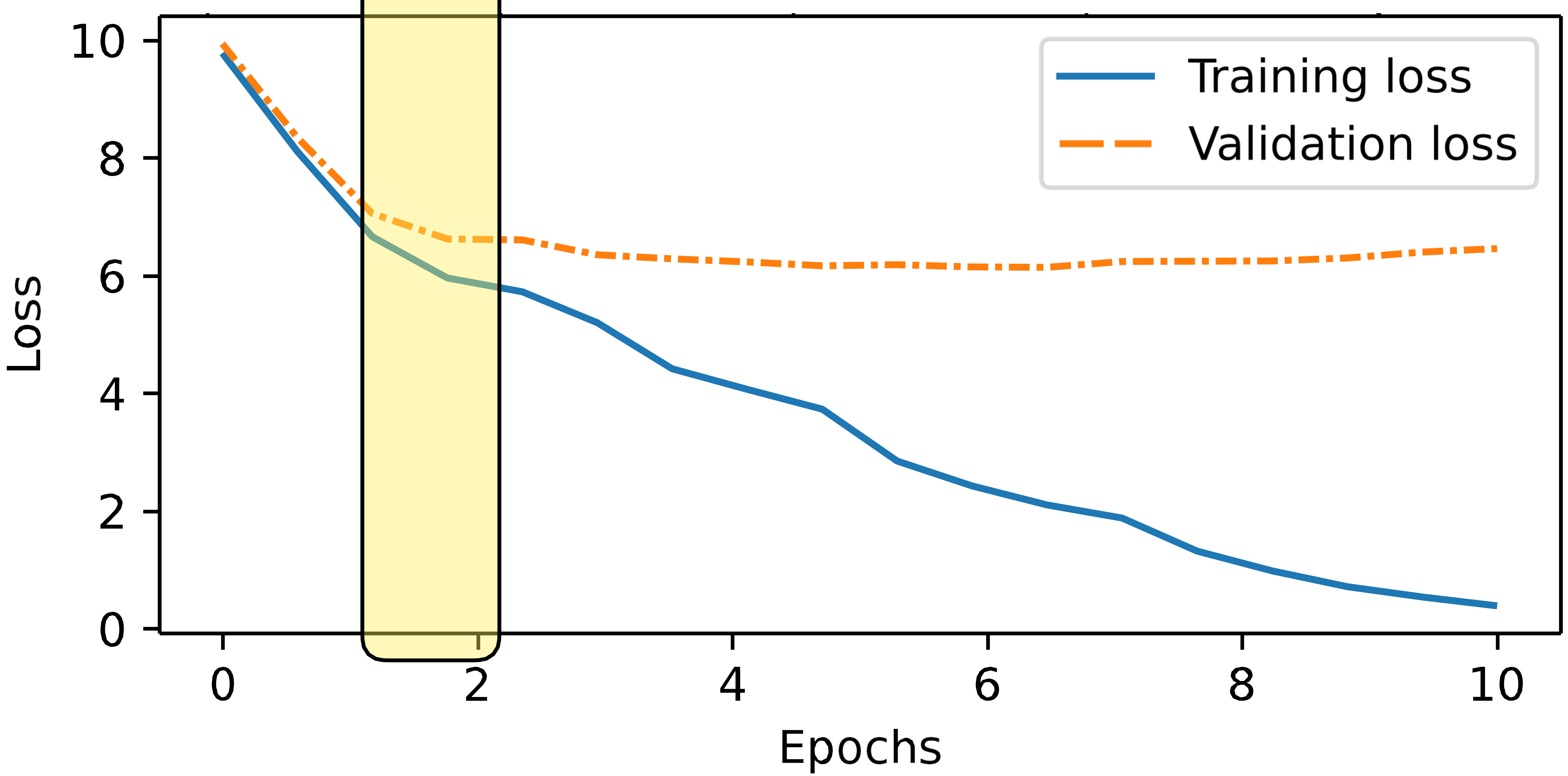
Tensor containing the inputs

```
x = tensor([[ "In",       “the",      "heart",  "of"    ],
            [ "the",      "city",     "stood",  "the"   ],
            [ "old",      "library",  ",",      "a"     ],
            [ …                                         ]])
```

Tensor containing the targets

```
y = tensor([[ "the",      "heart",    "of",     "the"   ],
            [ "city",     "stood",    "the",    "old"   ],
            [ “library",  "a",        "relic",  "from"  ],
            [ …                                         ]])
```

