Deep Learning for Data Science
DS 542

https://dl4ds.github.io/fa2025/

Training, Tuning and Evaluating LLMs

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

Plan for Today

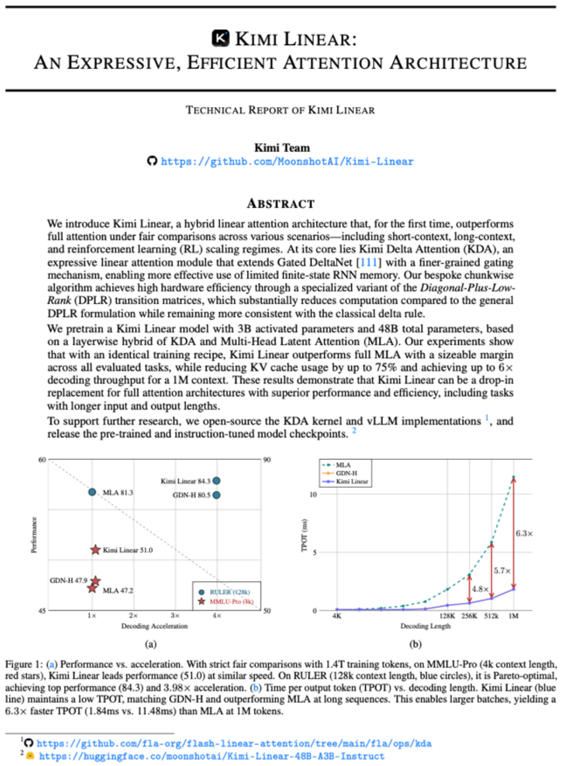
- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)

Re: Sub-Quadratic Attention

- Last time, I cast shade on attempts at sub-quadratic so far…

- My summary of results so far -
    - Demonstrations only for smaller model and/or data set sizes (10B vs 1T).
    - Limited context windows (e.g. 4K vs 1M).
    - None of the state of art models have adopted these supposedly better practices.

That very night on Twitter...



KIMI LINEAR:
AN EXPRESSIVE, EFFICIENT ATTENTION ARCHITECTURE

TECHNICAL REPORT OF KIMI LINEAR

Kimi Team
https://github.com/MoonshotAI/Kimi-Linear

ABSTRACT

We introduce Kimi Linear, a hybrid linear attention architecture that, for the first time, outperforms full attention under fair comparisons across various scenarios—including short-context, long-context, and reinforcement learning (RL) scaling regimes. At its core lies Kimi Delta Attention (KDA), an expressive linear attention module that extends Gated DeltaNet [111] with a finer-grained gating mechanism, enabling more effective use of limited finite-state RNN memory. Our bespoke chunkwise algorithm achieves high hardware efficiency through a specialized variant of the *Diagonal-Plus-Low-Rank* (DPLR) transition matrices, which substantially reduces computation compared to the general DPLR formulation while remaining more consistent with the classical delta rule.

We pretrain a Kimi Linear model with 3B activated parameters and 48B total parameters, based on a layerwise hybrid of KDA and Multi-Head Latent Attention (MLA). Our experiments show that with an identical training recipe, Kimi Linear outperforms full MLA with a sizeable margin across all evaluated tasks, while reducing KV cache usage by up to 75% and achieving up to 6× decoding throughput for a 1M context. These results demonstrate that Kimi Linear can be a drop-in replacement for full attention architectures with superior performance and efficiency, including tasks with longer input and output lengths.

To support further research, we open-source the KDA kernel and vLLM implementations [1], and release the pre-trained and instruction-tuned model checkpoints. [2]

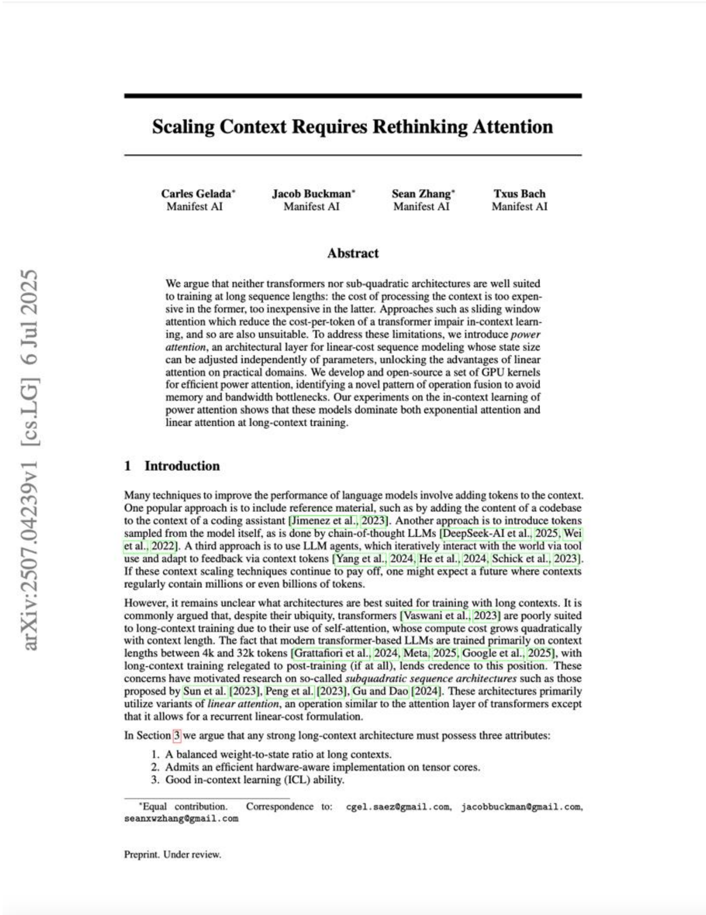
(a)

(b)

Figure 1: (a) Performance vs. acceleration. With strict fair comparisons with 1.4T training tokens, on MMLU-Pro (4k context length, red stars), Kimi Linear leads performance (51.0) at similar speed. On RULER (128k context length, blue circles), it is Pareto-optimal, achieving top performance (84.3) and 3.98× acceleration. (b) Time per output token (TPOT) vs. decoding length. Kimi Linear (blue line) maintains a low TPOT, matching GDN-H and outperforming MLA at long sequences. This enables larger batches, yielding a 6.3× faster TPOT (1.84ms vs. 11.48ms) than MLA at 1M tokens.

---

[1] https://github.com/fla-org/flash-linear-attention/tree/main/fla/ops/kda
[2] https://huggingface.co/moonshotai/Kimi-Linear-48B-A3B-Instruct



arXiv:2507.04239v1 [cs.LG] 6 Jul 2025

Scaling Context Requires Rethinking Attention

Carles Gelada* Manifest AI
Jacob Buckman* Manifest AI
Sean Zhang* Manifest AI
Txus Bach Manifest AI

Abstract

We argue that neither transformers nor sub-quadratic architectures are well suited to training at long sequence lengths: the cost of processing the context is too expensive in the former, too inexpensive in the latter. Approaches such as sliding window attention which reduce the cost-per-token of a transformer impair in-context learning, and so are also unsuitable. To address these limitations, we introduce *power attention*, an architectural layer for linear-cost sequence modeling whose state size can be adjusted independently of parameters, unlocking the advantages of linear attention on practical domains. We develop and open-source a set of GPU kernels for efficient power attention, identifying a novel pattern of operation fusion to avoid memory and bandwidth bottlenecks. Our experiments on the in-context learning of power attention shows that these models dominate both exponential attention and linear attention at long-context training.

1 Introduction

Many techniques to improve the performance of language models involve adding tokens to the context. One popular approach is to include reference material, such as by adding the content of a codebase to the context of a coding assistant [Jimenez et al., 2023]. Another approach is to introduce tokens sampled from the model itself, as is done by chain-of-thought LLMs [DeepSeek-AI et al., 2025, Wei et al., 2022]. A third approach is to use LLM agents, which iteratively interact with the world via tool use and adapt to feedback via context tokens [Yang et al., 2024, He et al., 2024, Schick et al., 2023]. If these context scaling techniques continue to pay off, one might expect a future where contexts regularly contain millions or even billions of tokens.

However, it remains unclear what architectures are best suited for training with long contexts. It is commonly argued that, despite their ubiquity, transformers [Vaswani et al., 2023] are poorly suited to long-context training due to their use of self-attention, whose compute cost grows quadratically with context length. The fact that modern transformer-based LLMs are trained primarily on context lengths between 4k and 32k tokens [Grattafiori et al., 2024, Meta, 2025, Google et al., 2025], with long-context training relegated to post-training (if at all), lends credence to this position. These concerns have motivated research on so-called *subquadratic sequence architectures* such as those proposed by Sun et al. [2023], Peng et al. [2023], Gu and Dao [2024]. These architectures primarily utilize variants of *linear attention*, an operation similar to the attention layer of transformers except that it allows for a recurrent linear-cost formulation.

In Section 3 we argue that any strong long-context architecture must possess three attributes:

1. A balanced weight-to-state ratio at long contexts.
2. Admits an efficient hardware-aware implementation on tensor cores.
3. Good in-context learning (ICL) ability.

---

*Equal contribution. Correspondence to: `cgel.saez@gmail.com`, `jacobbuckman@gmail.com`, `seanxwzhang@gmail.com`

Preprint. Under review.

Twitter’s response

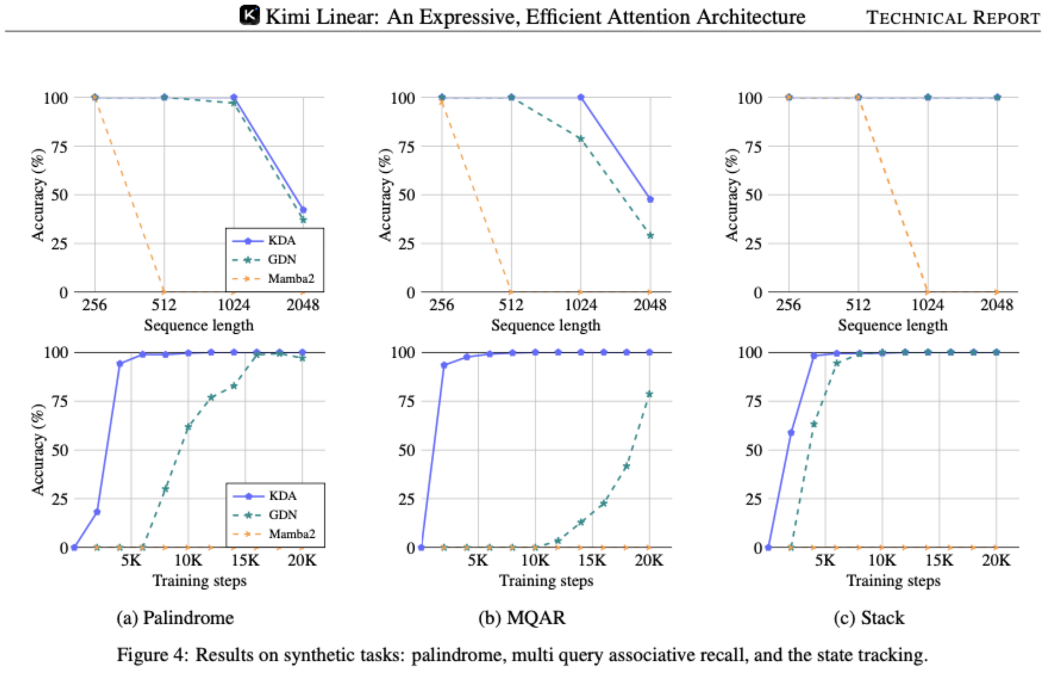
🤷‍♂️ 🤦🏻‍♂️ 🔥 🤮

Linear Language Models are Easy (RNNs). Good Language Models are Hard.

What we want from these -

- LLM with infinite context.
  - Really need a way to refer to any part of the input depending on final question…
- Runs on our laptop or phone.

What we got instead -



Kimi Linear: An Expressive, Efficient Attention Architecture TECHNICAL REPORT

(a) Palindrome (b) MQAR (c) Stack

Figure 4: Results on synthetic tasks: palindrome, multi query associative recall, and the state tracking.

We Want Something Like This to Work, but…

- An attention replacement that took linear or $n \log n$ time would be amazing.

- Repackaging recurrent neural networks don’t address the fundamental problem.

- Testing at small scale with big constant factors hides the real performance at scale.
    - Evaluation on the right problems is important!

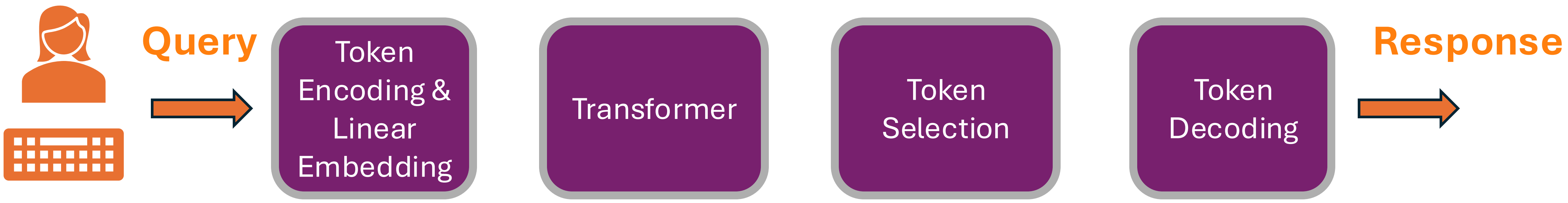
Any questions?

???

Moving on

- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)

LLM Generative Flow



Query

Token Encoding & Linear Embedding

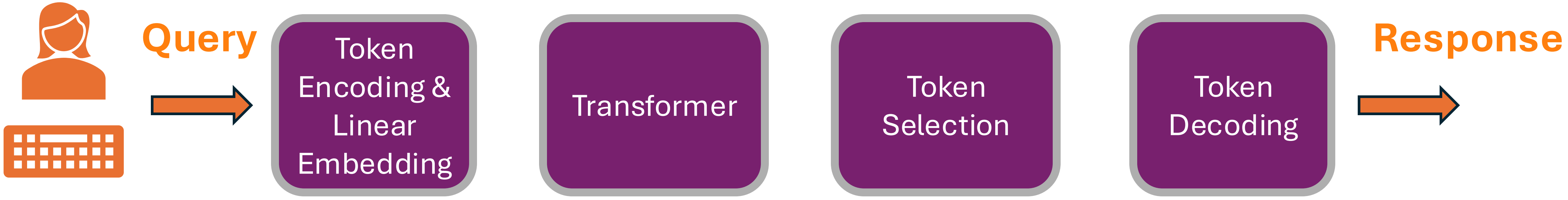
Transformer

Token Selection

Token Decoding

Response

LLM Generative Flow


Query

Token Encoding & Linear Embedding

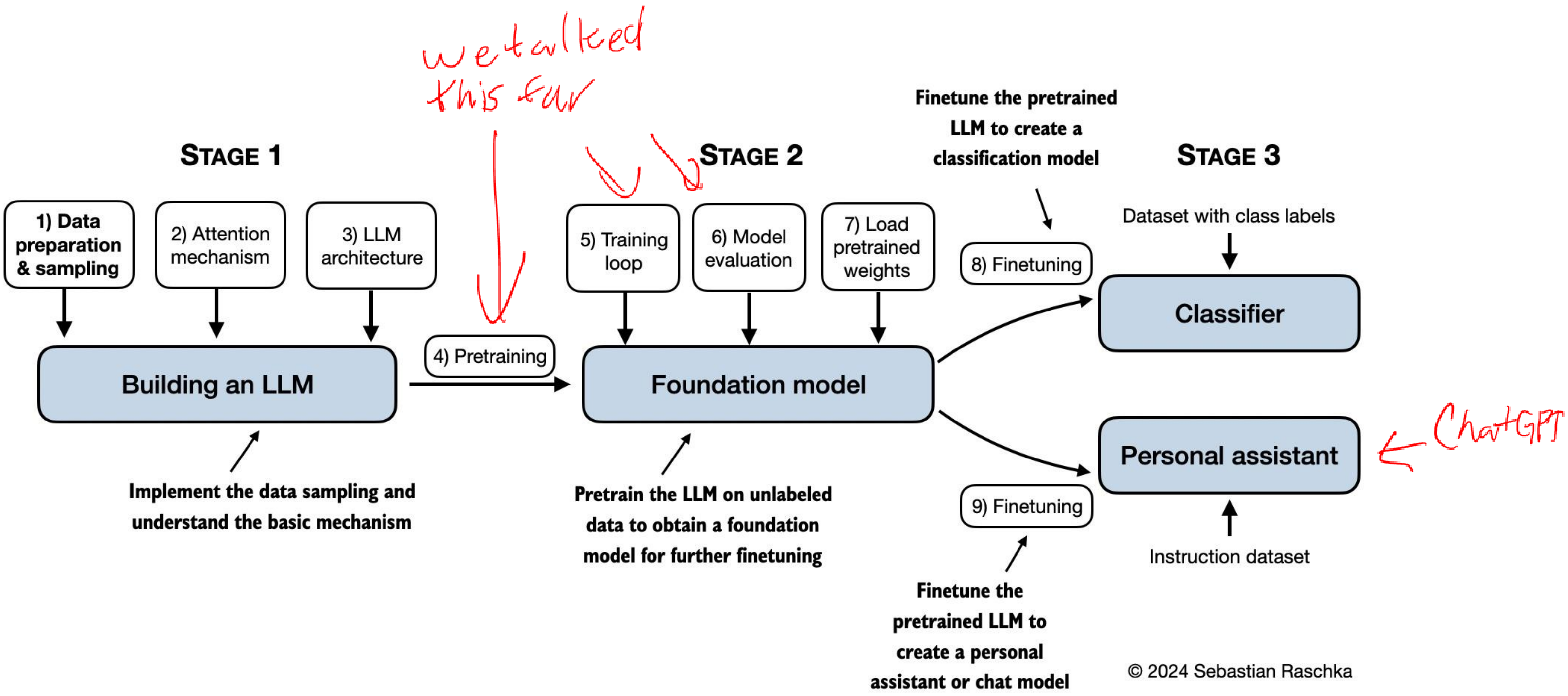
Transformer

Token Selection

Token Decoding

Response

- How do we improve the response?
- How do we evaluate the response?



we talked this far

STAGE 1

1) Data preparation & sampling

2) Attention mechanism

3) LLM architecture

Building an LLM

Implement the data sampling and understand the basic mechanism

4) Pretraining

STAGE 2

5) Training loop

6) Model evaluation

7) Load pretrained weights

Foundation model

Pretrain the LLM on unlabeled data to obtain a foundation model for further finetuning

Finetune the pretrained LLM to create a classification model

8) Finetuning

STAGE 3

Dataset with class labels

Classifier

Personal assistant

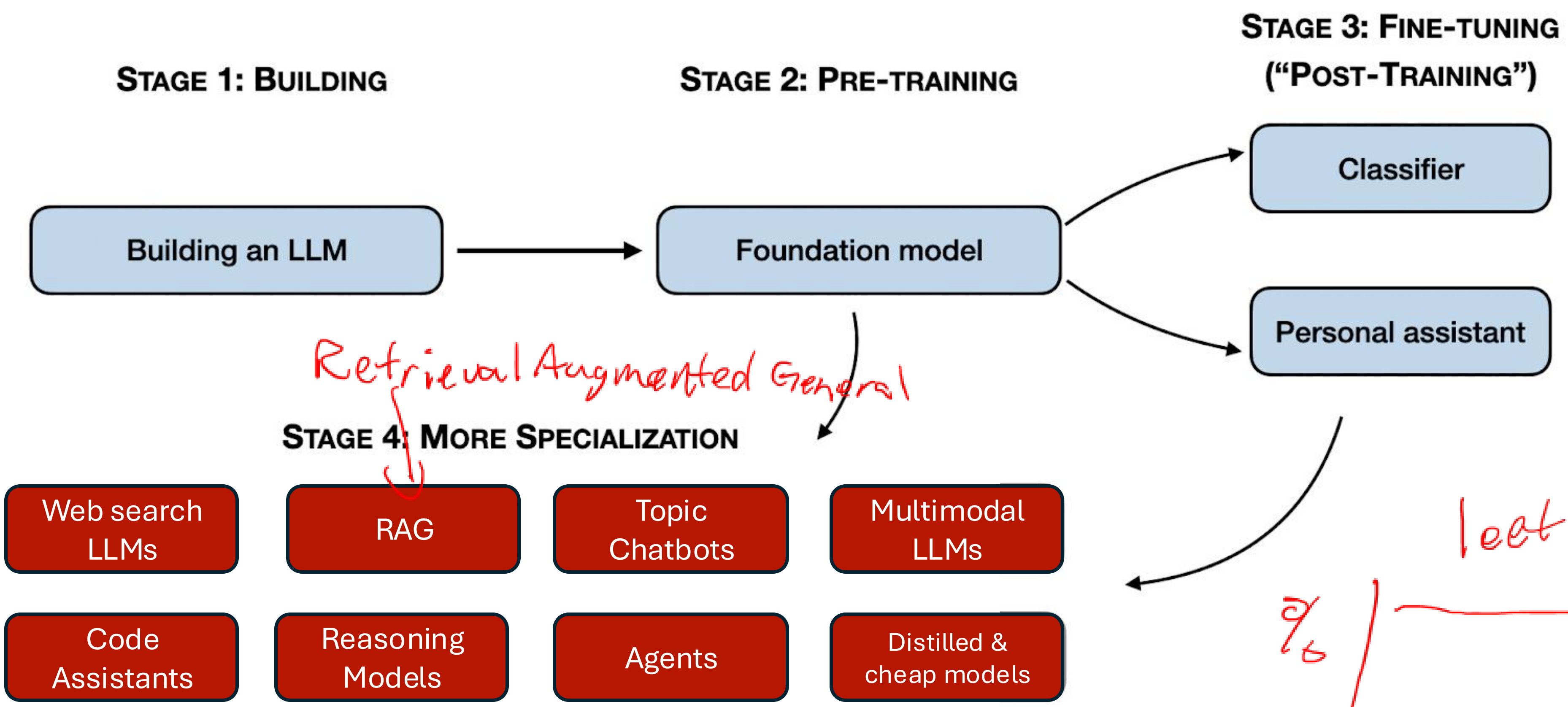
← ChatGPT

9) Finetuning

Instruction dataset

Finetune the pretrained LLM to create a personal assistant or chat model

© 2024 Sebastian Raschka



STAGE 1: BUILDING

Building an LLM

STAGE 2: PRE-TRAINING

Foundation model

STAGE 3: FINE-TUNING ("POST-TRAINING")