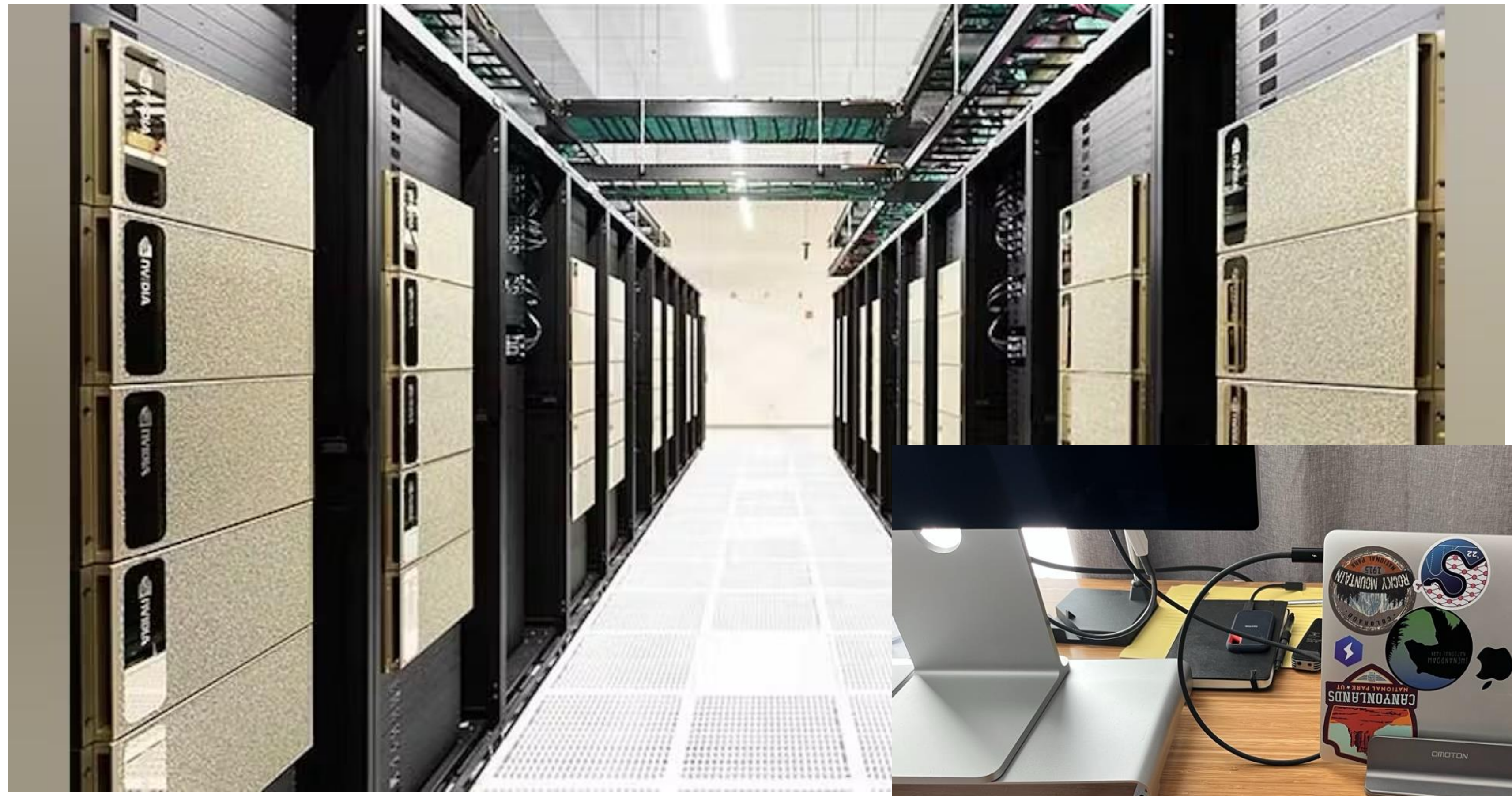
Training for ~1-2 epochs is usually a good sweet spot



“Pretty standard” training loop

~2018

Today



What makes it hard then?