Classifier

Personal assistant

Retrieval Augmented General

STAGE 4: MORE SPECIALIZATION

Web search LLMs

RAG

Topic Chatbots

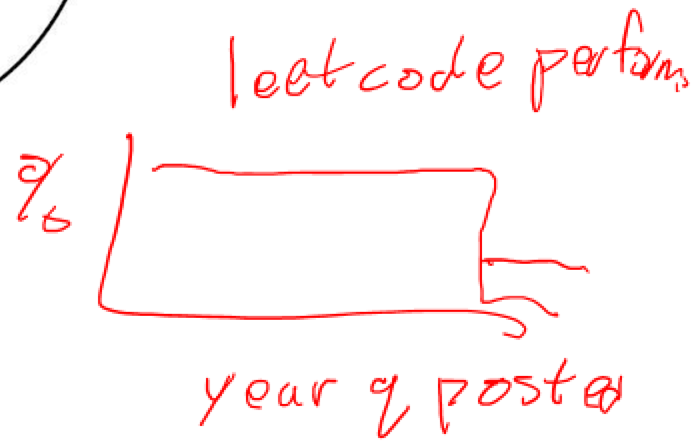
Multimodal LLMs

Code Assistants

Reasoning Models

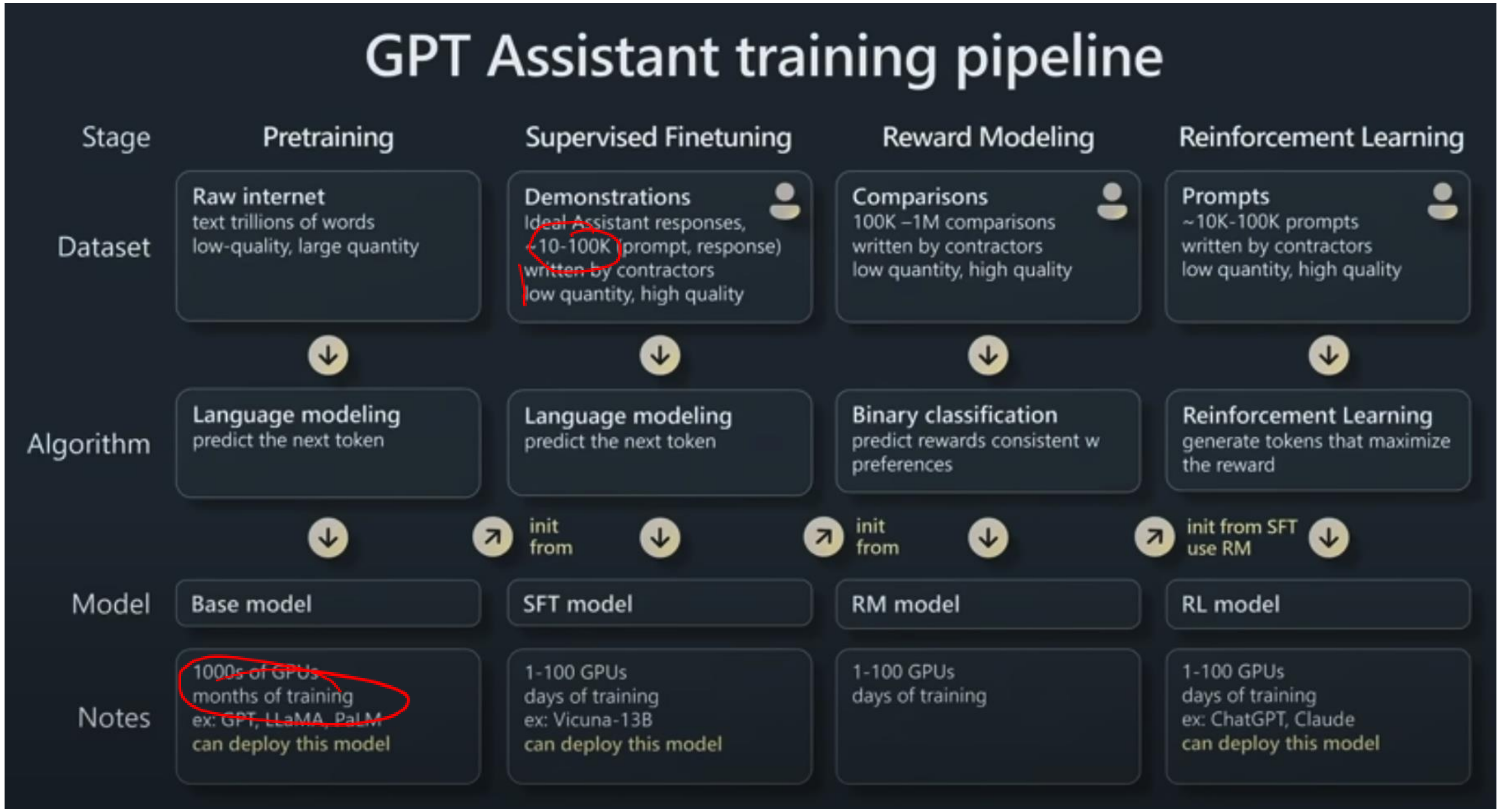
Agents

Distilled & cheap models



https://magazine.sebastianraschka.com/p/understanding-reasoning-llms

How do we build a chat model?

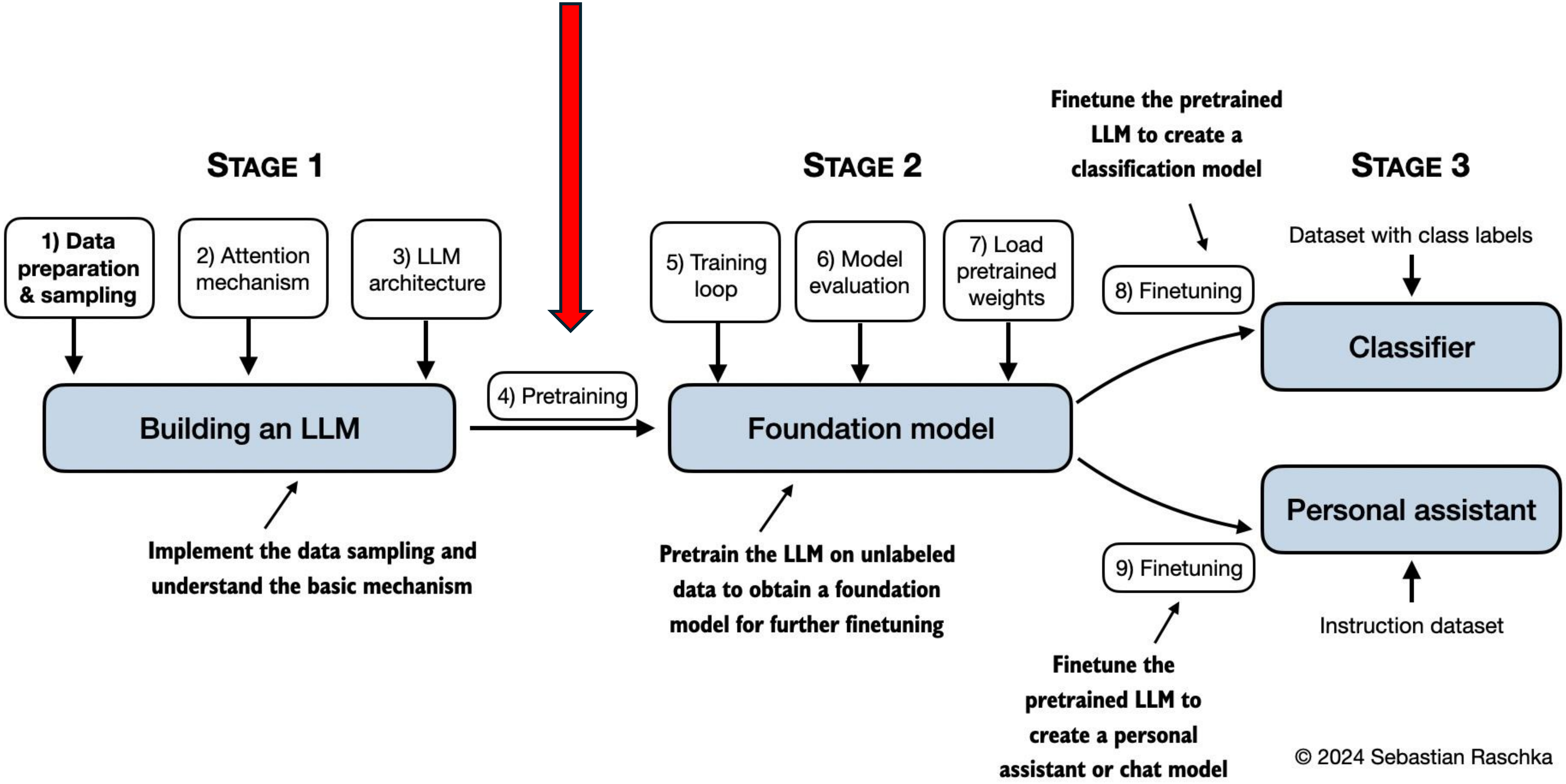


GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM can deploy this model	1-100 GPUs days of training ex: Vicuna-13B can deploy this model	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude can deploy this model

[State of GPT, Andrej Karpathy, MS Build Keynote]

Pre-Training



STAGE 1

1) Data preparation & sampling

2) Attention mechanism

3) LLM architecture

Building an LLM

Implement the data sampling and understand the basic mechanism

4) Pretraining

STAGE 2

5) Training loop

6) Model evaluation

7) Load pretrained weights

Foundation model

Pretrain the LLM on unlabeled data to obtain a foundation model for further finetuning

Finetune the pretrained LLM to create a classification model

8) Finetuning

9) Finetuning

Finetune the pretrained LLM to create a personal assistant or chat model

STAGE 3

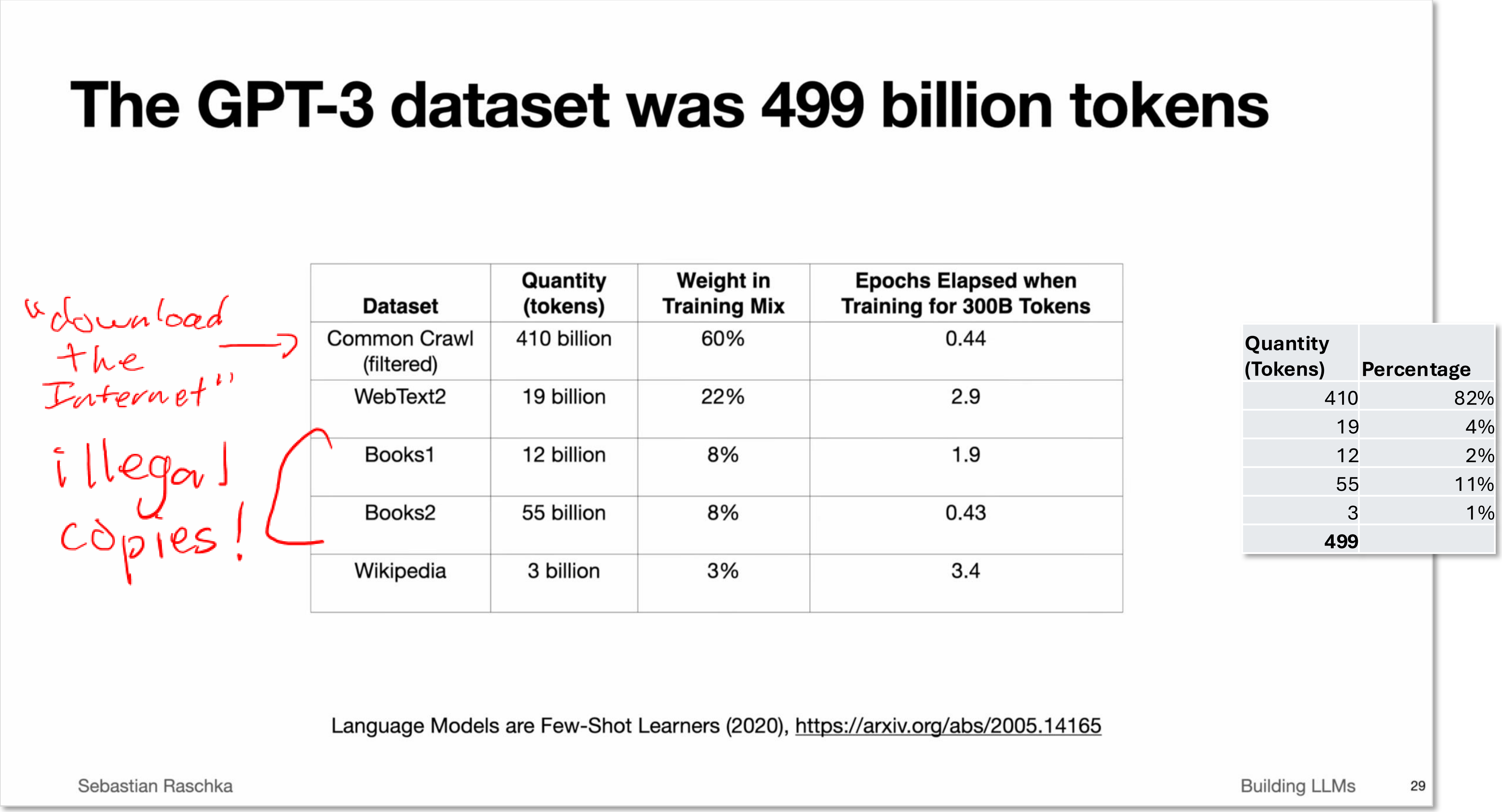
Dataset with class labels

Classifier

Personal assistant

Instruction dataset

© 2024 Sebastian Raschka



The GPT-3 dataset was 499 billion tokens

"download the Internet"

illegal copies!

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

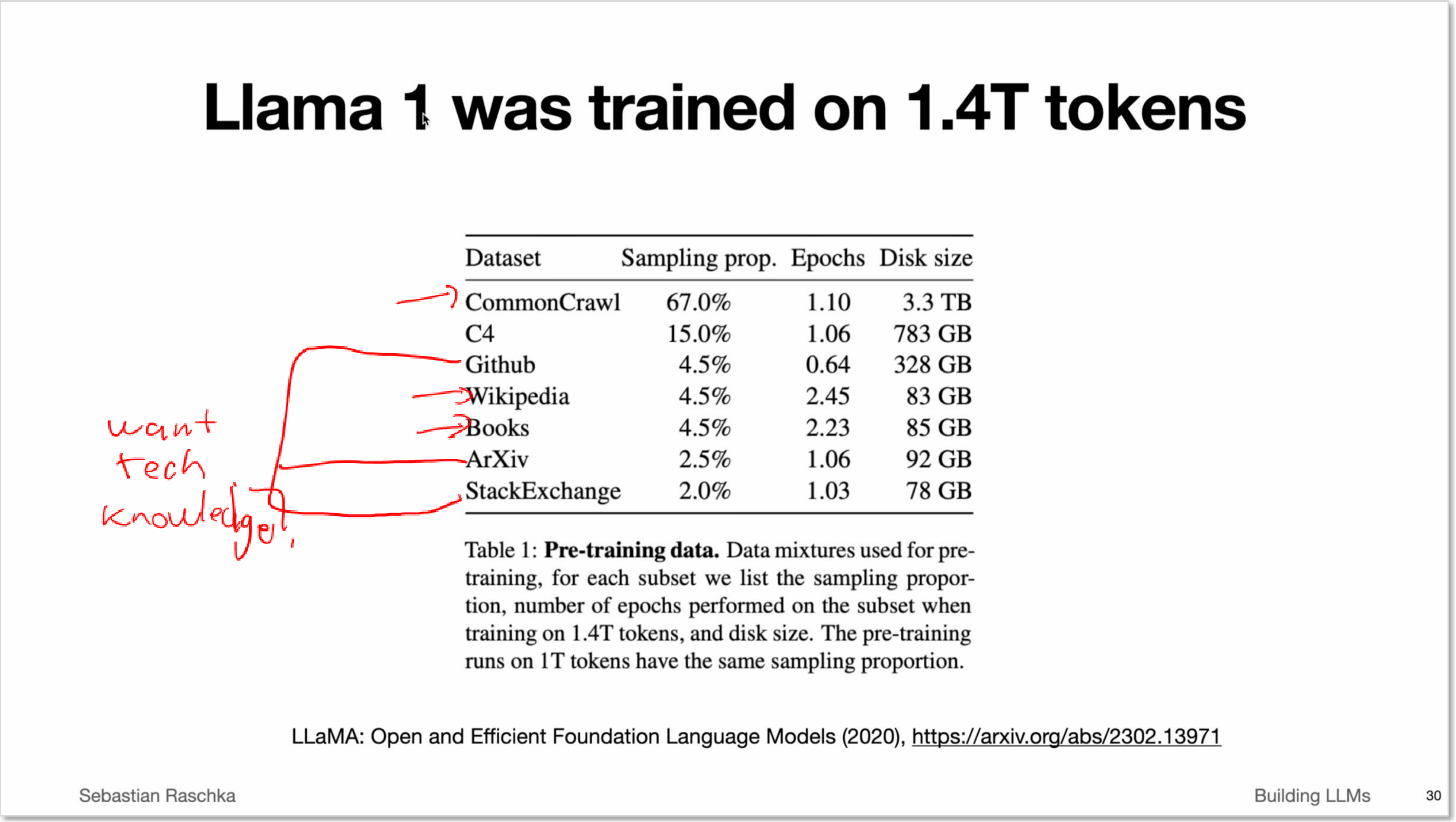
Quantity (Tokens)	Percentage
410	82%
19	4%
12	2%
55	11%
3	1%
499	

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

Sebastian Raschka

Building LLMs 29

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



Llama 1 was trained on 1.4T tokens

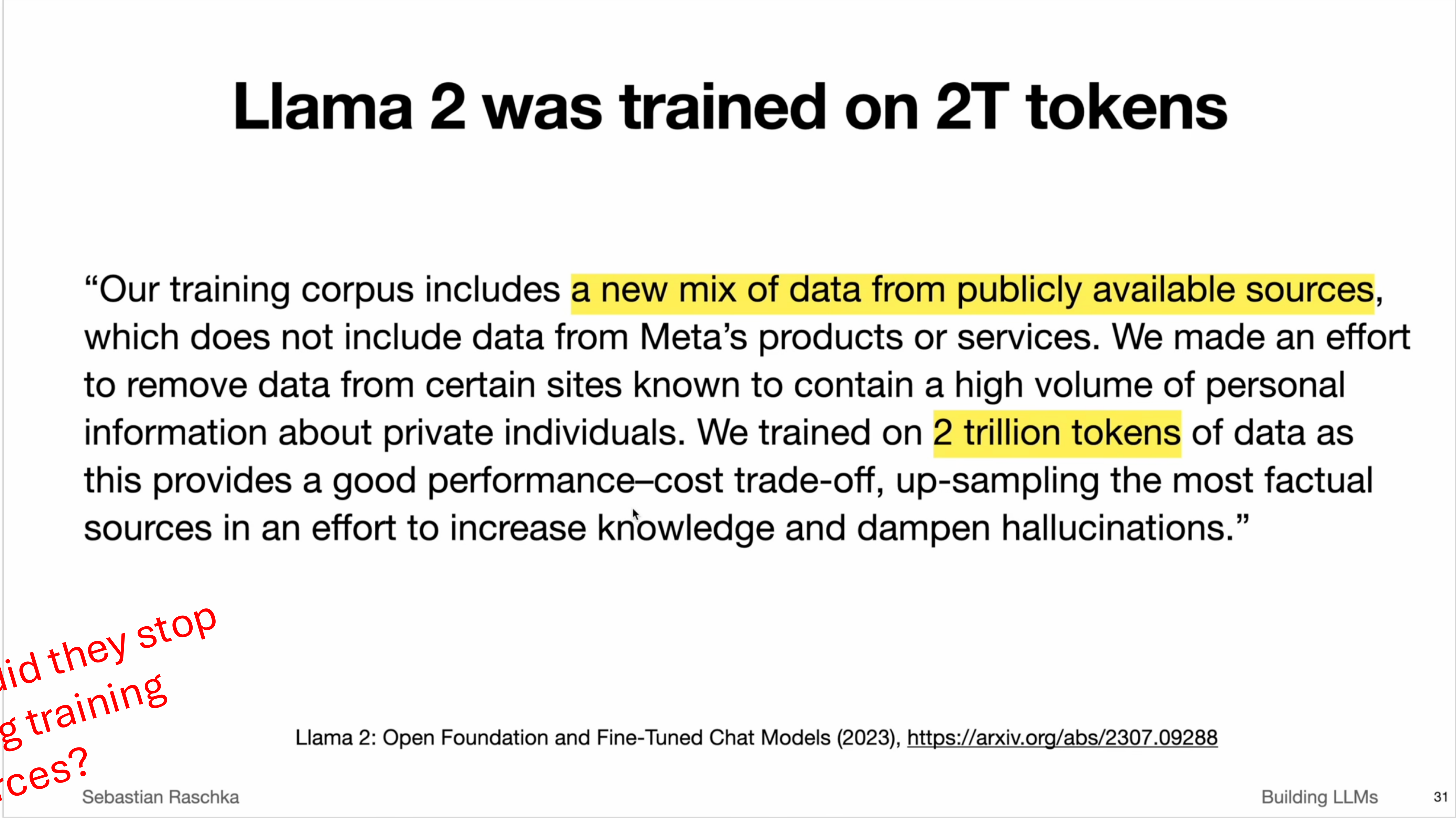
Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

want tech knowledge!