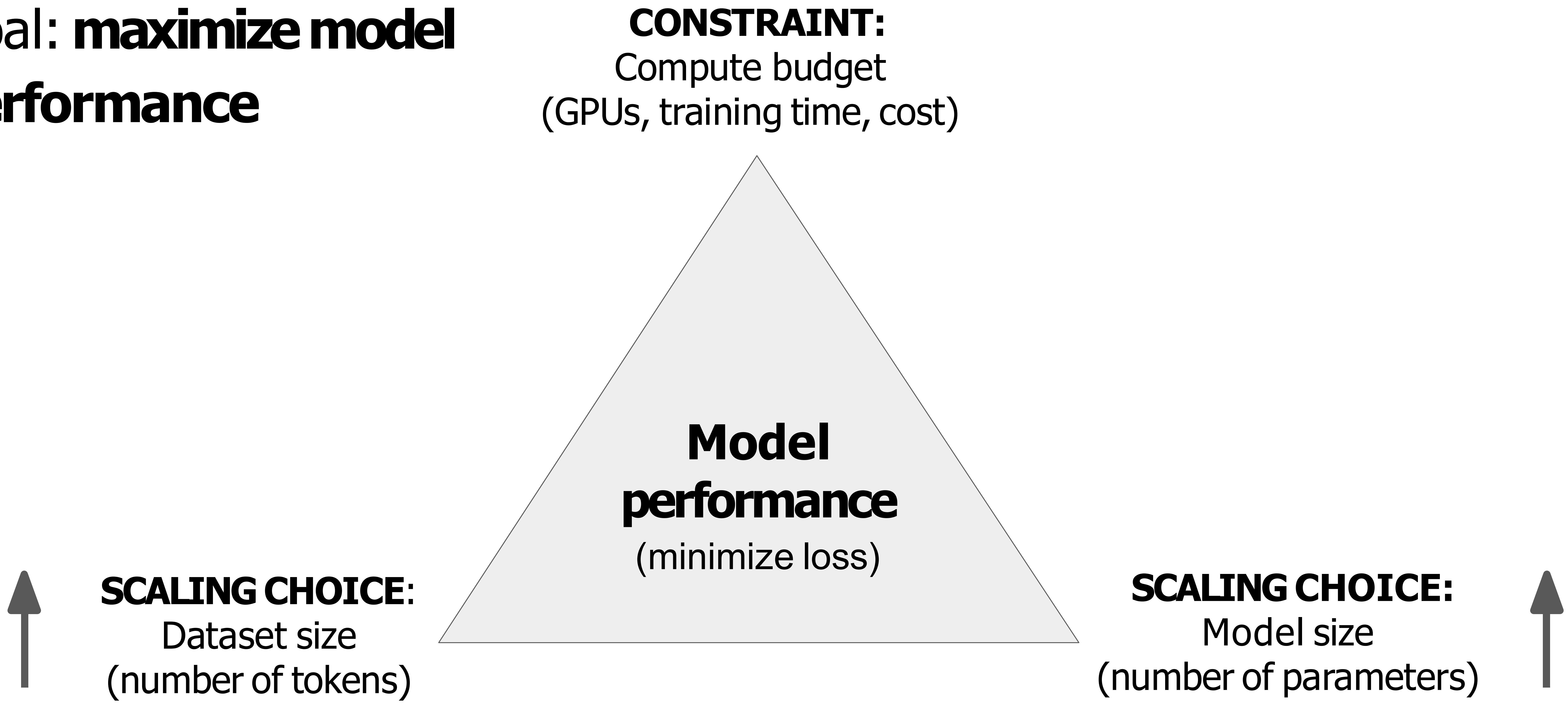
Being able to afford… compute resources

Getting access to… compute resources

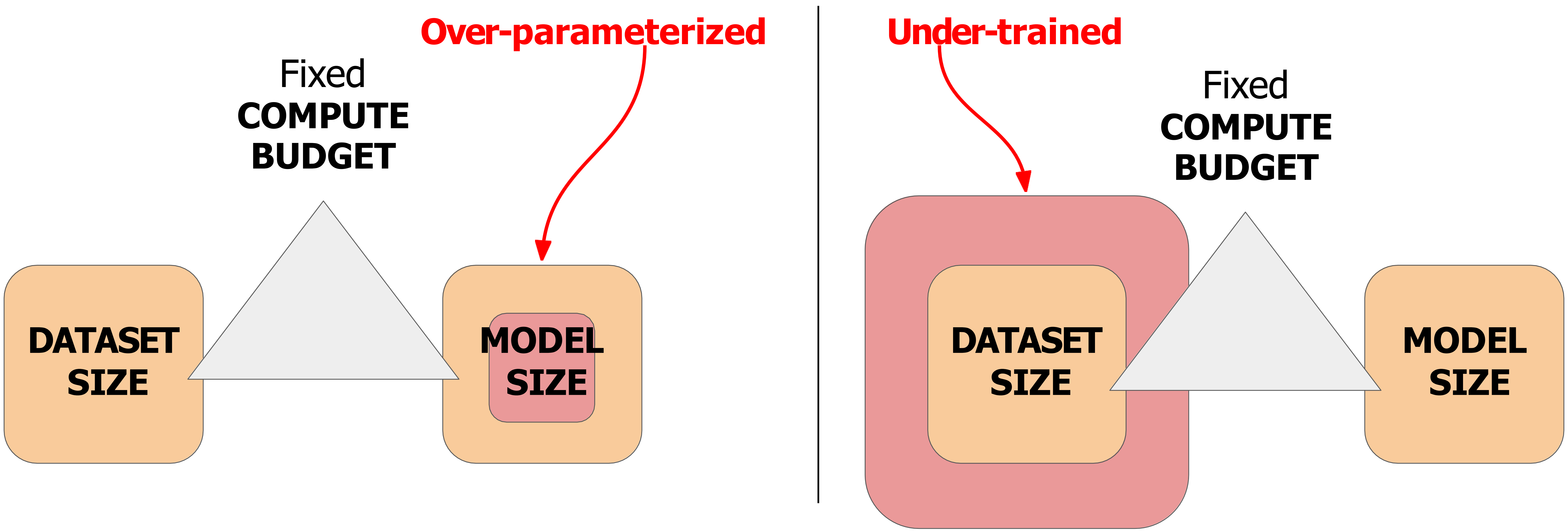
Managing… compute resources

Scaling choices

Goal: **maximize model performance**



Compute-optimal models



Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Memory needs beyond parameters