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 (2020), https://arxiv.org/abs/2302.13971

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



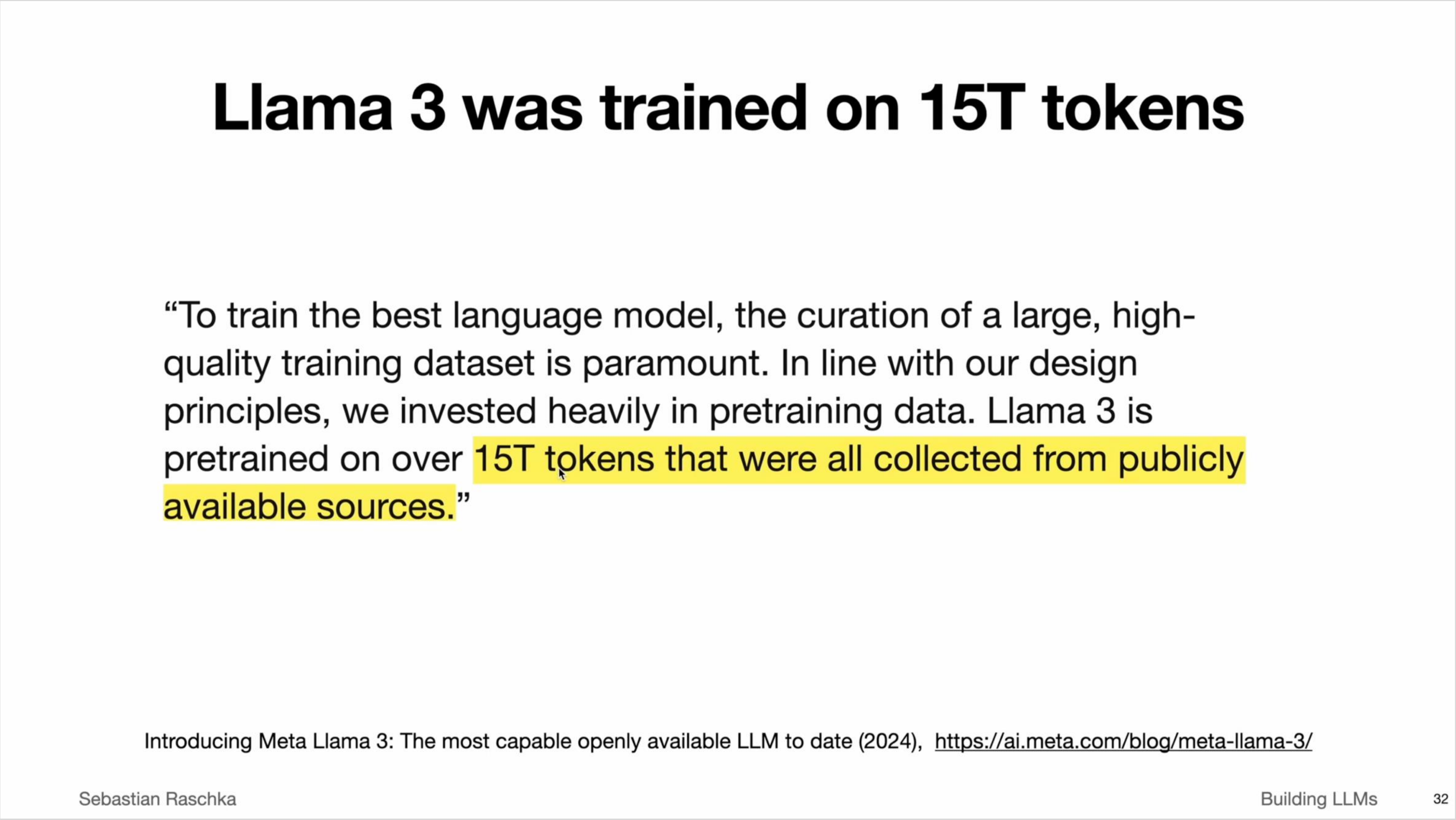
Llama 2 was trained on 2T tokens

"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

Why did they stop listing training sources?

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



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."

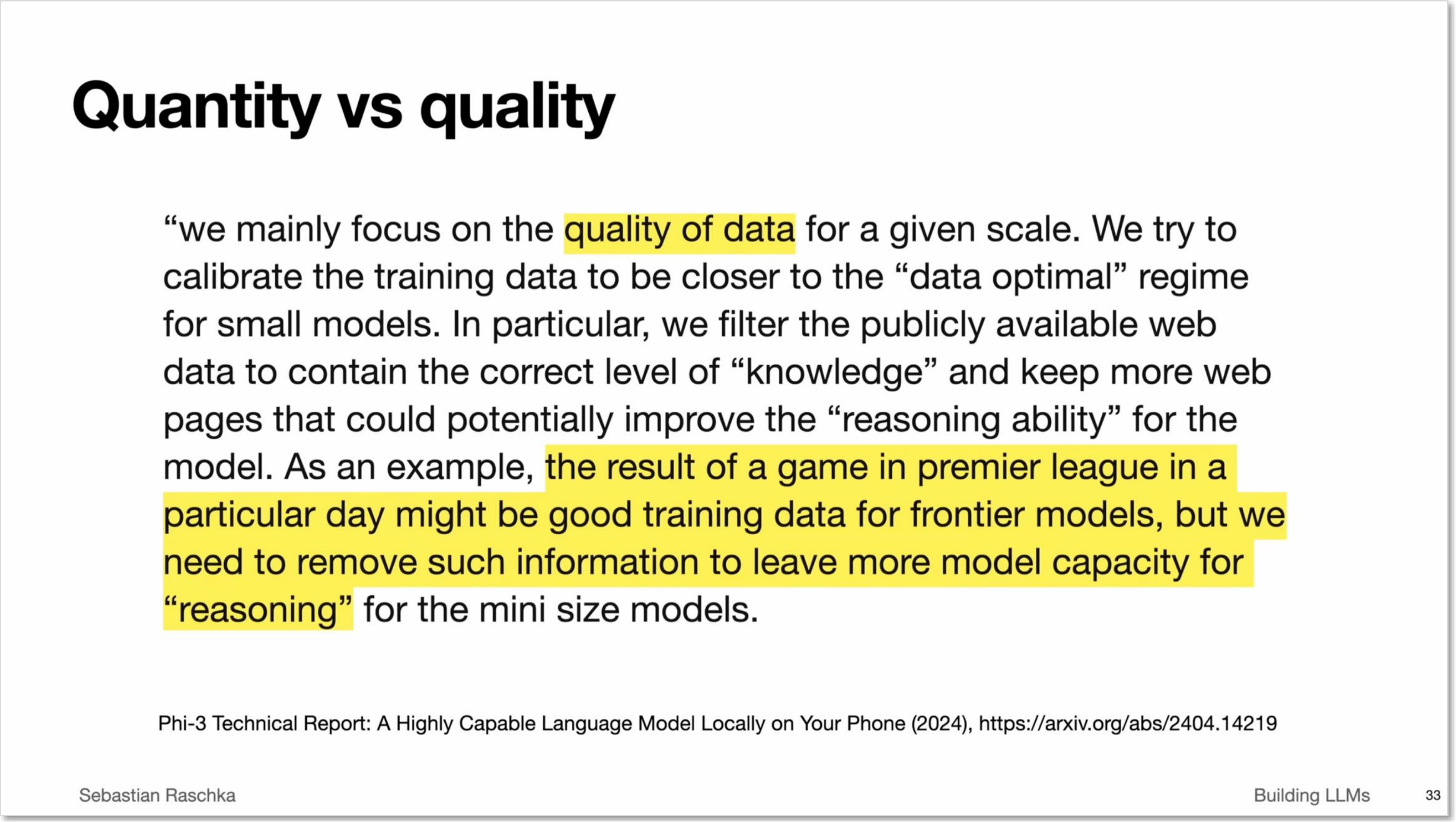
Introducing Meta Llama 3: The most capable openly available LLM to date (2024), https://ai.meta.com/blog/meta-llama-3/

Sebastian Raschka

Building LLMs

32

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



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.

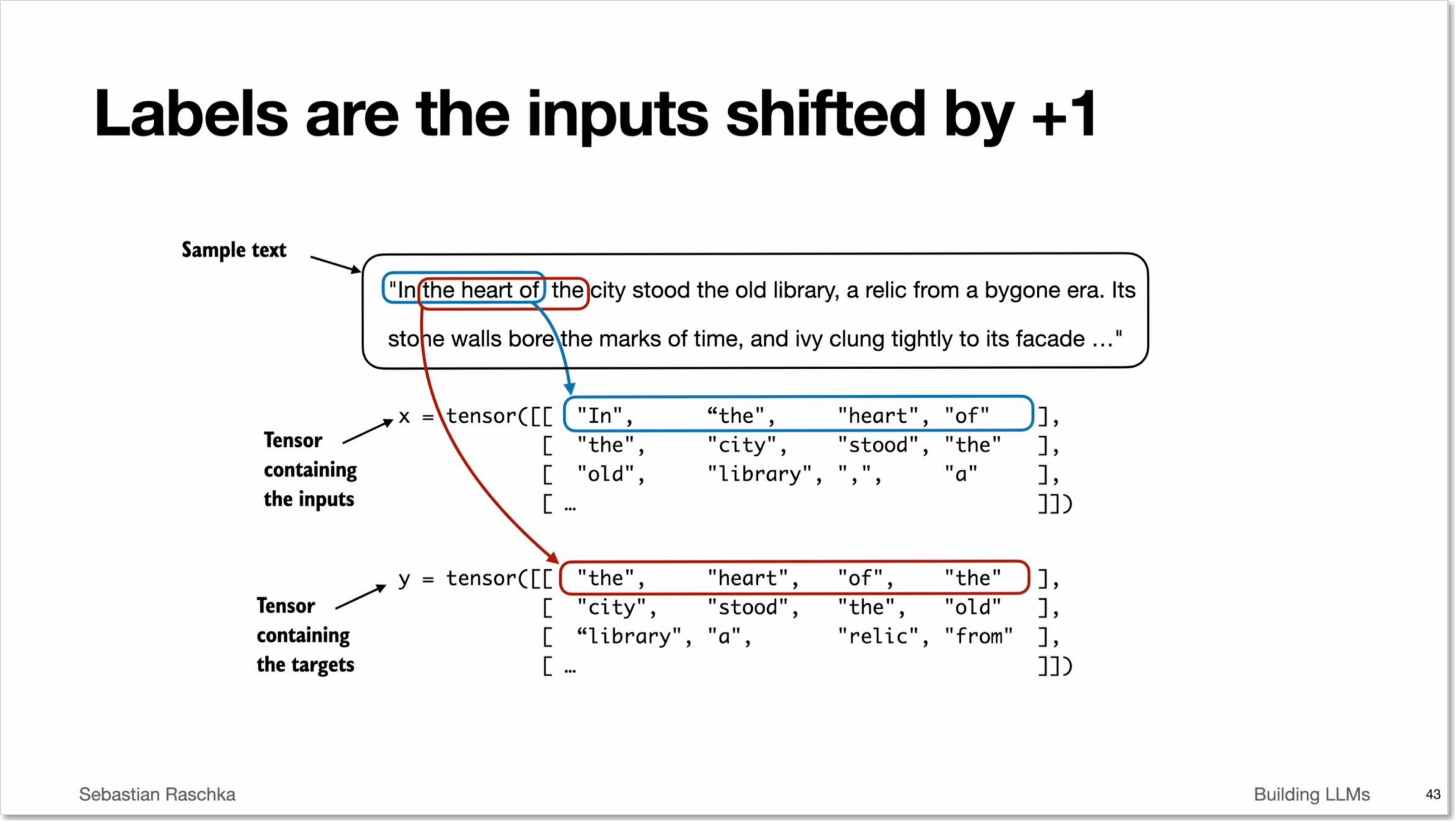
Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone (2024), https://arxiv.org/abs/2404.14219

Sebastian Raschka

Building LLMs

33

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



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 …"

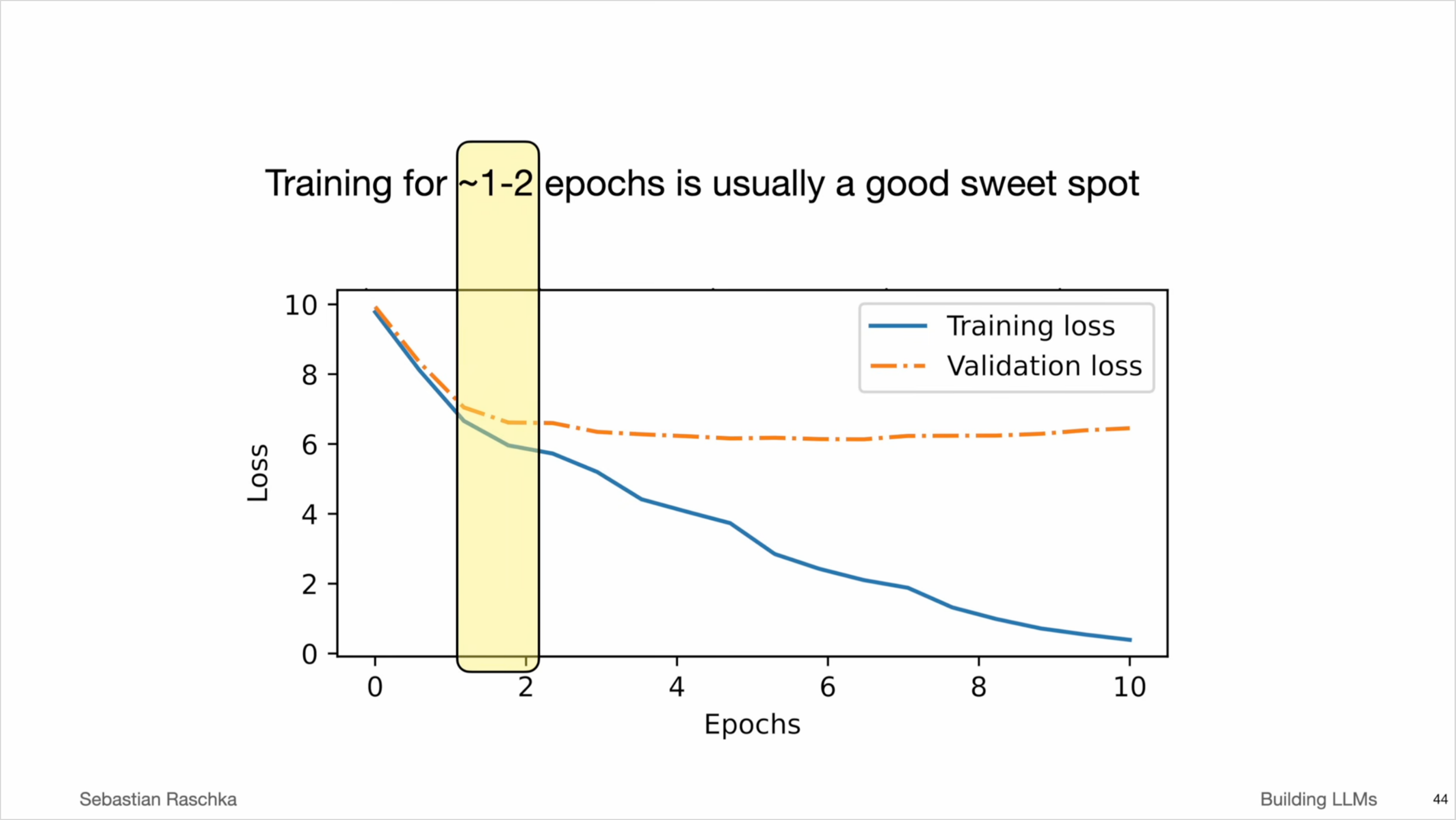
Tensor containing the inputs

```
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" ],
             [ …                                       ]])
```

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]



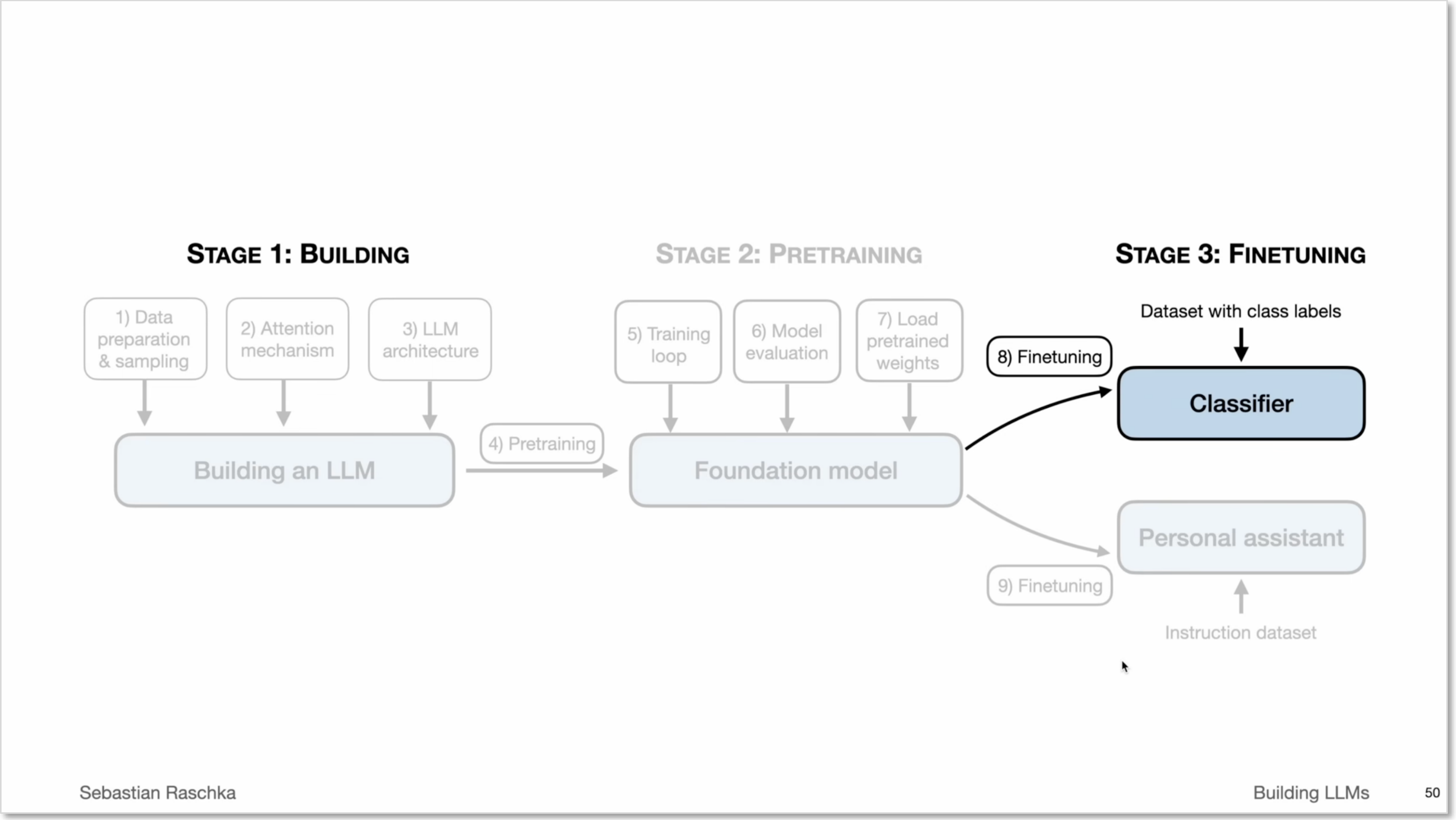
Training for ~1-2 epochs is usually a good sweet spot

Sebastian Raschka

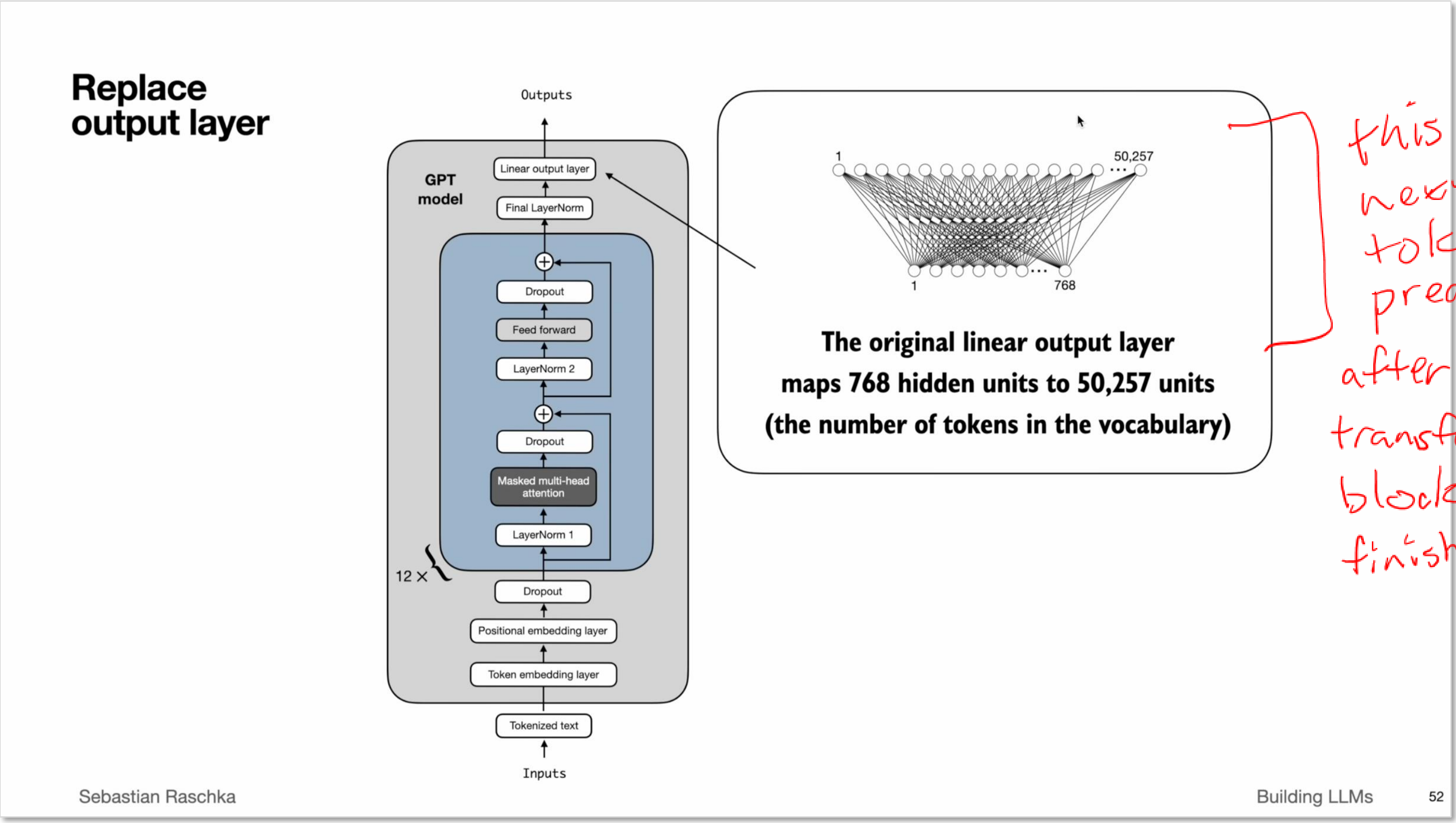
Building LLMs 44

S. Raschka, "Developing an LLM: Building, Training, Finetuning"

Classifier Finetuning



[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]



Replace output layer

The original linear output layer maps 768 hidden units to 50,257 units (the number of tokens in the vocabulary)

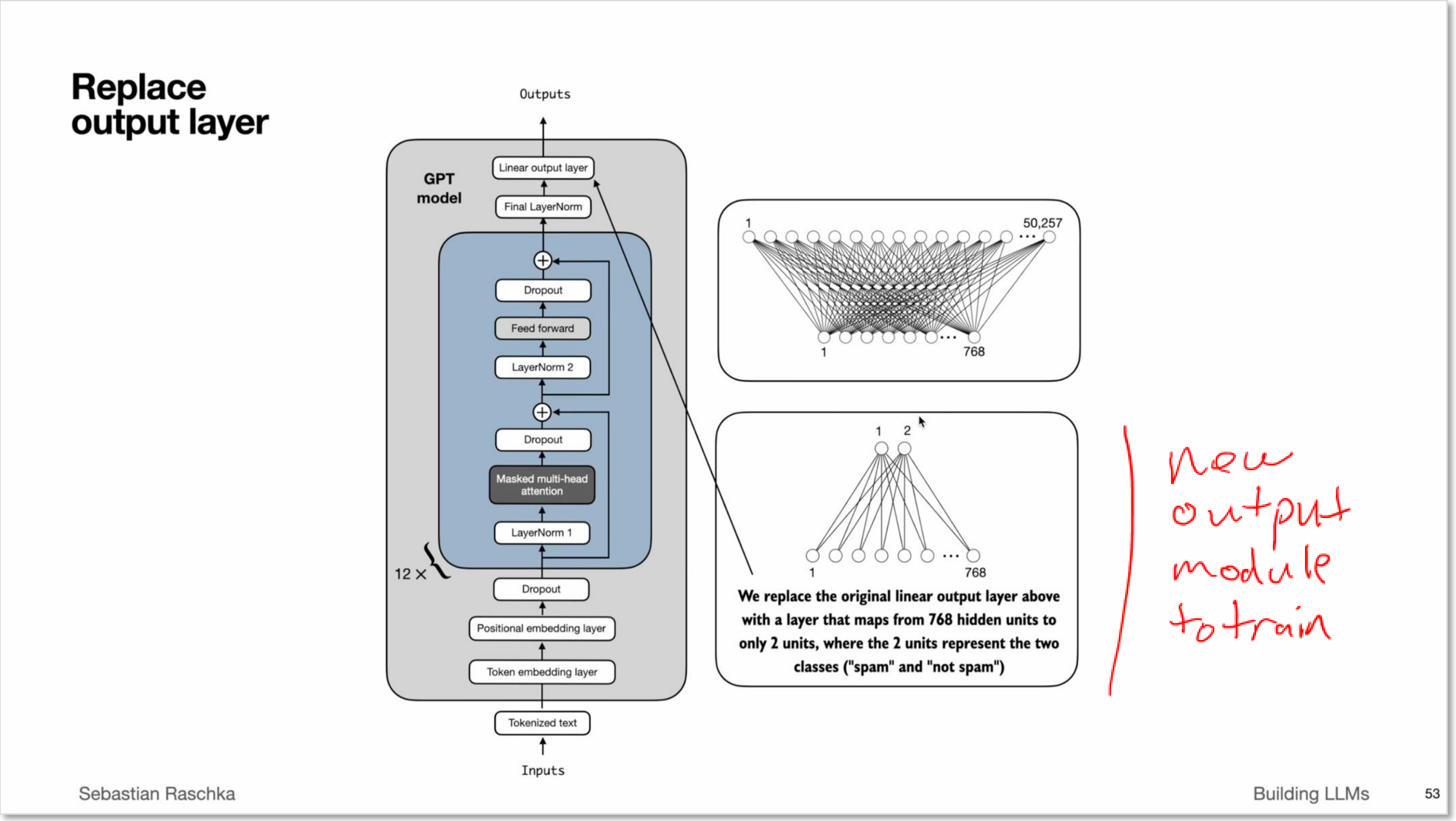
Sebastian Raschka

Building LLMs 52

this did
next
token
prediction
after
transformer
blocks
finished

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]

Example: Spam/Ham Classifier



Replace output layer

We replace the original linear output layer above with a layer that maps from 768 hidden units to only 2 units, where the 2 units represent the two classes ("spam" and "not spam")

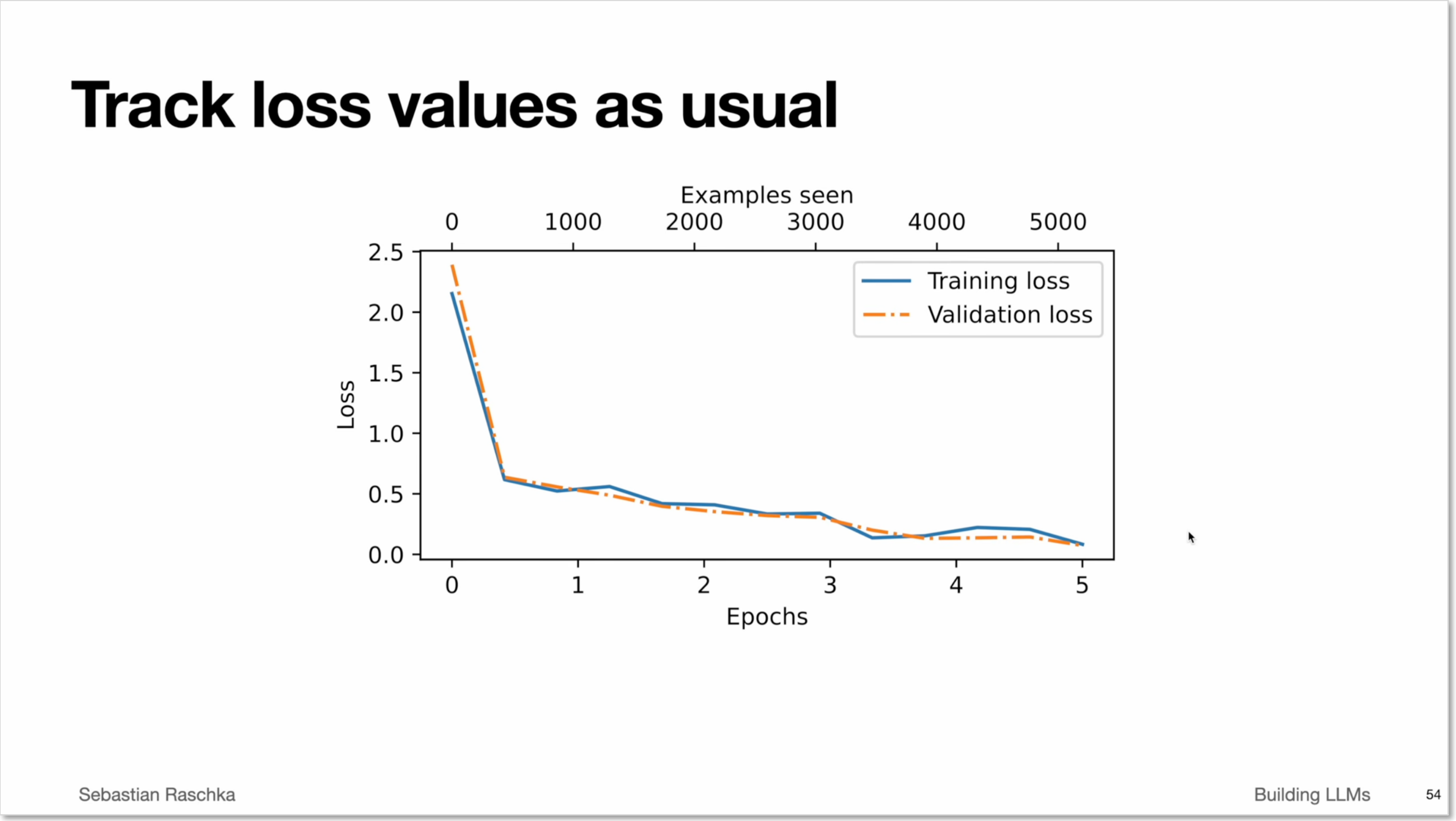
Sebastian Raschka

Building LLMs

53

new
output
module
to train

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



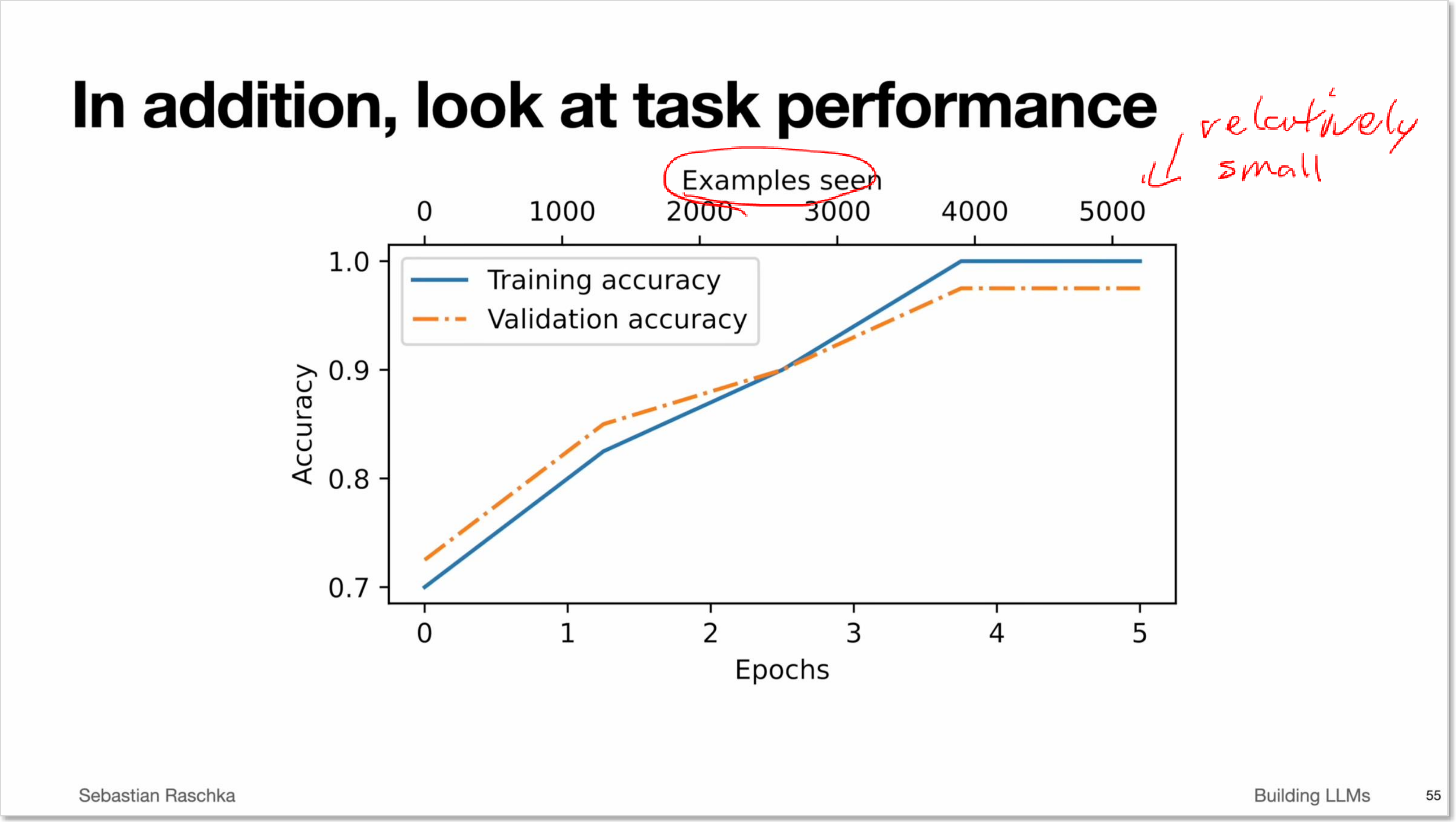
Track loss values as usual

Sebastian Raschka

Building LLMs

54

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]



In addition, look at task performance

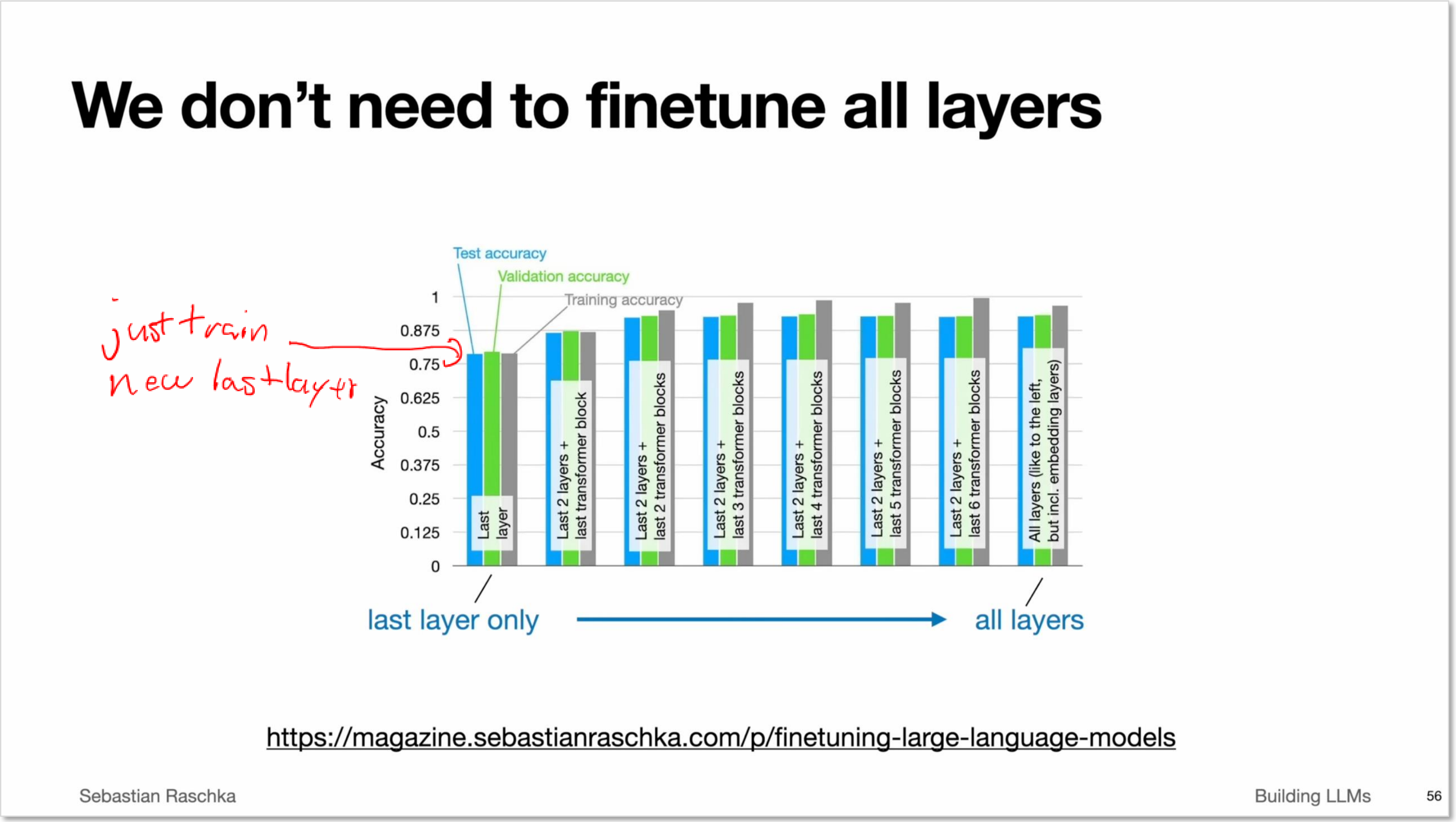
relatively small

Sebastian Raschka

Building LLMs

55

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]



We don't need to finetune all layers

just train new last layer

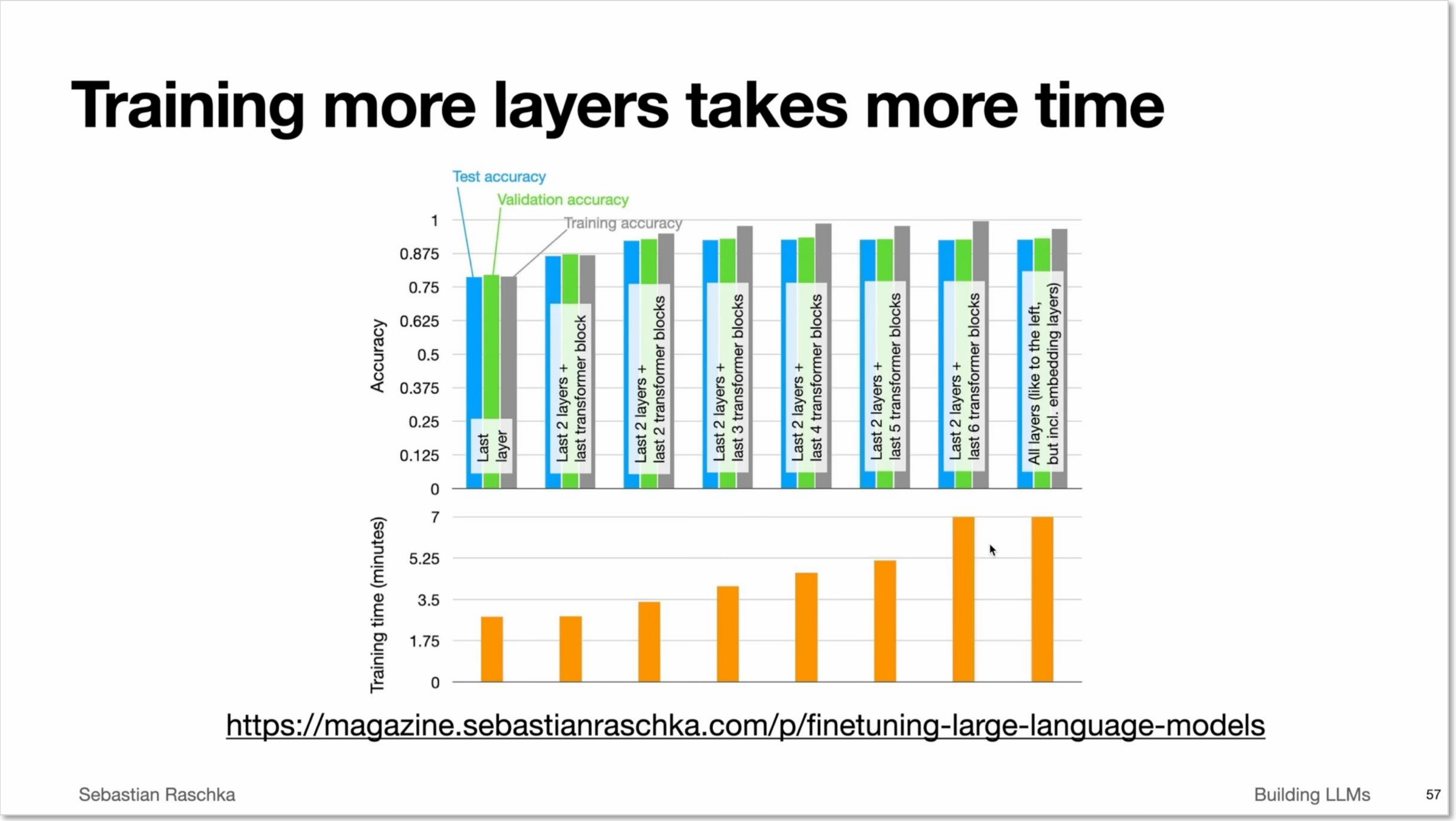
https://magazine.sebastianraschka.com/p/finetuning-large-language-models