Memory needed to store the model

1B

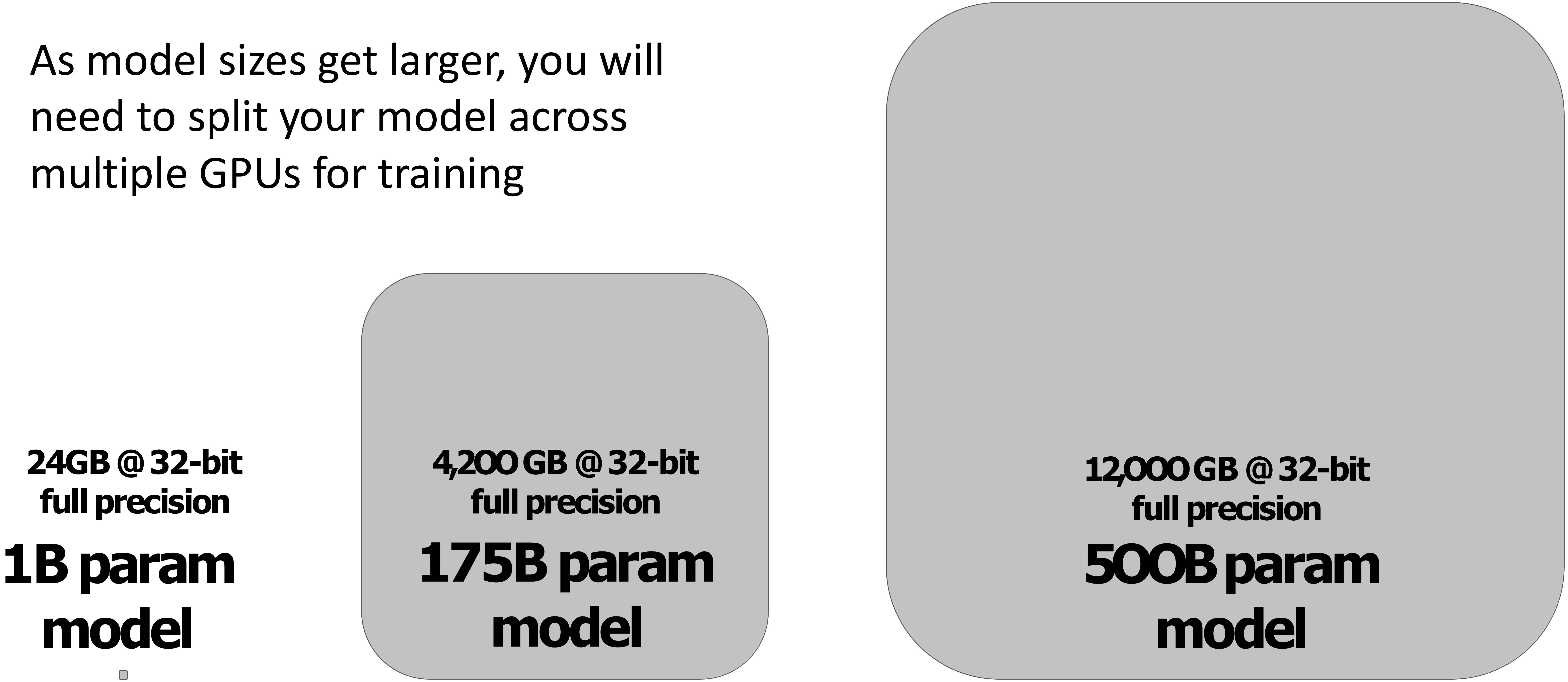
4GB @ 32-bit full precision

Memory needed to train the model

24GB @ 32-bit full precision

Memory needs beyond parameters

As model sizes get larger, you will need to split your model across multiple GPUs for training



Metric	DeepSeek V3	Llama 3.1
Parameters	671B total (37B active per token)	405B
GPU Type	NVIDIA H800	NVIDIA H100
GPU Count	2,048	Up to 16,000
Training Duration	~2 months	~2.6 months (estimated)
Tokens Processed	14.8T	15.6T
GPU Hours	2.788M	~30.8M
Training Cost	~$5.6M	~$92.4M$–123.2M (estimated)
Cost per Trillion Tokens	~$378K	~$5.93M$–7.90M

Note: Cost estimations uses an average of **$2/hour** for H800 GPUs (DeepSeek V3) and **$3/hour** for H100 GPUs (Llama 3.1) based on rental GPU prices.

Increasing price and complexity due to

larger models

larger datasets

multi-stage training

3.3.4 Reliability and Operational Challenges

The complexity and potential failure scenarios of 16K GPU training surpass those of much larger CPU clusters that we have operated. Moreover, the synchronous nature of training makes it less fault-tolerant—a single GPU failure may require a restart of the entire job. Despite these challenges, for Llama 3, we achieved higher than 90% effective training time while supporting automated cluster maintenance, such as firmware and Linux kernel upgrades (Vigraham and Leonhardi, 2024), which resulted in at least one training interruption daily. The effective training time measures the time spent on useful training over the elapsed time.

During a 54-day snapshot period of pre-training, we experienced a total of 466 job interruptions. Of these, 47 were planned interruptions due to automated maintenance operations such as firmware upgrades or operator-initiated operations like configuration or dataset updates. The remaining 419 were unexpected interruptions, which are classified in Table 5. Approximately 78% of the unexpected interruptions are attributed to confirmed hardware issues, such as GPU or host component failures, or suspected hardware-related issues like silent data corruption and unplanned individual host maintenance events. GPU issues are the largest category, accounting for 58.7% of all unexpected issues. Despite the large number of failures, significant manual intervention was required only three times during this period, with the rest of issues handled by automation.

The Llama 3 Herd of Models (31 Jul 2024), https://arxiv.org/abs/2407.21783

Should you pretrain models? Nope

Many open-weight models (0.5 B to 672 B) are available

Focus on “fine-tuning” / “post-training”

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	\$5.328M	\$0.238M	\$0.01M	\$5.576M

Table 1 | Training costs of DeepSeek-V3, assuming the rental price of H800 is \$2 per GPU hour.

DeepSeek-V3 Technical Report (27 Dec, 2024) https://arxiv.org/abs/2412.19437

Loading pretrained weights

Choose from 20+ LLMs

LitGPT has **custom, from-scratch implementations** of 20+ LLMs without layers of abstraction:

Model	Model size	Author	Reference
Llama 3	8B, 70B	Meta AI	Meta AI 2024
Llama 2	7B, 13B, 70B	Meta AI	Touvron et al. 2023
Code Llama	7B, 13B, 34B, 70B	Meta AI	Rozière et al. 2023
Mixtral MoE	8x7B	Mistral AI	Mistral AI 2023
Mistral	7B	Mistral AI	Mistral AI 2023
CodeGemma	7B	Google	Google Team, Google Deepmind
...	...	...	...