Sebastian Raschka

Building LLMs

56

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



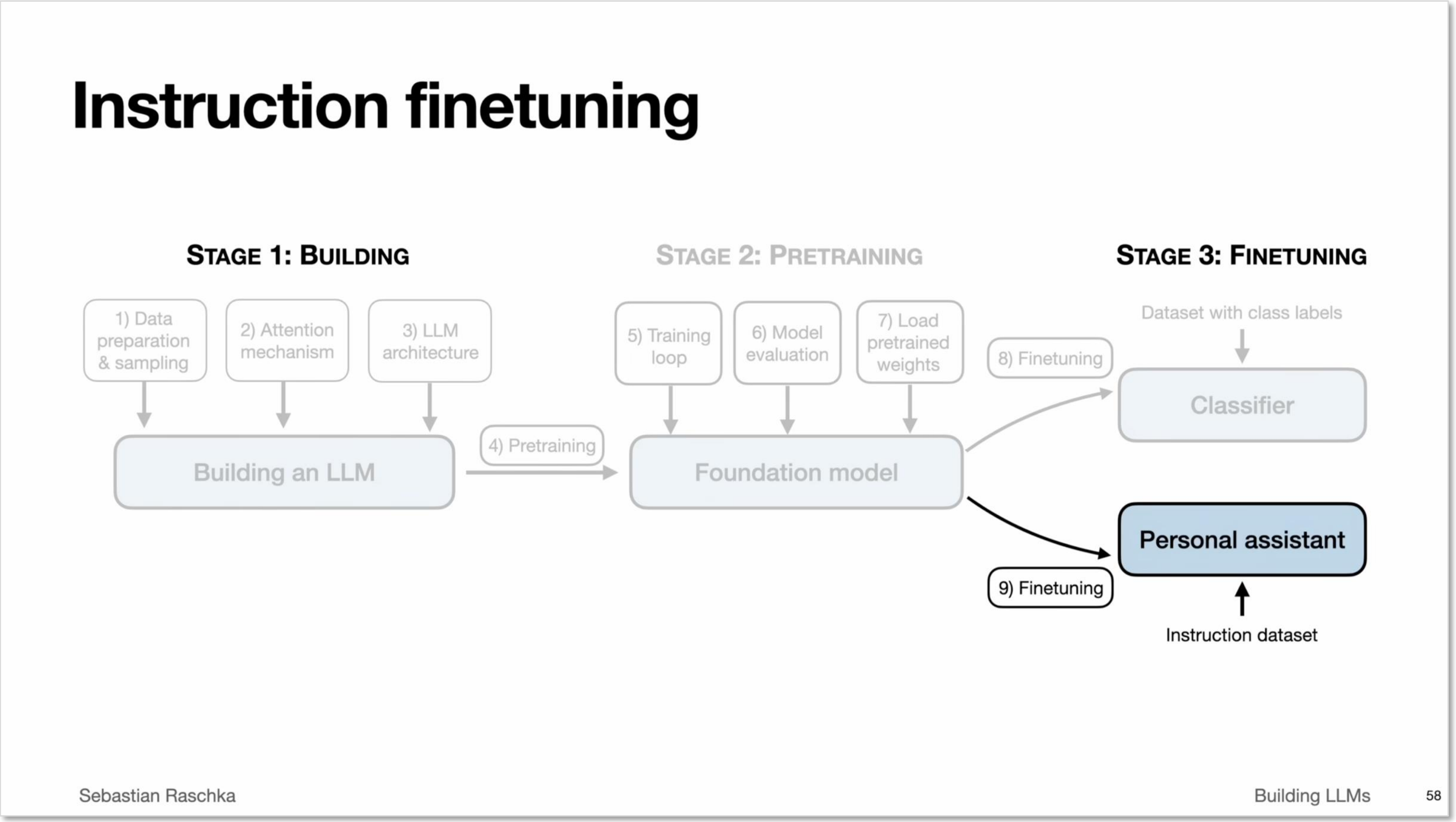
Training more layers takes more time

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

Sebastian Raschka

Building LLMs 57

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



Instruction finetuning

STAGE 1: BUILDING

1) Data preparation & sampling

2) Attention mechanism

3) LLM architecture

Building an LLM

4) Pretraining

STAGE 2: PRETRAINING

5) Training loop

6) Model evaluation

7) Load pretrained weights

Foundation model

STAGE 3: FINETUNING

8) Finetuning

Dataset with class labels

Classifier

9) Finetuning

Personal assistant

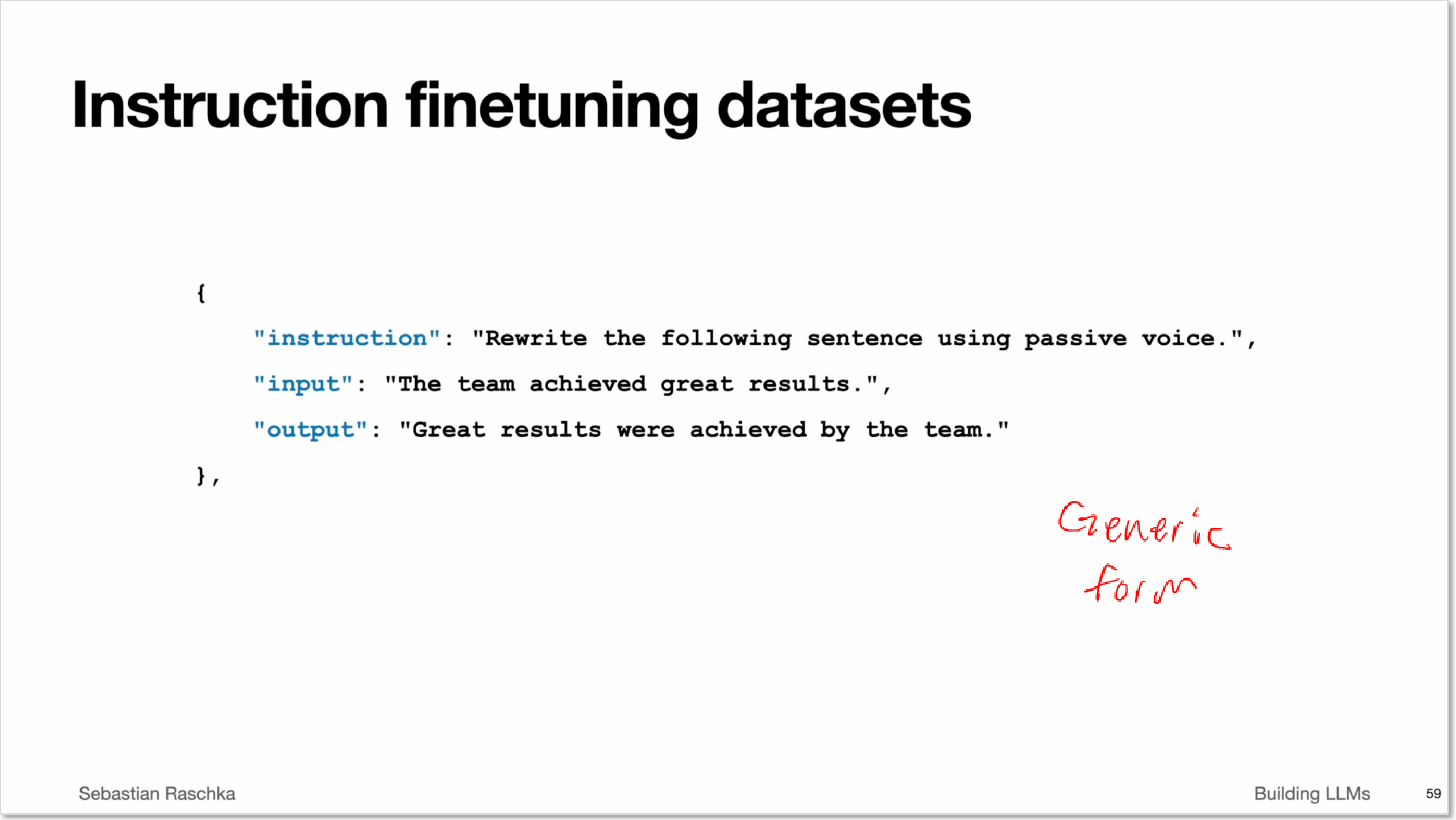
Instruction dataset

Sebastian Raschka

Building LLMs

58

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]



Instruction finetuning datasets

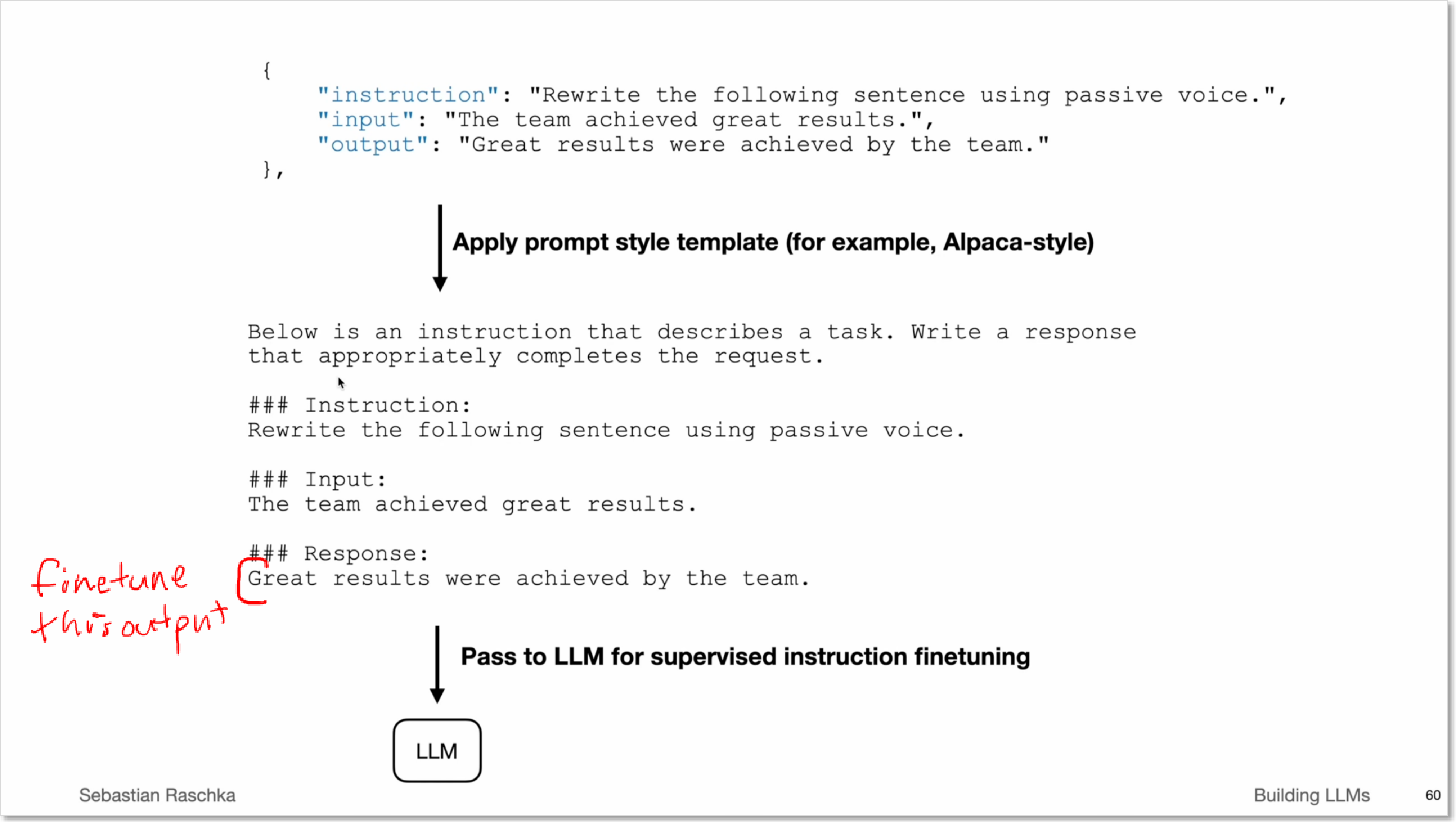
```
{
    "instruction": "Rewrite the following sentence using passive voice.",
    "input": "The team achieved great results.",
    "output": "Great results were achieved by the team."
},
```

Generic form

Sebastian Raschka

Building LLMs 59

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



```
{
    "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.
```

finetune this output

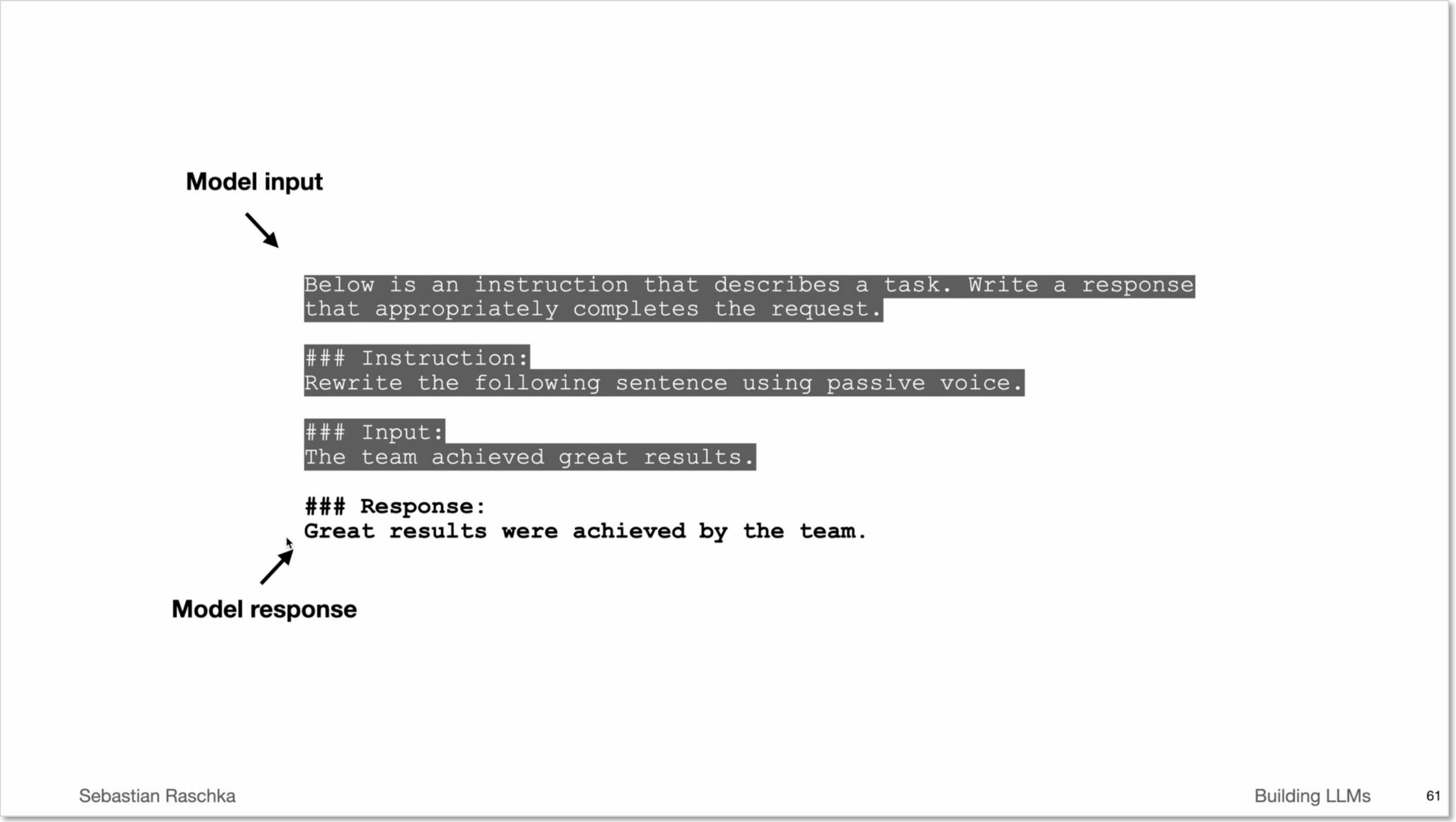
Pass to LLM for supervised instruction finetuning

Sebastian Raschka

Building LLMs

60

[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]()



Model input

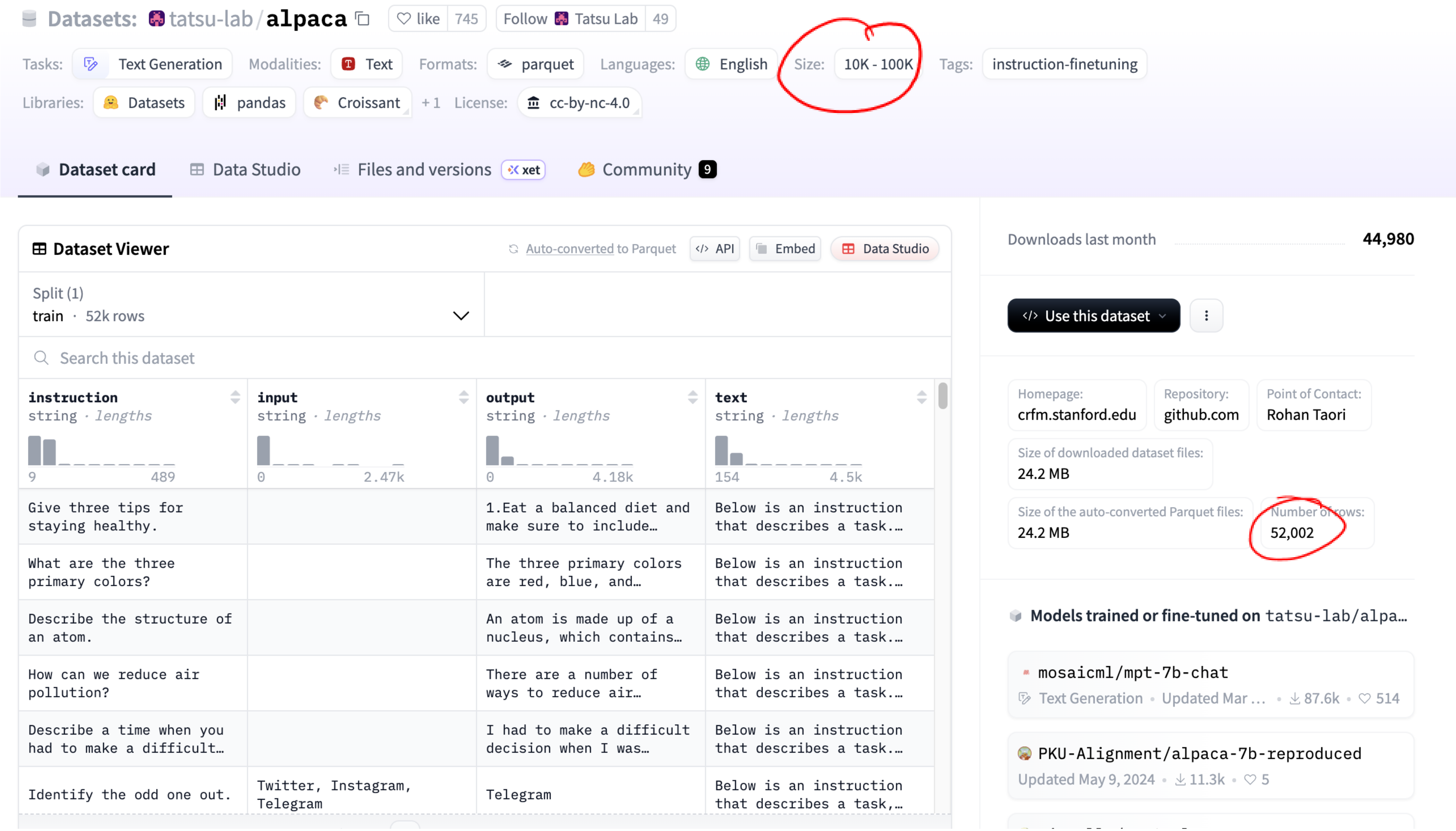
```
Below is an instruction that describes a task. Write a response
that appropriately completes the request.

### Instruction:
Rewrite the following sentence using passive voice.

### Input:
The team achieved great results.

### Response:
Great results were achieved by the team.
```

Model response

Sebastian Raschka Building LLMs 61

S. Raschka, "Developing an LLM: Building, Training, Finetuning"

Alpaca Instruction Tuning Dataset



Datasets: tatsu-lab/alpaca

Tasks: Text Generation Modalities: Text Formats: parquet Languages: English Size: 10K - 100K Tags: instruction-finetuning

Libraries: Datasets pandas Croissant +1 License: cc-by-nc-4.0

Dataset Viewer

Split (1)
train · 52k rows

instruction	input	output	text
string · *lengths*	string · *lengths*	string · *lengths*	string · *lengths*
9 489	0 2.47k	0 4.18k	154 4.5k
Give three tips for staying healthy.		1.Eat a balanced diet and make sure to include…	Below is an instruction that describes a task.…
What are the three primary colors?		The three primary colors are red, blue, and…	Below is an instruction that describes a task.…
Describe the structure of an atom.		An atom is made up of a nucleus, which contains…	Below is an instruction that describes a task.…
How can we reduce air pollution?		There are a number of ways to reduce air…	Below is an instruction that describes a task.…
Describe a time when you had to make a difficult…		I had to make a difficult decision when I was…	Below is an instruction that describes a task.…
Identify the odd one out.	Twitter, Instagram, Telegram	Telegram	Below is an instruction that describes a task,…

Downloads last month 44,980

Homepage: crfm.stanford.edu
Repository: github.com
Point of Contact: Rohan Taori

Size of downloaded dataset files: 24.2 MB

Size of the auto-converted Parquet files: 24.2 MB

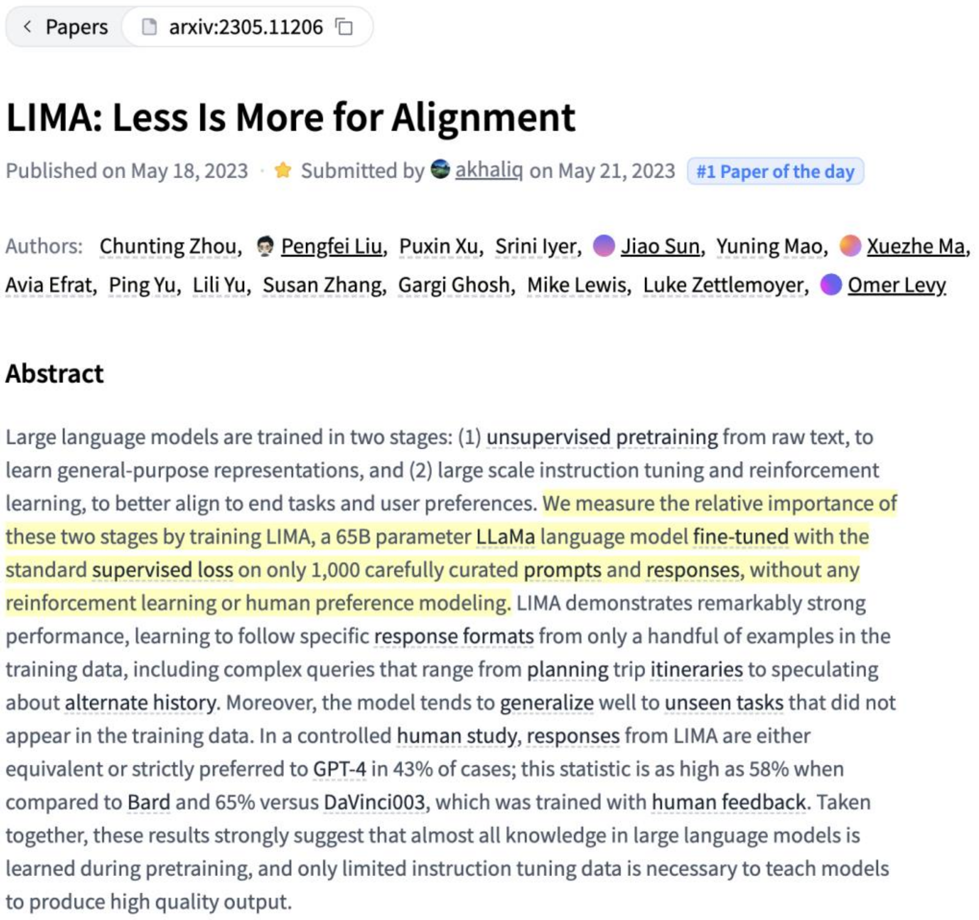
Number of rows: 52,002

Models trained or fine-tuned on tatsu-lab/alpa…

mosaicml/mpt-7b-chat
Text Generation • Updated Mar … • 87.6k • 514

PKU-Alignment/alpaca-7b-reproduced
Updated May 9, 2024 • 11.3k • 5

LIMA: Finetuning with only 1K instructions



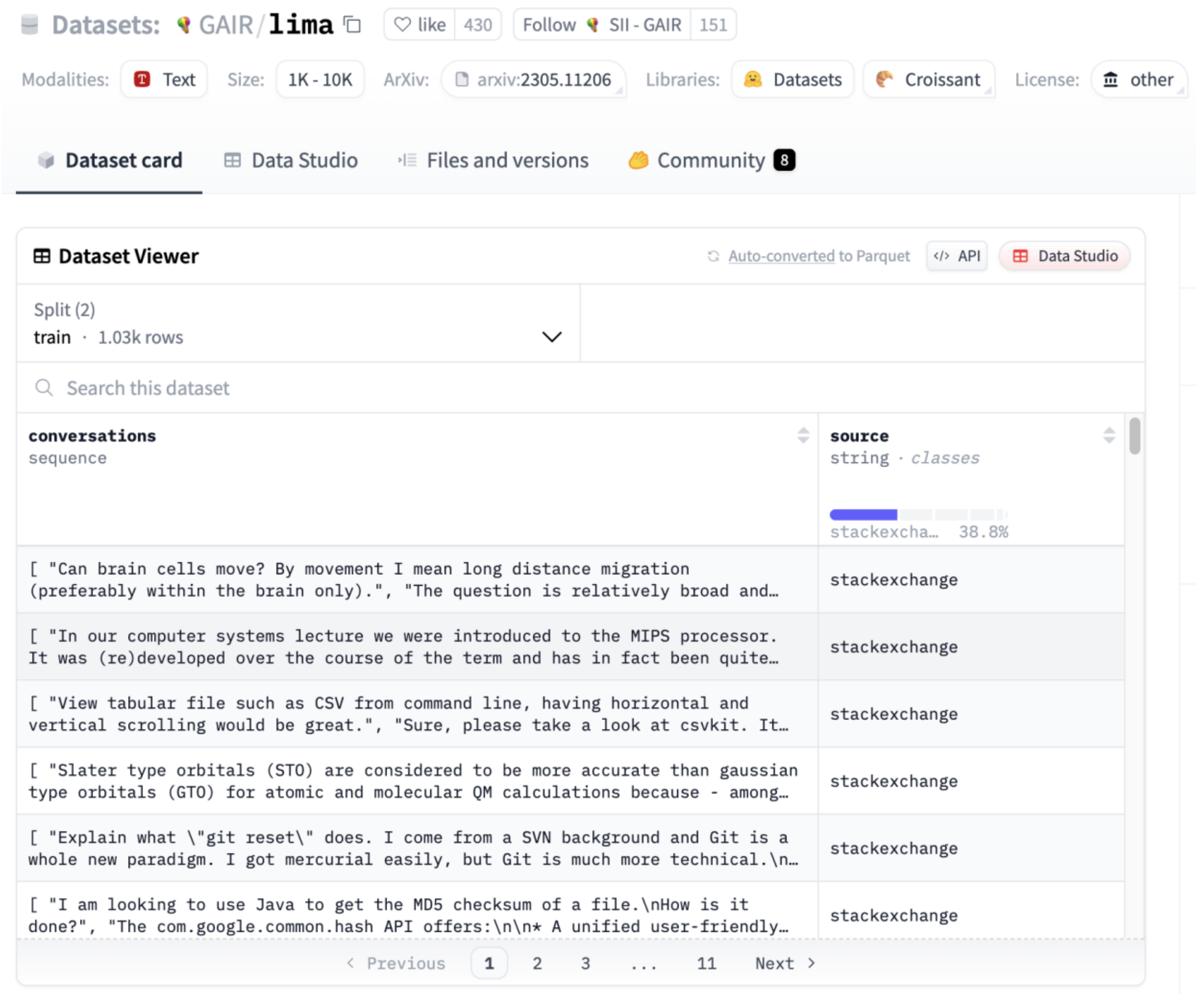
LIMA: Less Is More for Alignment

Published on May 18, 2023 · Submitted by akhaliq on May 21, 2023 #1 Paper of the day