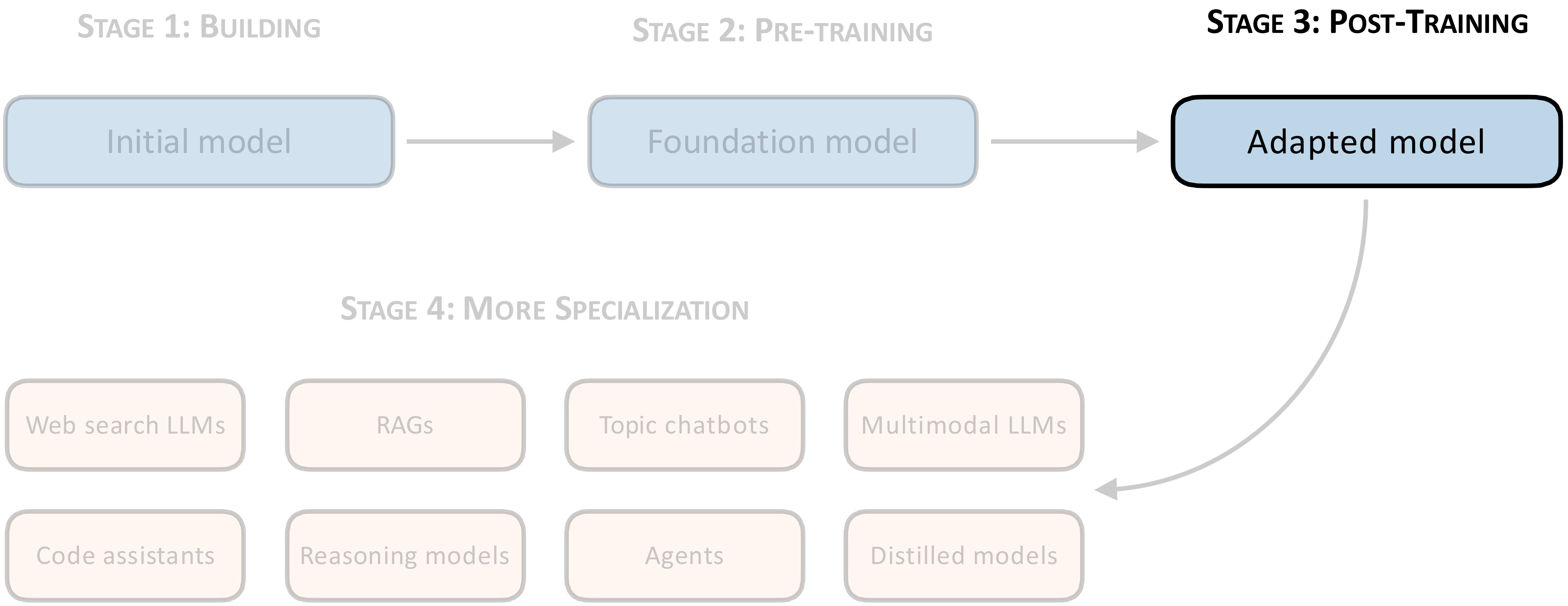
https://github.com/Lightning-AI/litgpt

LitGPT

```
# ligpt [action] [model]
litgpt  download  meta-llama/Meta-Llama-3-8B-Instruct
litgpt  chat      meta-llama/Meta-Llama-3-8B-Instruct
litgpt  finetune  meta-llama/Meta-Llama-3-8B-Instruct
litgpt  pretrain  meta-llama/Meta-Llama-3-8B-Instruct
litgpt  serve     meta-llama/Meta-Llama-3-8B-Instruct
```

https://github.com/Lightning-AI/litgpt

Developing an LLM



Instruction tuning

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

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

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

But model only sees inputs during inference

Model input

```
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.
```

Model response

Still a next-token prediction task!

Sample text

"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade …"

Tensor containing the inputs

```
x = tensor([[ "In",      “the",     "heart", "of"    ],
            [ "the",     "city",    "stood", "the"   ],
            [ "old",     "library", ",",     "a"     ],
            [ …                                      ]])
```

Tensor containing the targets

```
y = tensor([[ "the",      "heart",  "of",    "the"   ],
            [ "city",     "stood",  "the",   "old"   ],
            [ “library",  "a",      "relic", "from"  ],
            [ …                                      ]])
```

Still a next-token prediction task!

Sample text

Below is an instruction that describes a task. Write a response that …

… ### Response: \nGreat results were achieved by the team.

Tensor containing the inputs

```
x = tensor([[  "Below", "is",    "…",   "the",    "team",      "."          ],
            [                    …                                          ],
            [                    …                                          ],
            [                    …                                          ]])
```

Tensor containing the targets

```
y = tensor([[  "is",    "an",    "…",   "team",   ".",   "<|endoftext|>" ],
            [                    …                                          ],
            [                    …                                          ],
            [                    …                                          ]])
```

Preference tuning (aka alignment)

Input Prompt:
"What are the key features to look for when purchasing a new laptop?"

Answer 1: Technical Response
"When purchasing a new laptop, focus on key specifications such as the processor speed, RAM size, storage type (SSD vs. HDD), and battery life. The processor should be powerful enough for your software needs, and sufficient RAM will ensure smooth multitasking. Opt for an SSD for faster boot times and file access. Additionally, screen resolution and port types are important for connectivity and display quality."

Answer 2: User-Friendly Response
"When looking for a new laptop, think about how it fits into your daily life. Choose a lightweight model if you travel frequently, and consider a laptop with a comfortable keyboard and a responsive touchpad. Battery life is crucial if you're often on the move, so look for a model that can last a full day on a single charge. Also, make sure it has enough USB ports and possibly an HDMI port to connect with other devices easily."

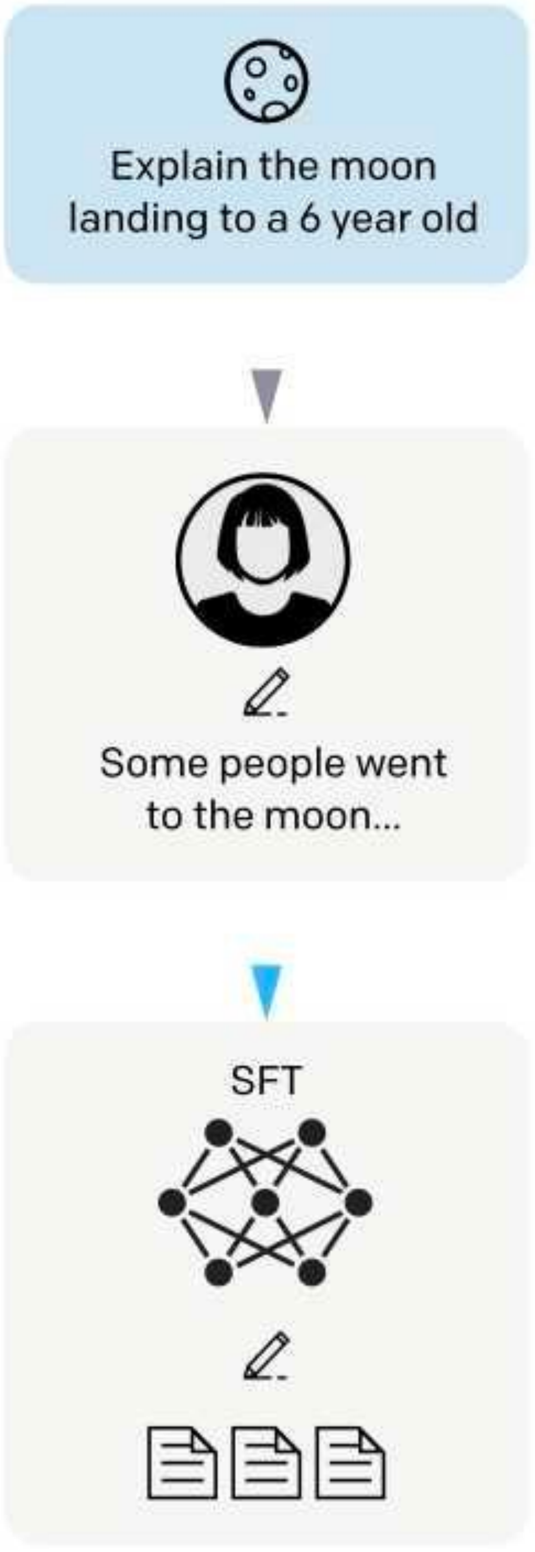
Example: alignment via RLHF

Step 1: SFT

Sample prompt

Human writes response

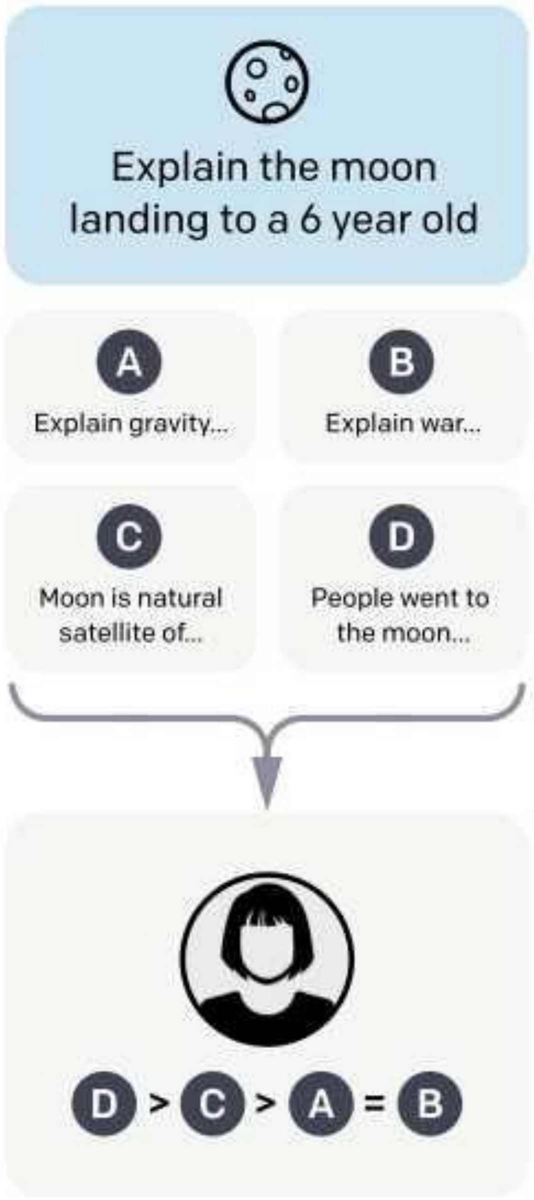
Supervised fine-tuning
of pre-trained LLM
(as discussed previously)