Authors: Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, Omer Levy

Abstract

Large language models are trained in two stages: (1) unsupervised pretraining from raw text, to learn general-purpose representations, and (2) large scale instruction tuning and reinforcement learning, to better align to end tasks and user preferences. We measure the relative importance of these two stages by training LIMA, a 65B parameter LLaMa language model fine-tuned with the standard supervised loss on only 1,000 carefully curated prompts and responses, without any reinforcement learning or human preference modeling. LIMA demonstrates remarkably strong performance, learning to follow specific response formats from only a handful of examples in the training data, including complex queries that range from planning trip itineraries to speculating about alternate history. Moreover, the model tends to generalize well to unseen tasks that did not appear in the training data. In a controlled human study, responses from LIMA are either equivalent or strictly preferred to GPT-4 in 43% of cases; this statistic is as high as 58% when compared to Bard and 65% versus DaVinci003, which was trained with human feedback. Taken together, these results strongly suggest that almost all knowledge in large language models is learned during pretraining, and only limited instruction tuning data is necessary to teach models to produce high quality output.



Modalities: Text Size: 1K - 10K ArXiv: arxiv:2305.11206 Libraries: Datasets Croissant License: other

Split (2)
train · 1.03k rows

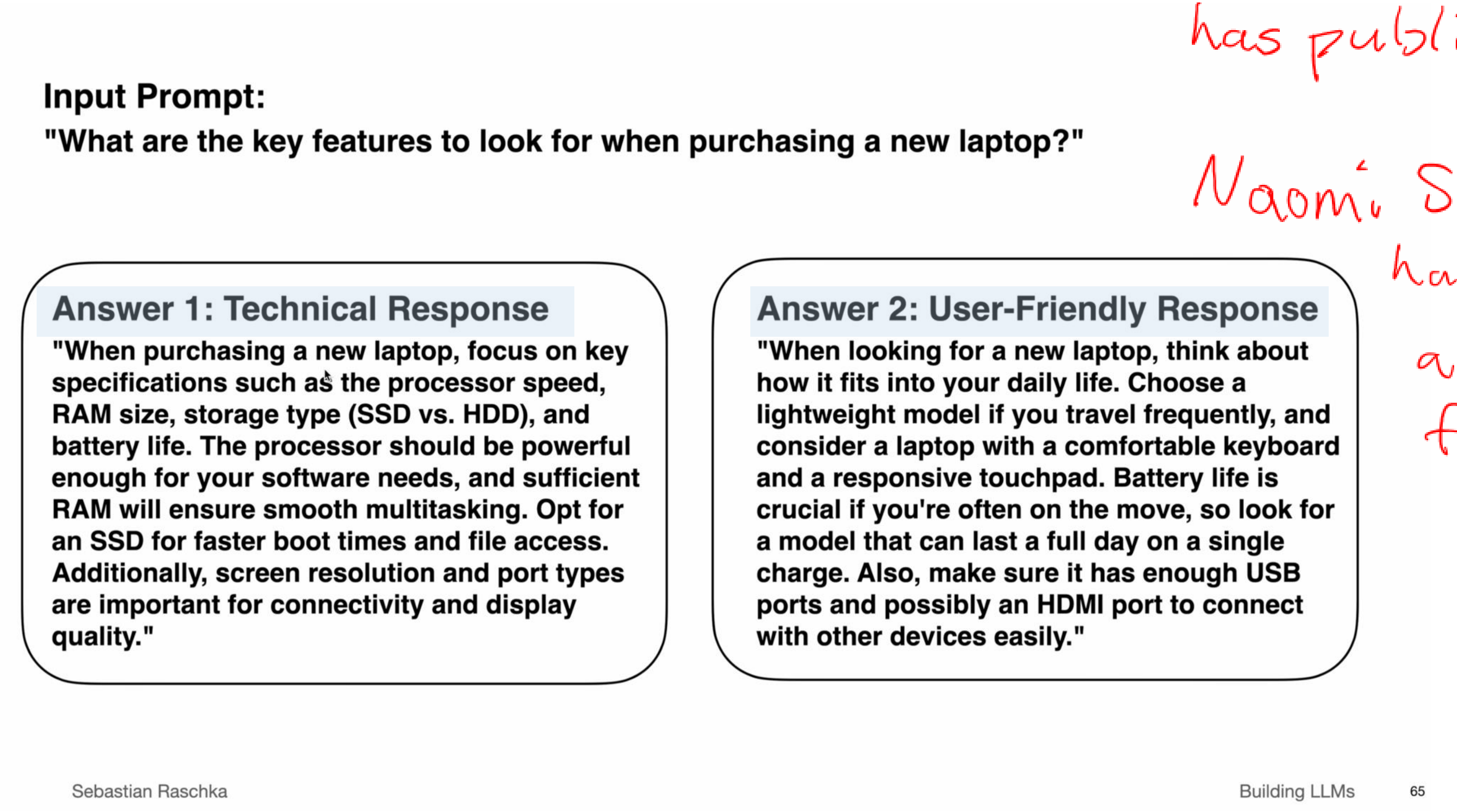
conversations sequence	source string · classes stackexcha… 38.8%
["Can brain cells move? By movement I mean long distance migration (preferably within the brain only).", "The question is relatively broad and…	stackexchange
["In our computer systems lecture we were introduced to the MIPS processor. It was (re)developed over the course of the term and has in fact been quite…	stackexchange
["View tabular file such as CSV from command line, having horizontal and vertical scrolling would be great.", "Sure, please take a look at csvkit. It…	stackexchange
["Slater type orbitals (STO) are considered to be more accurate than gaussian type orbitals (GTO) for atomic and molecular QM calculations because - among…	stackexchange
["Explain what \"git reset\" does. I come from a SVN background and Git is a whole new paradigm. I got mercurial easily, but Git is much more technical.\n…	stackexchange
["I am looking to use Java to get the MD5 checksum of a file.\nHow is it done?", "The com.google.common.hash API offers:\n\n* A unified user-friendly…	stackexchange

Preference Fine Tuning

- You can further finetune on preferences, safety or other aspects.

- Understanding tradeoffs from finetuning is an open research question.
  - Some finetuning can cause forgetting of “base knowledge”.
  - Some finetuning can reinforce concepts to be avoided.

Refine responses for style or safety
Reward preferred responses



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."

Sebastian Raschka

Building LLMs 65

Aldo Pacchiano has published on this.

Naomi Saphra has written about football fan bias.

S. Raschka, "Developing an LLM: Building, Training, Finetuning"

Any questions?

???

Moving on

- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)

Generative LLM Evaluations

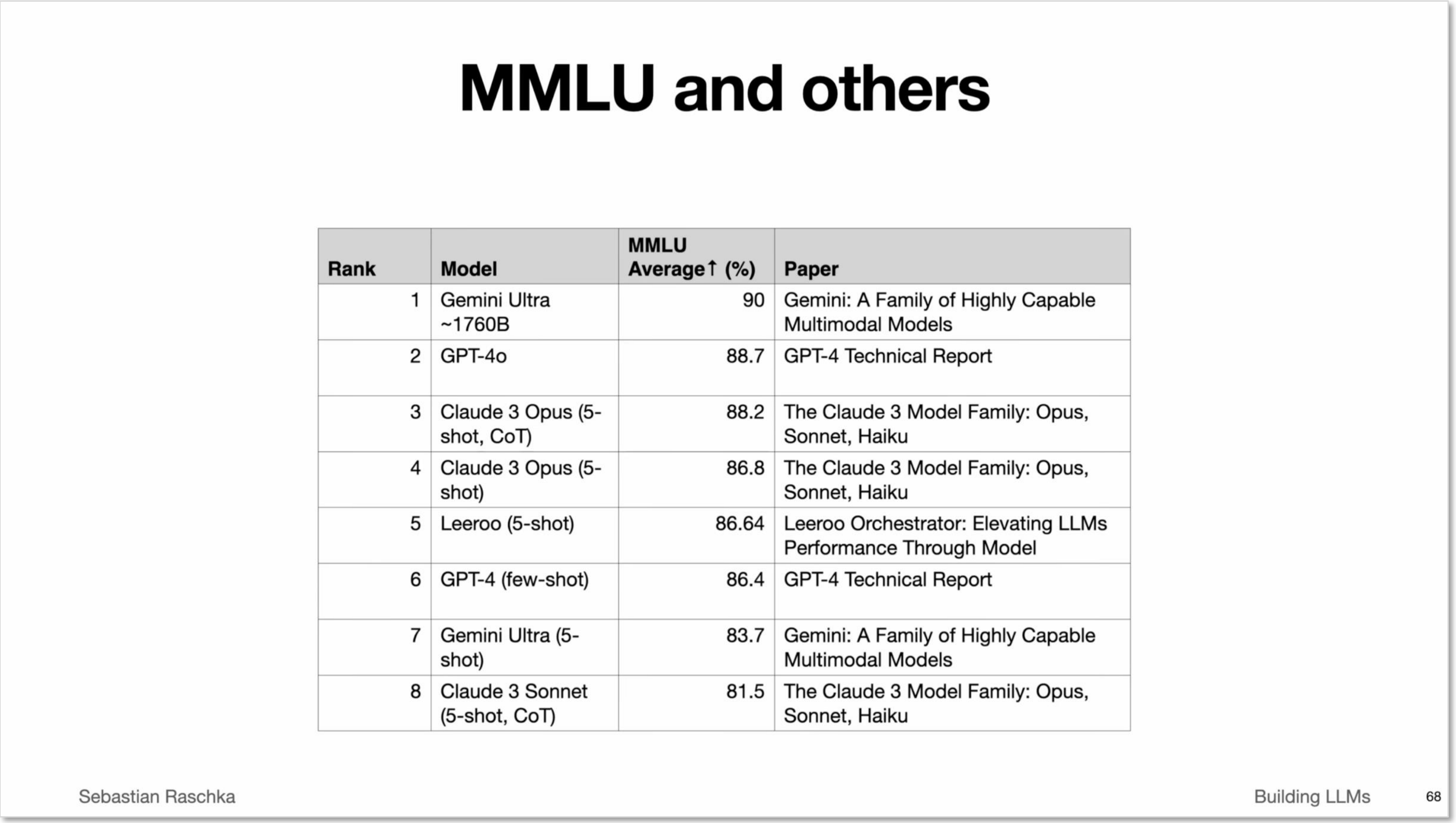
Evaluate for

- Accuracy (is it factual or hallucinated?)
- Relevance (is it answering the question?)
- Bias, Toxicity (Is it fair? Or even worse is it racist or toxic?)
- Diversity of Response (does it always give same response? or equally useful diverse responses?)

Ways to Evaluate

- Find a benchmark that matches your task
  - HellaSwag *(which evaluates how well an LLM can complete a sentence),*
  - TruthfulQA *(measuring truthfulness of model responses), and*
  - MMLU *(which measures how well the LLM can multitask),*
  - WinoGrande *(commonsense reasoning),*
  - GSM8K, *(arithmetic reasoning), etc.*
- Create your own evaluation prompt/response pairs –
  - need thousands!
- Use an LLM to evaluate your LLM!

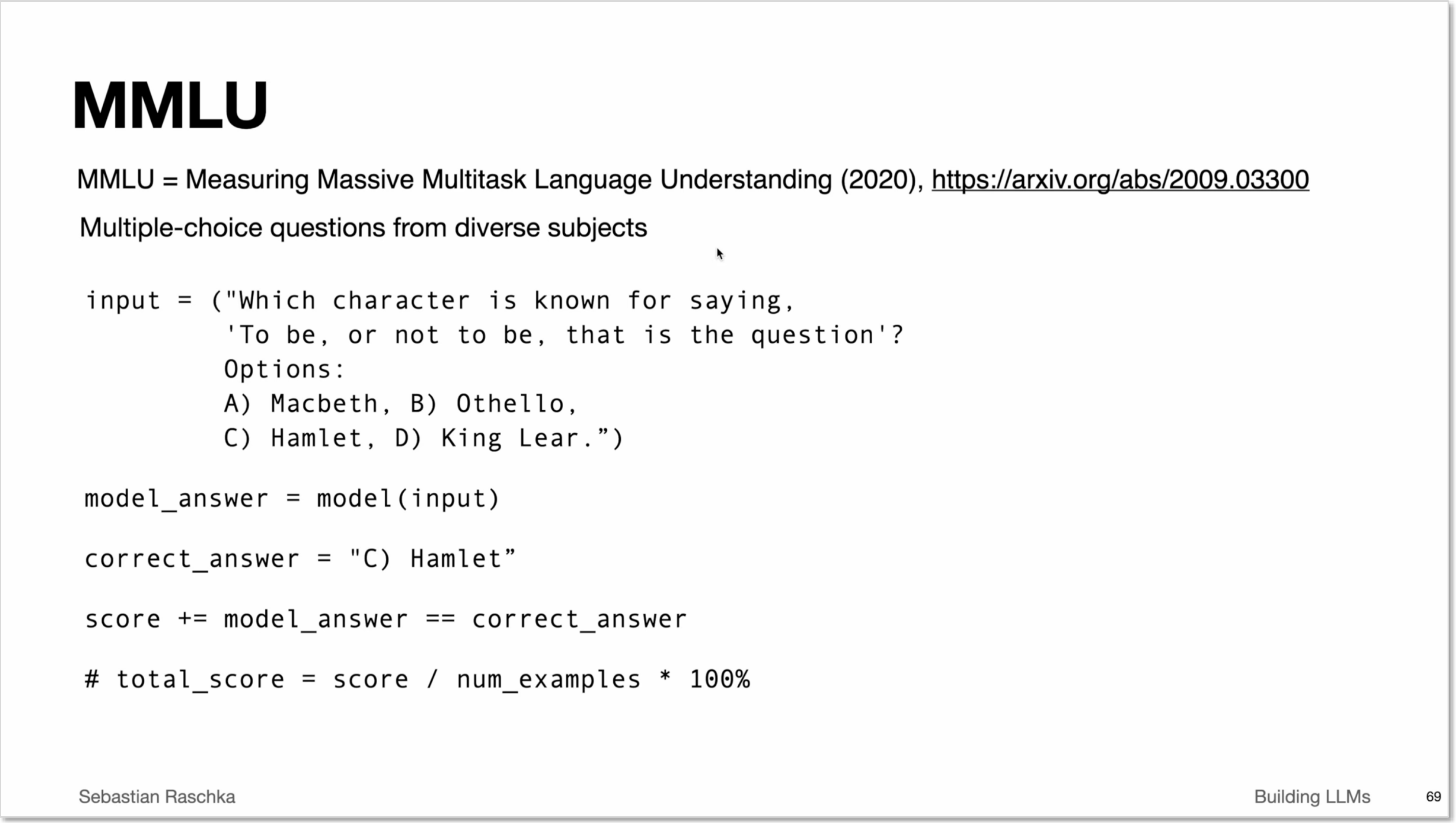
See: https://arize.com/blog-course/llm-evaluation-the-definitive-guide/ for a nice overview



MMLU and others

Rank	Model	MMLU Average↑ (%)	Paper
1	Gemini Ultra ~1760B	90	Gemini: A Family of Highly Capable Multimodal Models
2	GPT-4o	88.7	GPT-4 Technical Report
3	Claude 3 Opus (5-shot, CoT)	88.2	The Claude 3 Model Family: Opus, Sonnet, Haiku
4	Claude 3 Opus (5-shot)	86.8	The Claude 3 Model Family: Opus, Sonnet, Haiku
5	Leeroo (5-shot)	86.64	Leeroo Orchestrator: Elevating LLMs Performance Through Model
6	GPT-4 (few-shot)	86.4	GPT-4 Technical Report
7	Gemini Ultra (5-shot)	83.7	Gemini: A Family of Highly Capable Multimodal Models
8	Claude 3 Sonnet (5-shot, CoT)	81.5	The Claude 3 Model Family: Opus, Sonnet, Haiku

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



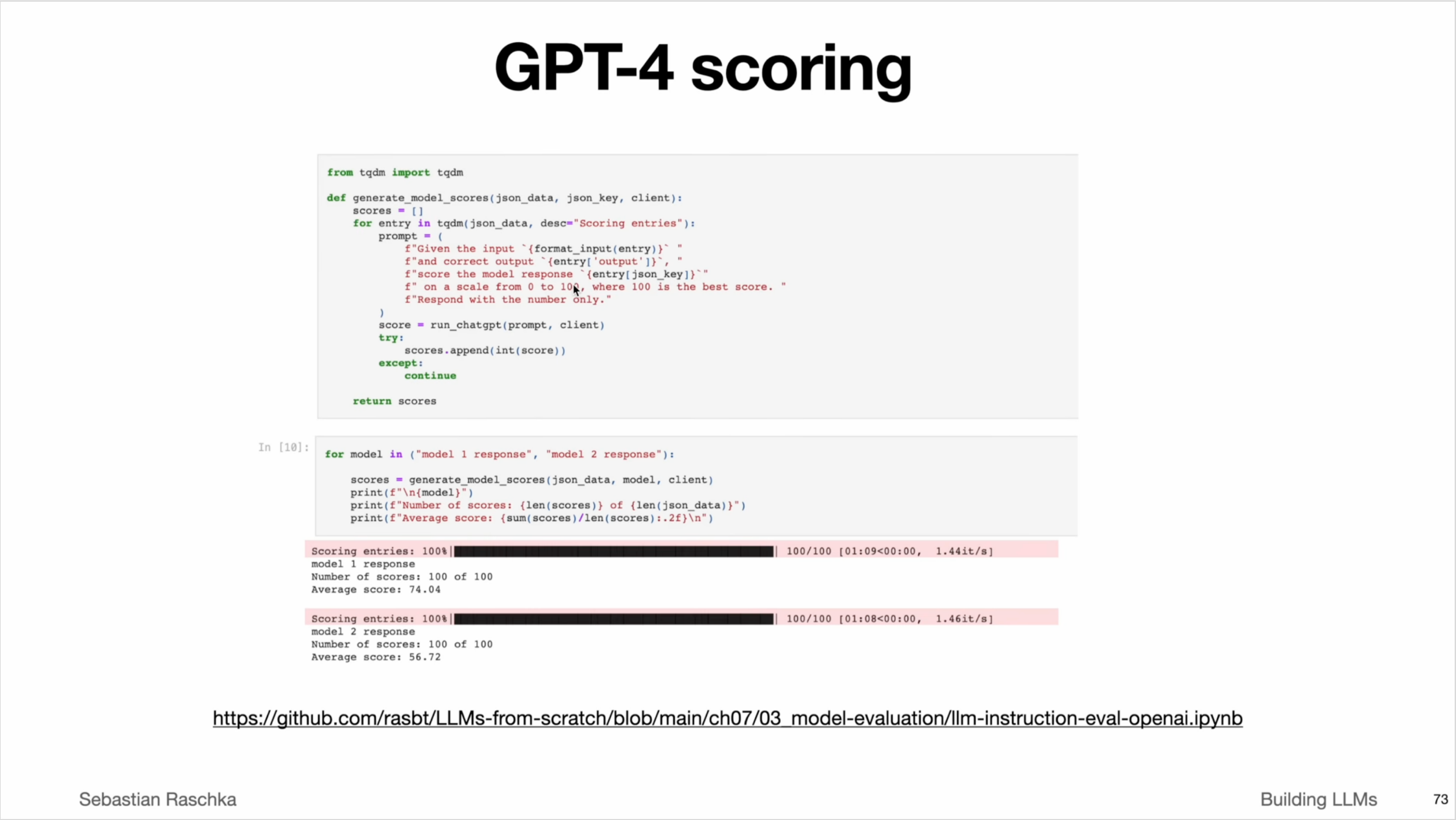
MMLU

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%
```

S. Raschka, "Developing an LLM: Building, Training, Finetuning"



GPT-4 scoring