Time & labor intensive

Example: alignment via RLHF

Step 2: Rank responses



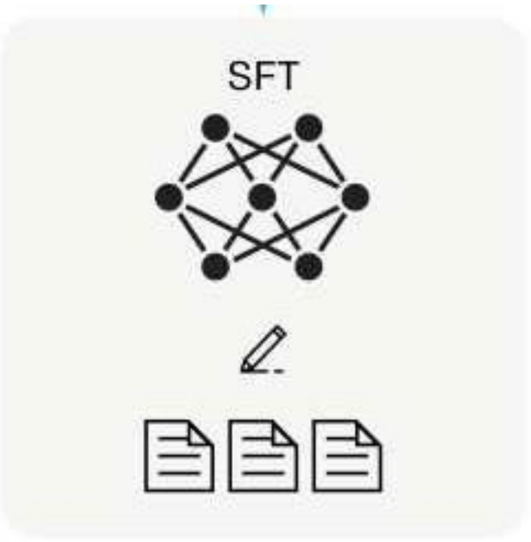
Sample prompt

Collect model responses

Human ranks responses

Time & labor intensive

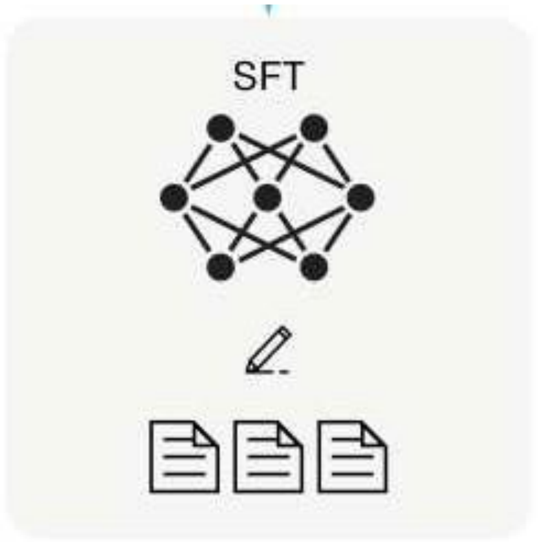
LLM fine-tuned in step 1:

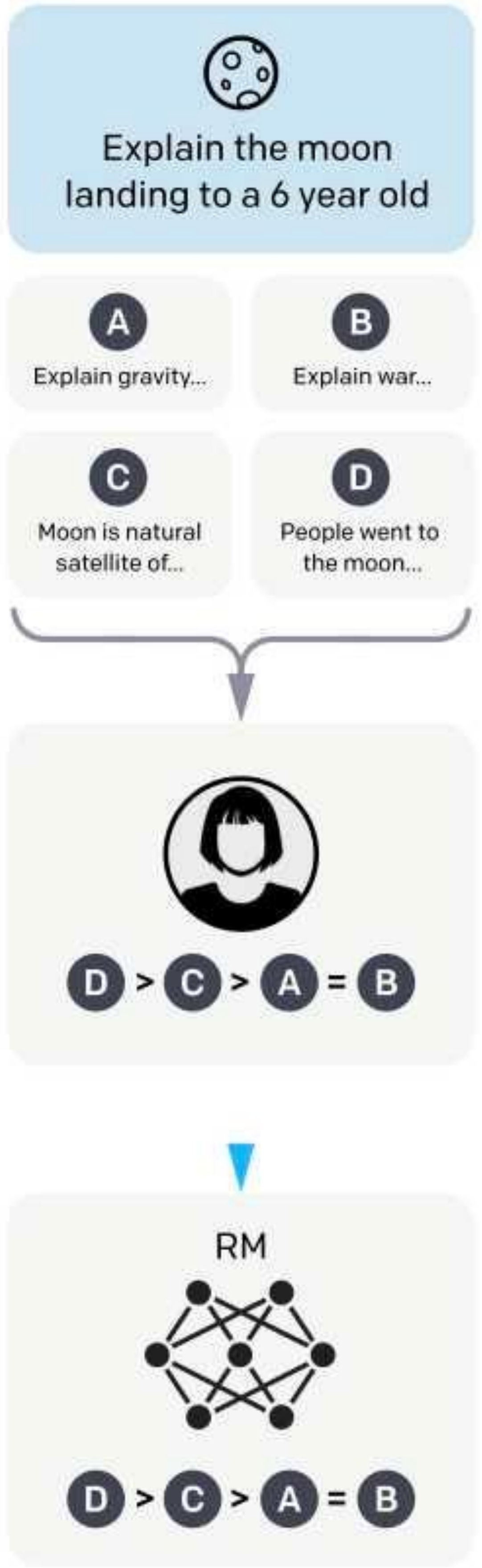


Example: alignment via RLHF

Step 3: Train reward model

LLM fine-tuned in step 1:





Sample prompt

Collect model responses

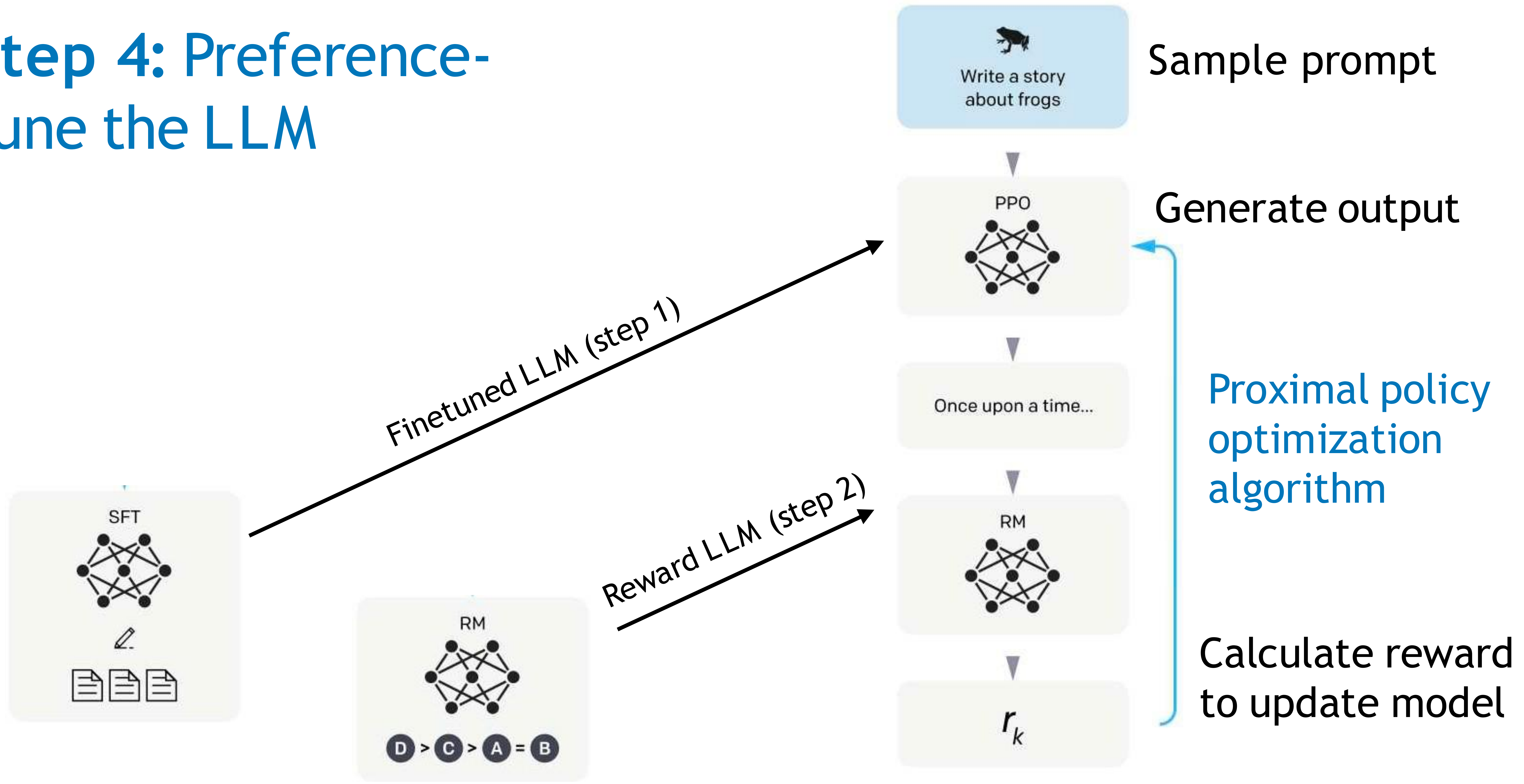
Human ranks responses

Time & labor intensive

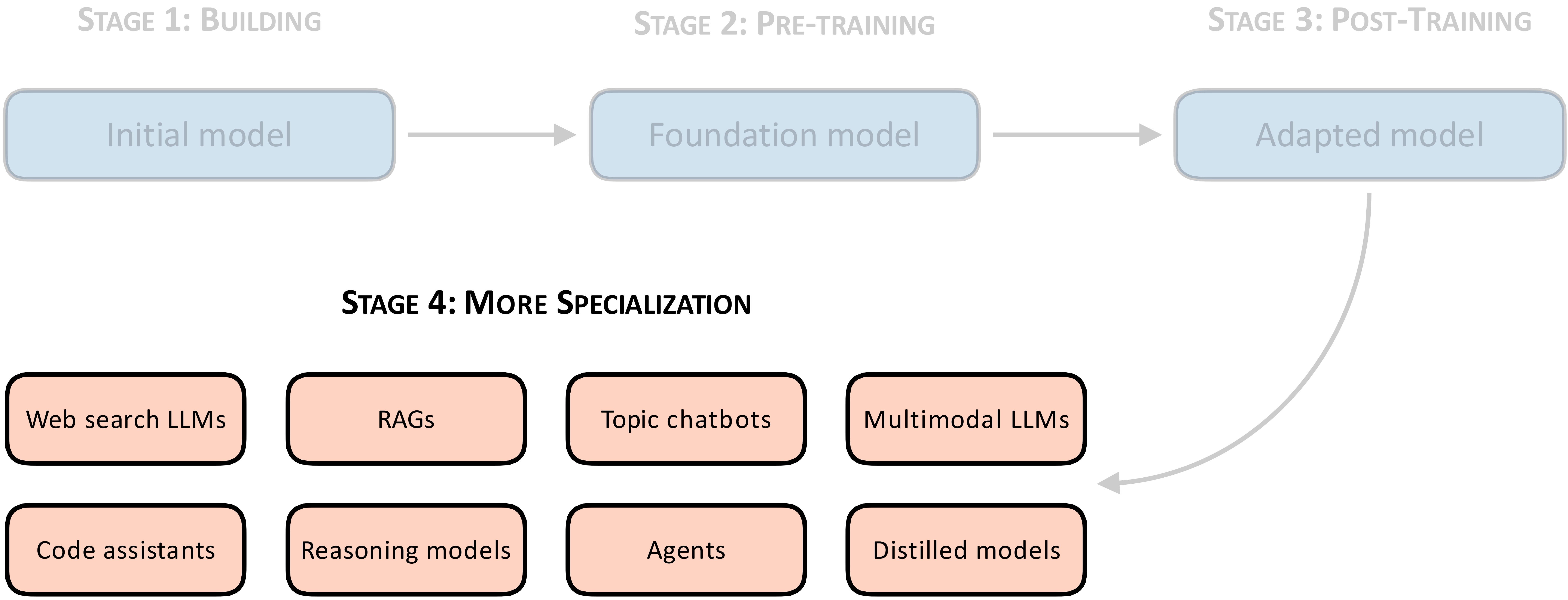
Train reward model
(Another LLM)

Example: alignment via RLHF

Step 4: Preference-tune the LLM



Developing an LLM



Summary

Pretraining from scratch: almost never necessary

Continued pretraining: expand knowledge

Finetuning: special use case, follow instructions

Alignment: improve helpfulness + safety

References

[Understanding the LLM Development Cycle]

- By Raschka (2024)

[From Foundations to Reasoning Models]

- By Raschka (2025)

UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Coming next...

Prompting

Rodrygo Santos
Anisio Lacerda