```python
from tqdm import tqdm

def generate_model_scores(json_data, json_key, client):
    scores = []
    for entry in tqdm(json_data, desc="Scoring entries"):
        prompt = (
            f"Given the input `{format_input(entry)}` "
            f"and correct output `{entry['output']}`, "
            f"score the model response `{entry[json_key]}`"
            f" on a scale from 0 to 100, where 100 is the best score. "
            f"Respond with the number only."
        )
        score = run_chatgpt(prompt, client)
        try:
            scores.append(int(score))
        except:
            continue

    return scores
```

```python
In [10]: for model in ("model 1 response", "model 2 response"):

             scores = generate_model_scores(json_data, model, client)
             print(f"\n{model}")
             print(f"Number of scores: {len(scores)} of {len(json_data)}")
             print(f"Average score: {sum(scores)/len(scores):.2f}\n")
```

```
Scoring entries: 100%|██████████████████████████████████████████████████| 100/100 [01:09<00:00,  1.44it/s]
model 1 response
Number of scores: 100 of 100
Average score: 74.04

Scoring entries: 100%|██████████████████████████████████████████████████| 100/100 [01:08<00:00,  1.46it/s]
model 2 response
Number of scores: 100 of 100
Average score: 56.72
```

https://github.com/rasbt/LLMs-from-scratch/blob/main/ch07/03_model-evaluation/llm-instruction-eval-openai.ipynb

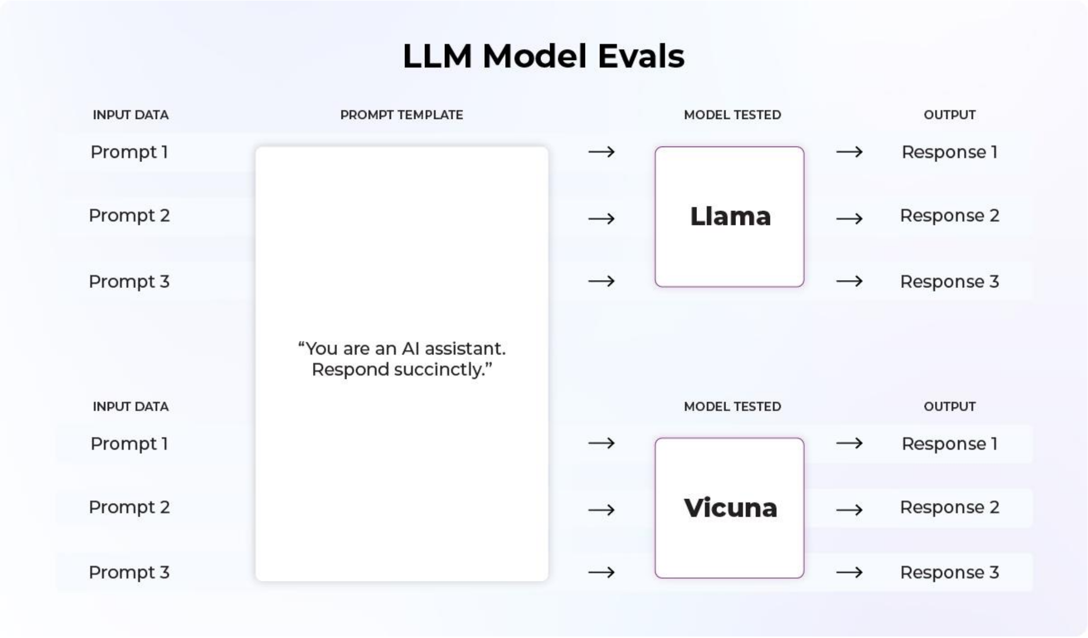
Sebastian Raschka

Building LLMs

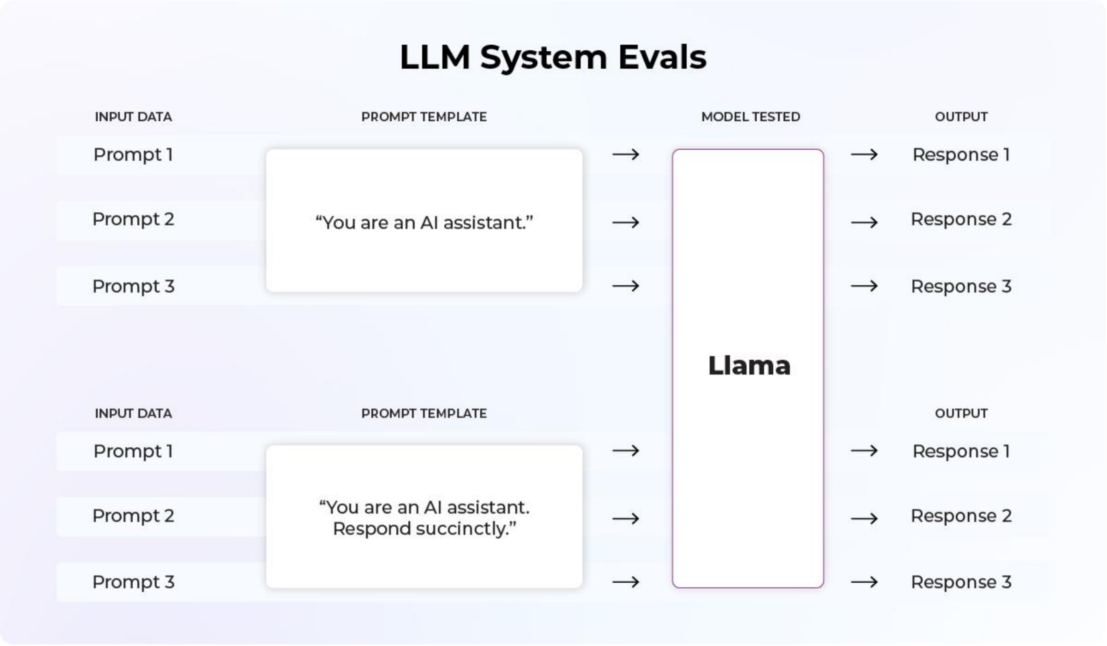
73

S. Raschka, "Developing an LLM: Building, Training, Finetuning"

Model vs System Evals



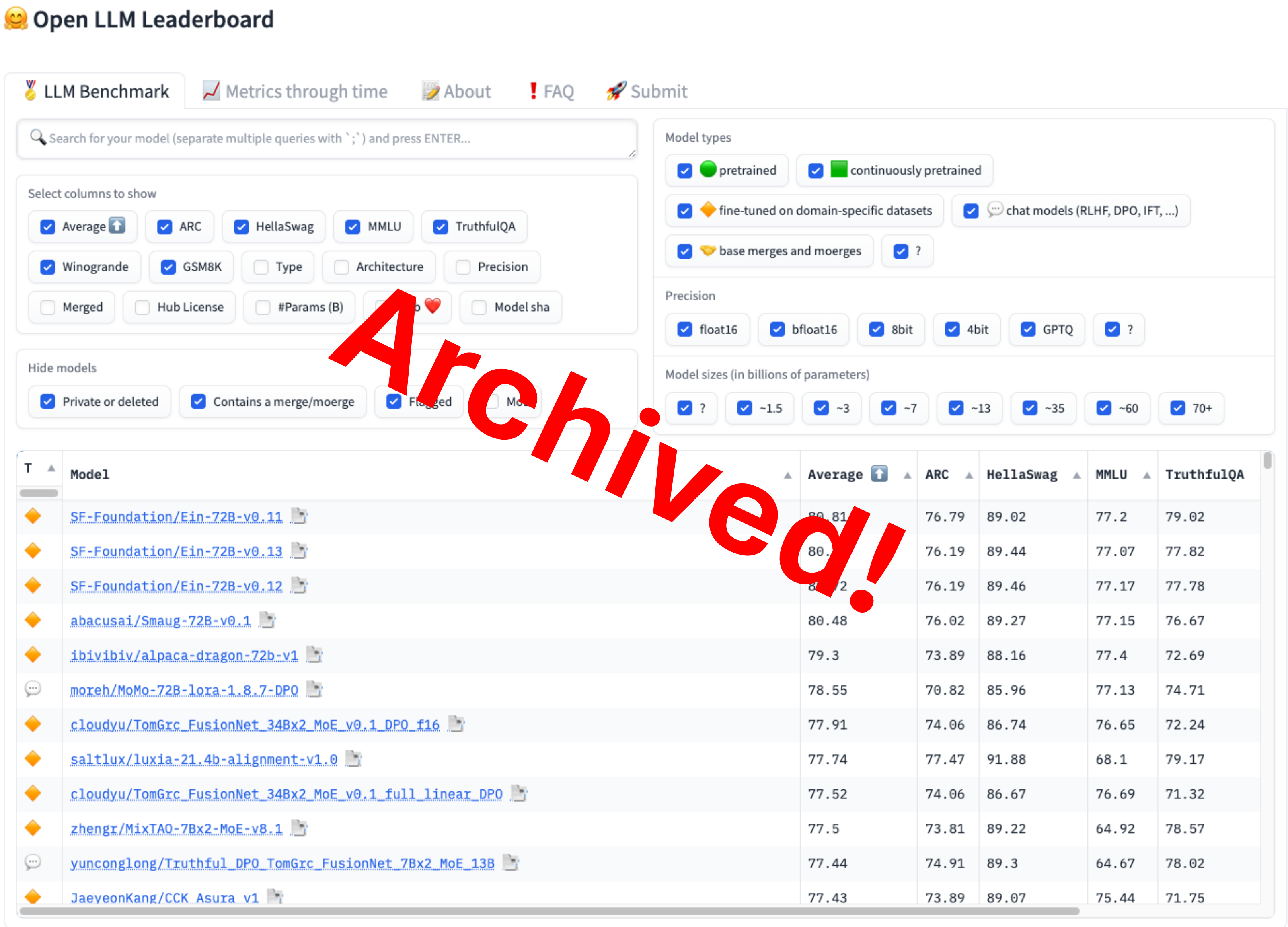
Useful for choosing a model or deciding when to switch.



Useful for prompt tuning and monitoring over time.

See: https://arize.com/blog-course/llm-evaluation-the-definitive-guide/ for a nice overview

Open LLM Leaderboard



Archived!

🤗 Open LLM Leaderboard

🏅 LLM Benchmark
📈 Metrics through time
📝 About
❗ FAQ
🚀 Submit
🔍 Search for your model (separate multiple queries with `;`) and press ENTER...

Select columns to show

☑ Average ⬆️ ☑ ARC ☑ HellaSwag ☑ MMLU ☑ TruthfulQA
☑ Winogrande ☑ GSM8K ☐ Type ☐ Architecture ☐ Precision
☐ Merged ☐ Hub License ☐ #Params (B) ☐ [illegible] ❤️ ☐ Model sha

Hide models

☑ Private or deleted ☑ Contains a merge/moerge ☑ Flagged ☐ Mo[illegible]

Model types

☑ 🟢 pretrained ☑ 🟩 continuously pretrained
☑ 🔶 fine-tuned on domain-specific datasets ☑ 💬 chat models (RLHF, DPO, IFT, ...)
☑ 🤝 base merges and moerges ☑ ?

Precision

☑ float16 ☑ bfloat16 ☑ 8bit ☑ 4bit ☑ GPTQ ☑ ?

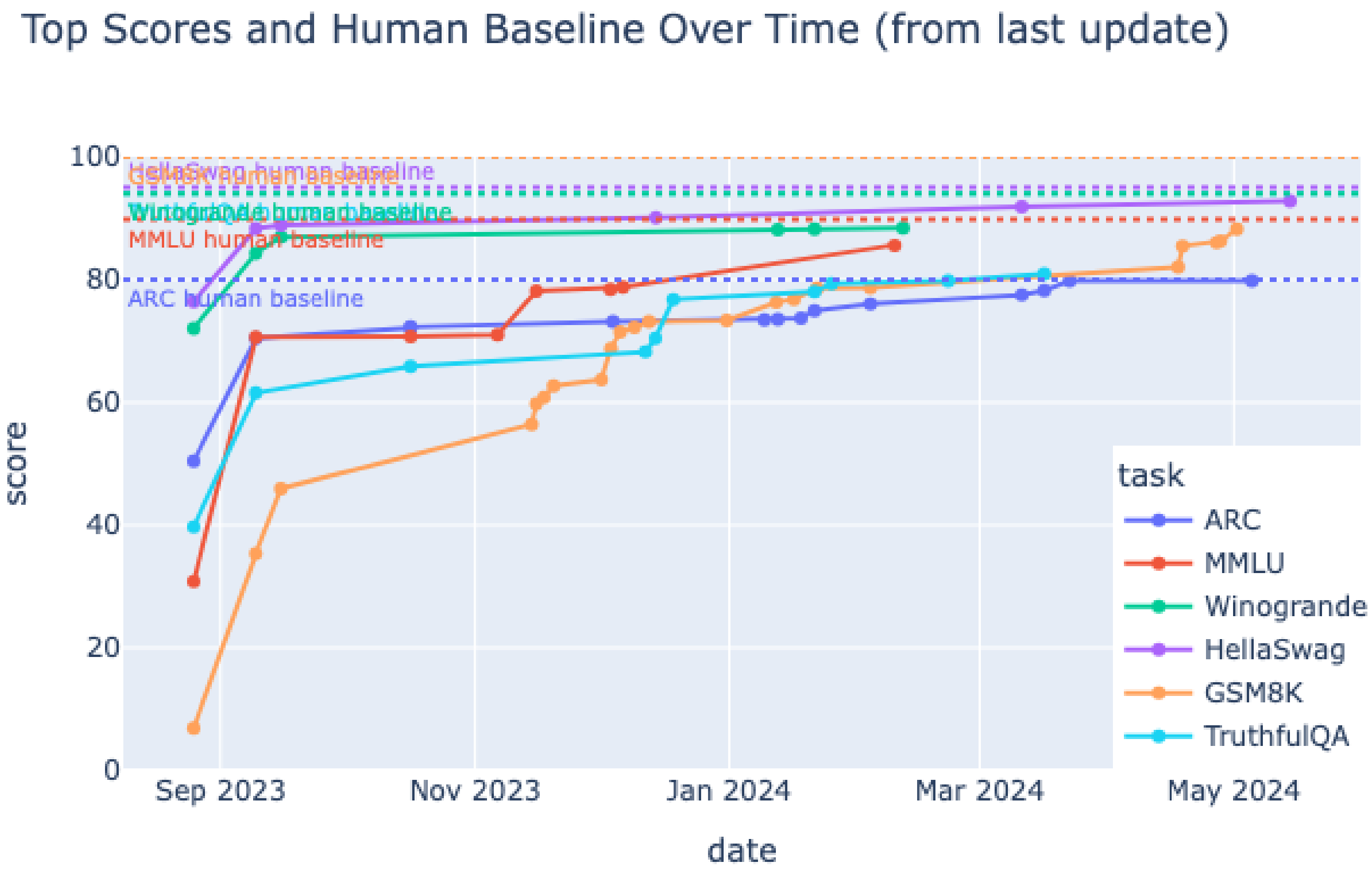
Model sizes (in billions of parameters)

☑ ? ☑ ~1.5 ☑ ~3 ☑ ~7 ☑ ~13 ☑ ~35 ☑ ~60 ☑ 70+

T	Model	Average ⬆️	ARC	HellaSwag	MMLU	TruthfulQA
🔶	SF-Foundation/Ein-72B-v0.11	80.81	76.79	89.02	77.2	79.02
🔶	SF-Foundation/Ein-72B-v0.13	[illegible]	76.19	89.44	77.07	77.82
🔶	SF-Foundation/Ein-72B-v0.12	[illegible]	76.19	89.46	77.17	77.78
🔶	abacusai/Smaug-72B-v0.1	80.48	76.02	89.27	77.15	76.67
🔶	ibivibiv/alpaca-dragon-72b-v1	79.3	73.89	88.16	77.4	72.69
💬	moreh/MoMo-72B-lora-1.8.7-DPO	78.55	70.82	85.96	77.13	74.71
🔶	cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_DPO_f16	77.91	74.06	86.74	76.65	72.24
🔶	saltlux/luxia-21.4b-alignment-v1.0	77.74	77.47	91.88	68.1	79.17
🔶	cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_full_linear_DPO	77.52	74.06	86.67	76.69	71.32
🔶	zhengr/MixTAO-7Bx2-MoE-v8.1	77.5	73.81	89.22	64.92	78.57
💬	yunconglong/Truthful_DPO_TomGrc_FusionNet_7Bx2_MoE_13B	77.44	74.91	89.3	64.67	78.02
🔶	JaeyeonKang/CCK_Asura_v1	77.43	73.89	89.07	75.44	71.75

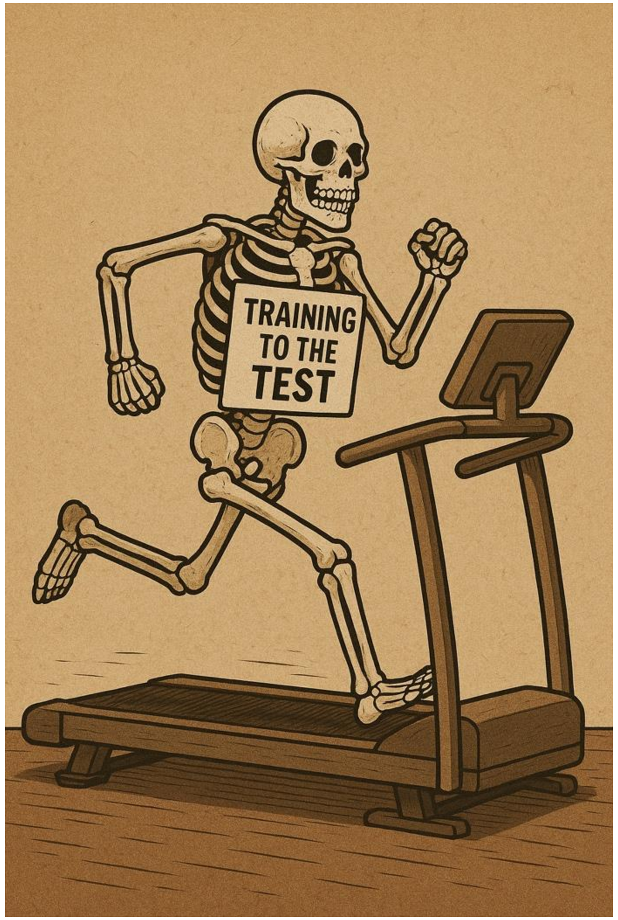
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

HF OpenLLM leaderboard became too easy
Models plateaued



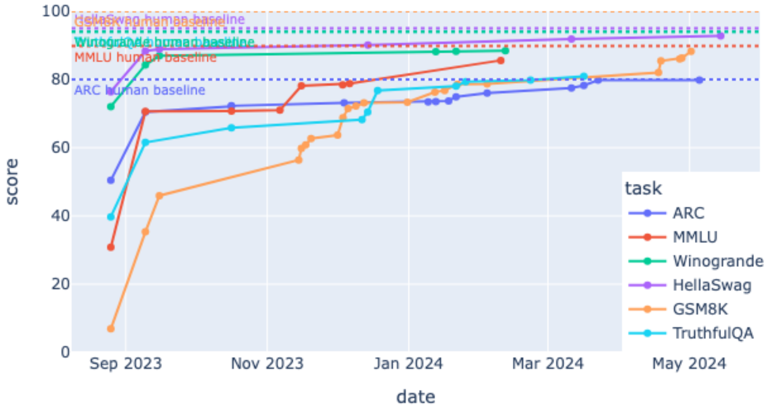
https://huggingface.co/spaces/open-llm-leaderboard/blog

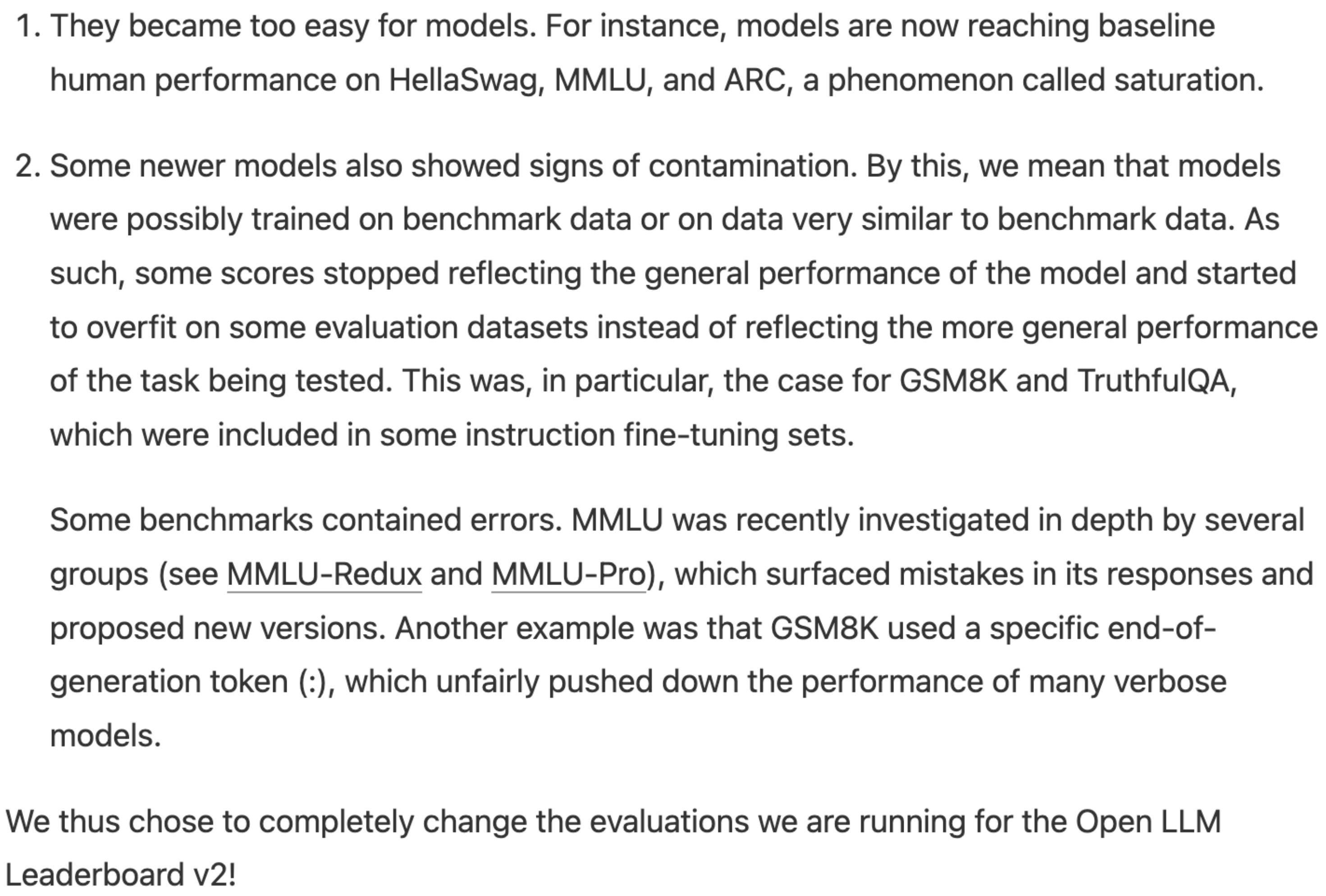
Common Problems with Static Evaluations



HF OpenLLM leaderboard became too easy
Models plateaued

Top Scores and Human Baseline Over Time (from last update)





1. They became too easy for models. For instance, models are now reaching baseline human performance on HellaSwag, MMLU, and ARC, a phenomenon called saturation.

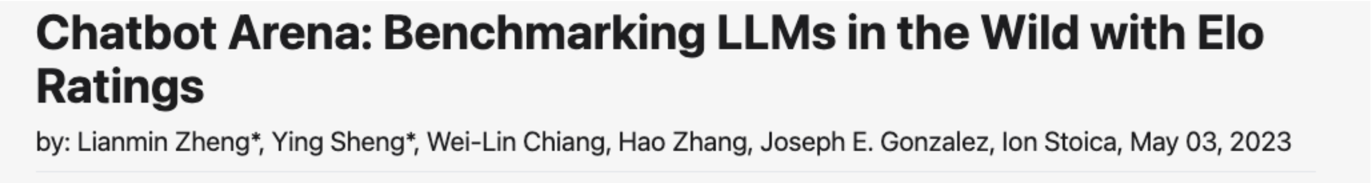
2. Some newer models also showed signs of contamination. By this, we mean that models were possibly trained on benchmark data or on data very similar to benchmark data. As such, some scores stopped reflecting the general performance of the model and started to overfit on some evaluation datasets instead of reflecting the more general performance of the task being tested. This was, in particular, the case for GSM8K and TruthfulQA, which were included in some instruction fine-tuning sets.

   Some benchmarks contained errors. MMLU was recently investigated in depth by several groups (see MMLU-Redux and MMLU-Pro), which surfaced mistakes in its responses and proposed new versions. Another example was that GSM8K used a specific end-of-generation token (:), which unfairly pushed down the performance of many verbose models.

We thus chose to completely change the evaluations we are running for the Open LLM Leaderboard v2!

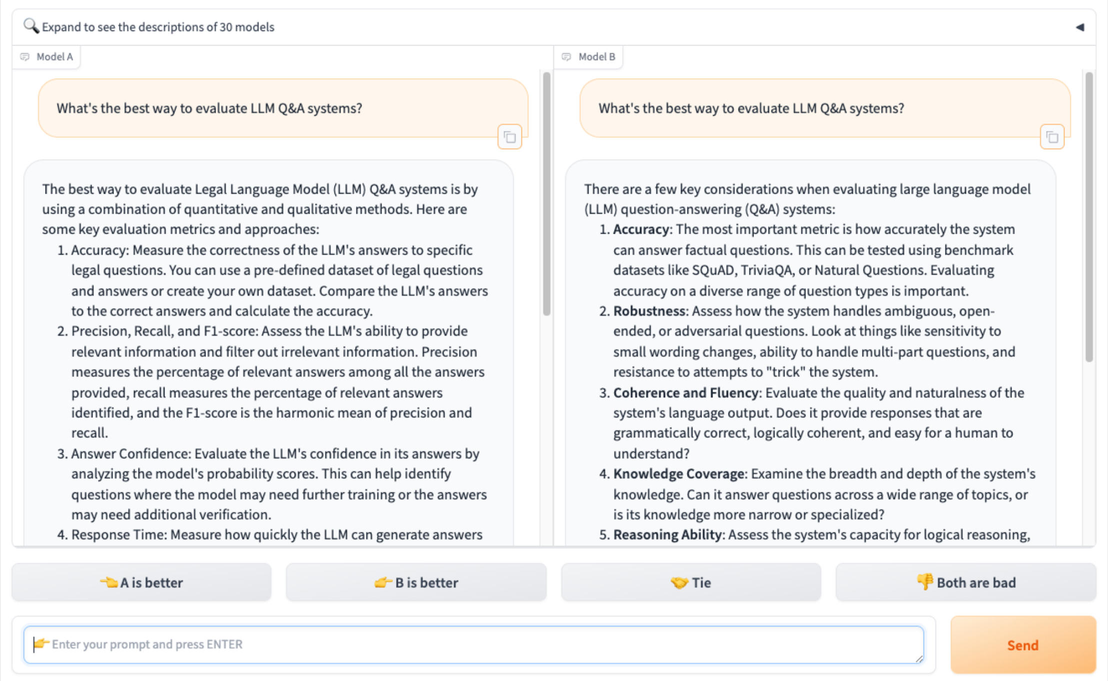
https://huggingface.co/spaces/open-llm-leaderboard/blog

Crowd-Sourcing Evaluations User Feedback



Chatbot Arena: Benchmarking LLMs in the Wild with Elo Ratings

by: Lianmin Zheng*, Ying Sheng*, Wei-Lin Chiang, Hao Zhang, Joseph E. Gonzalez, Ion Stoica, May 03, 2023


Expand to see the descriptions of 30 models
◀
Model A
Model B
A is better
B is better
Tie
Both are bad
Enter your prompt and press ENTER
Send

Model A

What's the best way to evaluate LLM Q&A systems?

The best way to evaluate Legal Language Model (LLM) Q&A systems is by using a combination of quantitative and qualitative methods. Here are some key evaluation metrics and approaches:

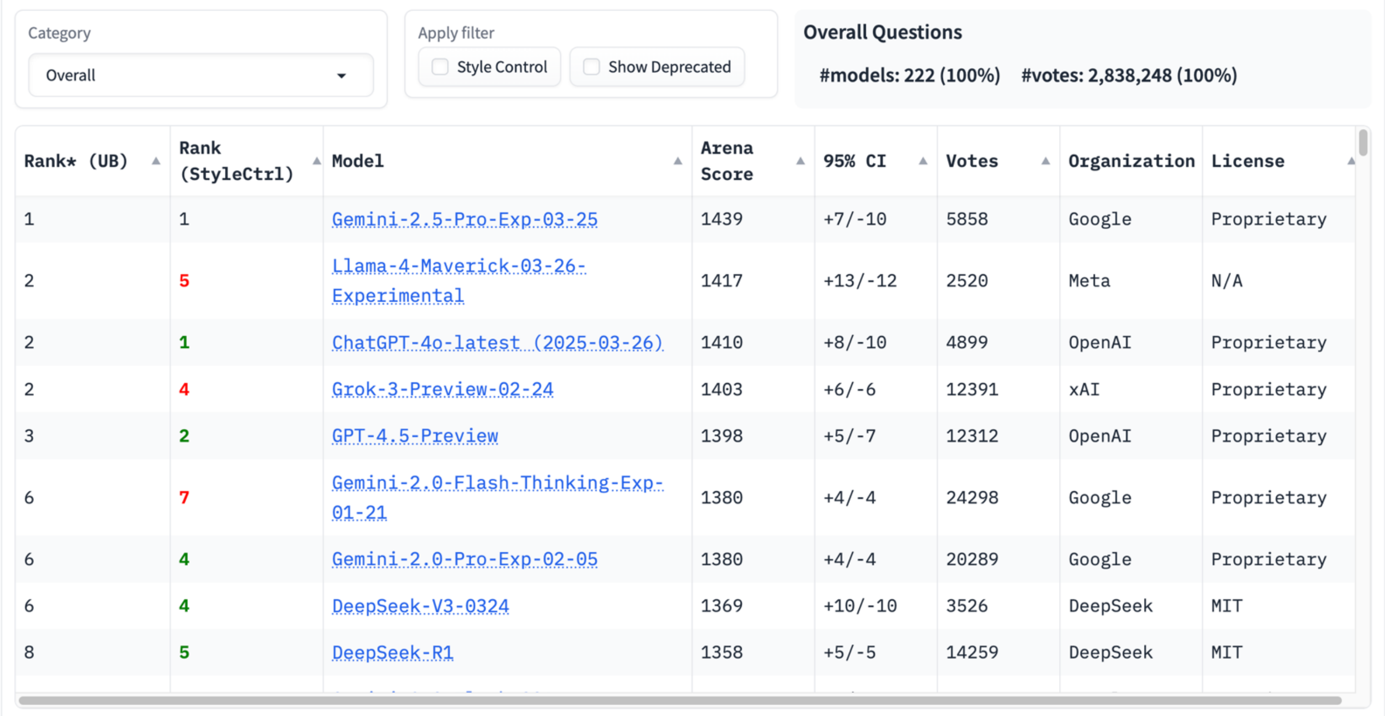
1. Accuracy: Measure the correctness of the LLM's answers to specific legal questions. You can use a pre-defined dataset of legal questions and answers or create your own dataset. Compare the LLM's answers to the correct answers and calculate the accuracy.
2. Precision, Recall, and F1-score: Assess the LLM's ability to provide relevant information and filter out irrelevant information. Precision measures the percentage of relevant answers among all the answers provided, recall measures the percentage of relevant answers identified, and the F1-score is the harmonic mean of precision and recall.
3. Answer Confidence: Evaluate the LLM's confidence in its answers by analyzing the model's probability scores. This can help identify questions where the model may need further training or the answers may need additional verification.
4. Response Time: Measure how quickly the LLM can generate answers

Model B

What's the best way to evaluate LLM Q&A systems?

There are a few key considerations when evaluating large language model (LLM) question-answering (Q&A) systems:

1. **Accuracy**: The most important metric is how accurately the system can answer factual questions. This can be tested using benchmark datasets like SQuAD, TriviaQA, or Natural Questions. Evaluating accuracy on a diverse range of question types is important.
2. **Robustness**: Assess how the system handles ambiguous, open-ended, or adversarial questions. Look at things like sensitivity to small wording changes, ability to handle multi-part questions, and resistance to attempts to "trick" the system.
3. **Coherence and Fluency**: Evaluate the quality and naturalness of the system's language output. Does it provide responses that are grammatically correct, logically coherent, and easy for a human to understand?
4. **Knowledge Coverage**: Examine the breadth and depth of the system's knowledge. Can it answer questions across a wide range of topics, or is its knowledge more narrow or specialized?
5. **Reasoning Ability**: Assess the system's capacity for logical reasoning,



Category: Overall

Apply filter: ☐ Style Control ☐ Show Deprecated

Overall Questions

#models: 222 (100%) **#votes: 2,838,248 (100%)**

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	Gemini-2.5-Pro-Exp-03-25	1439	+7/-10	5858	Google	Proprietary
2	5	Llama-4-Maverick-03-26-Experimental	1417	+13/-12	2520	Meta	N/A
2	1	ChatGPT-4o-latest (2025-03-26)	1410	+8/-10	4899	OpenAI	Proprietary
2	4	Grok-3-Preview-02-24	1403	+6/-6	12391	xAI	Proprietary
3	2	GPT-4.5-Preview	1398	+5/-7	12312	OpenAI	Proprietary
6	7	Gemini-2.0-Flash-Thinking-Exp-01-21	1380	+4/-4	24298	Google	Proprietary
6	4	Gemini-2.0-Pro-Exp-02-05	1380	+4/-4	20289	Google	Proprietary
6	4	DeepSeek-V3-0324	1369	+10/-10	3526	DeepSeek	MIT
8	5	DeepSeek-R1	1358	+5/-5	14259	DeepSeek	MIT

https://lmarena.ai/?leaderboard

Any questions?

???

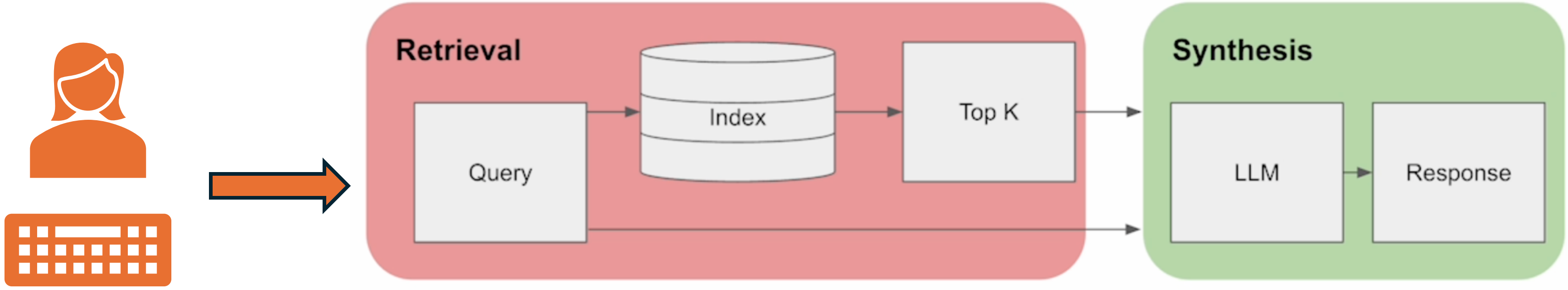
Moving on

- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)

Retrieval-Augmented Generation (RAG)

RAG enhances LLMs by referencing external knowledge to generate relevant responses.

- Integrates external data into LLM text generation.
- Reduces hallucination, improves response relevance.
- Works with
    - Unstructured data (e.g. documents)
    - Structured data (e.g. SQL data)
    - Code (e.g. python)



RAG Architecture

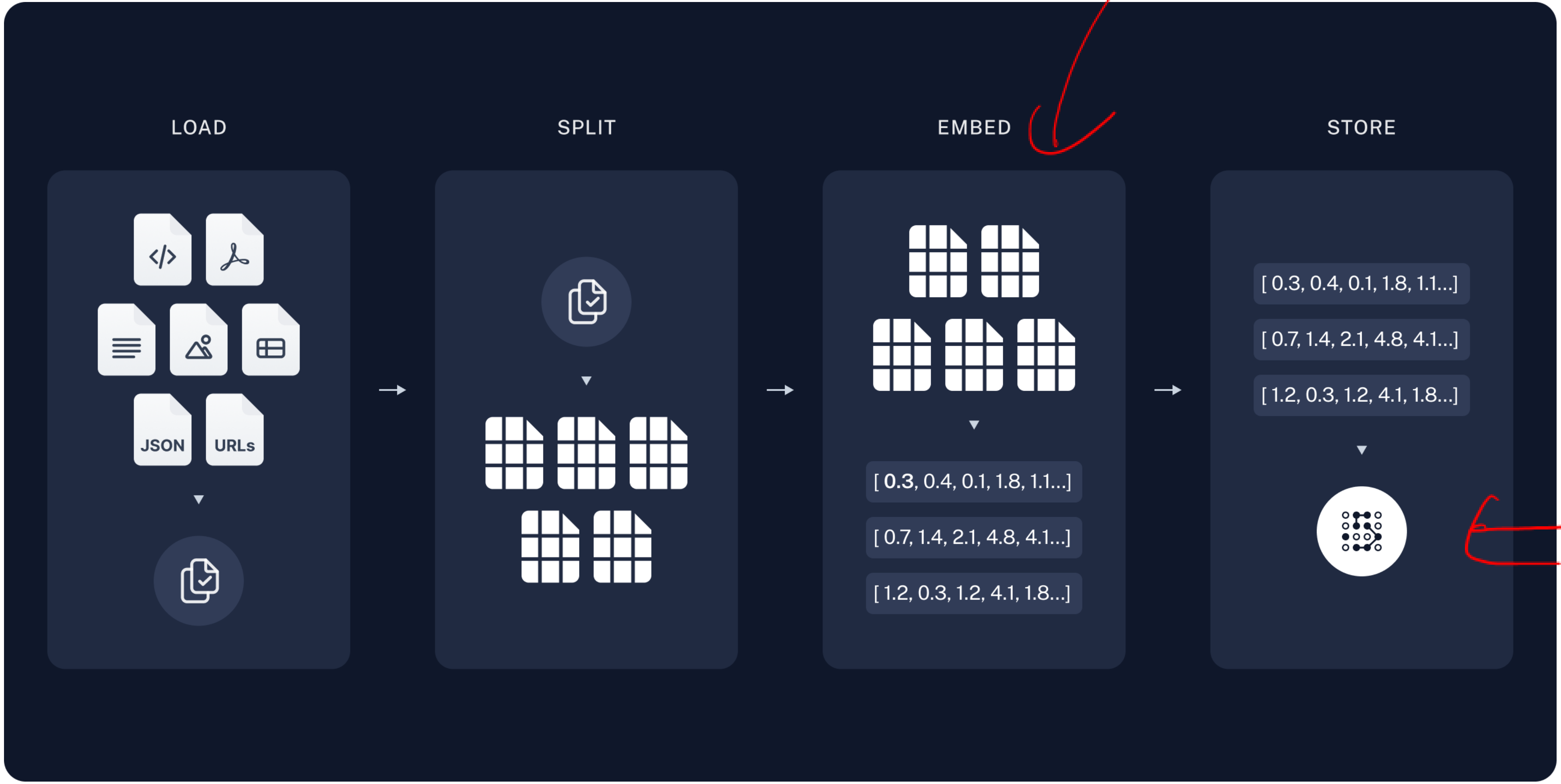
Typical RAG application has two main components:

- Loading and Indexing:
  - A pipeline for ingesting data from a source and indexing it
  - Usually happens offline
- Retrieval and Generation:
  - Takes user query at run time and retrieves relevant data from the index and passes it to the model

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Loading and Indexing

hidden values previously used as classification inputs



LOAD

JSON

URLs

SPLIT

EMBED

[0.3, 0.4, 0.1, 1.8, 1.1...]

[0.7, 1.4, 2.1, 4.8, 4.1...]

[1.2, 0.3, 1.2, 4.1, 1.8...]

STORE

[0.3, 0.4, 0.1, 1.8, 1.1...]

[0.7, 1.4, 2.1, 4.8, 4.1...]

[1.2, 0.3, 1.2, 4.1, 1.8...]

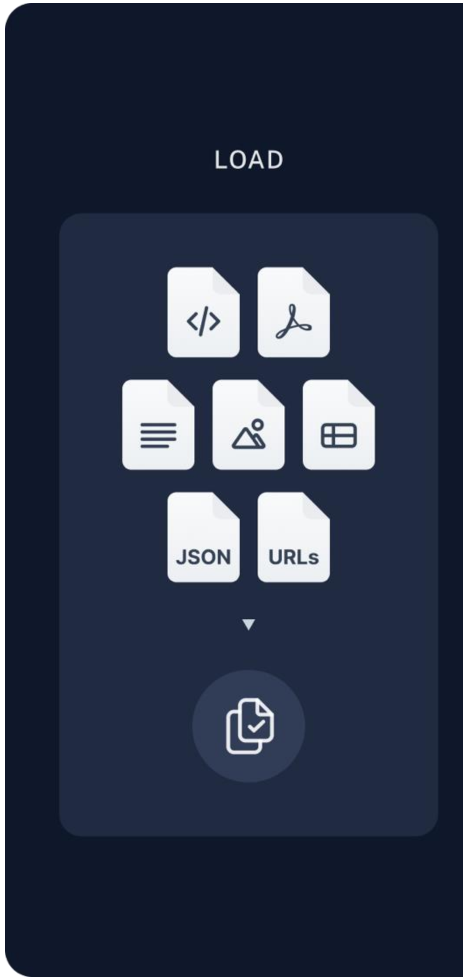
vector database

⇓

store + find vectors

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Load



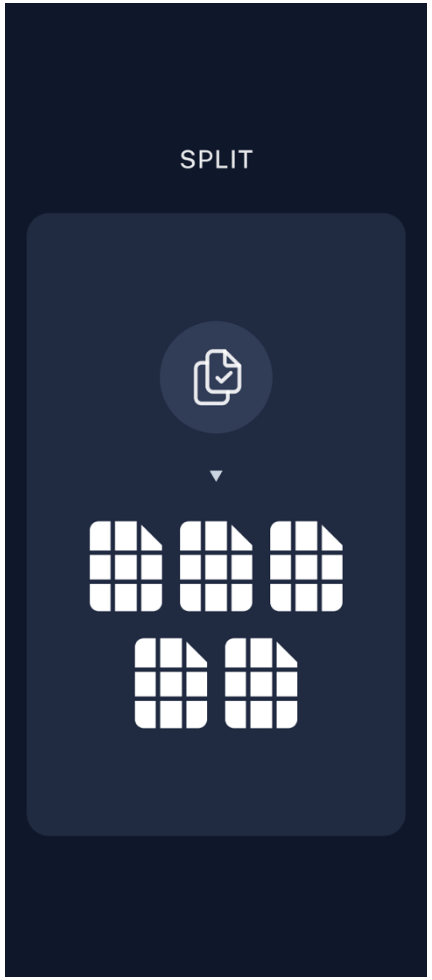
Load the data, e.g.

- PDFs
- HTML
- Plain text
- Images, video, audio
- Structured data (SQL, CSV/TSV, ...)
- JSON
- URLs
- ...

See for example: https://python.langchain.com/docs/modules/data_connection/document_loaders/

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Split



Break large documents into smaller chunks.

Easier to:

- index
- pass to model
- search
- fit into model's context window

limit on LLM input size

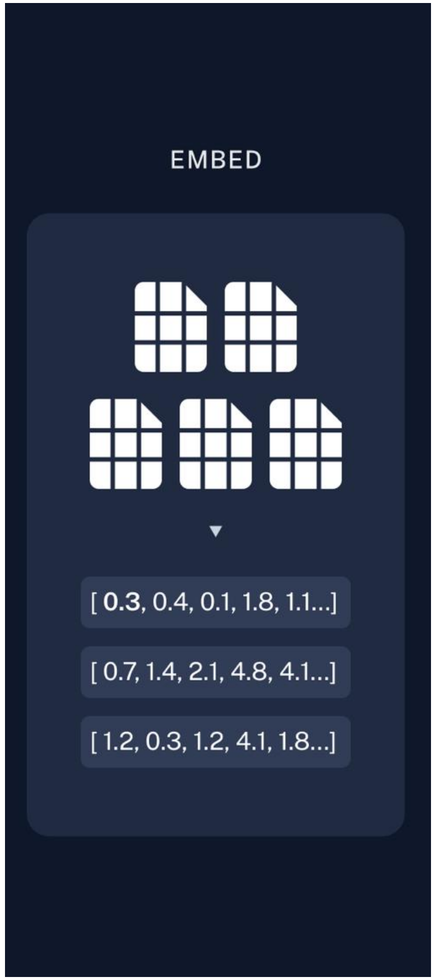
See for example: https://python.langchain.com/docs/modules/data_connection/document_transformers/

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Embed

- Encode (e.g. with Byte Pair Encoding) and
- Transform to embedding vectors with the learned embedding model.

usually LLM last hidden states



[0.3, 0.4, 0.1, 1.8, 1.1...]

[0.7, 1.4, 2.1, 4.8, 4.1...]

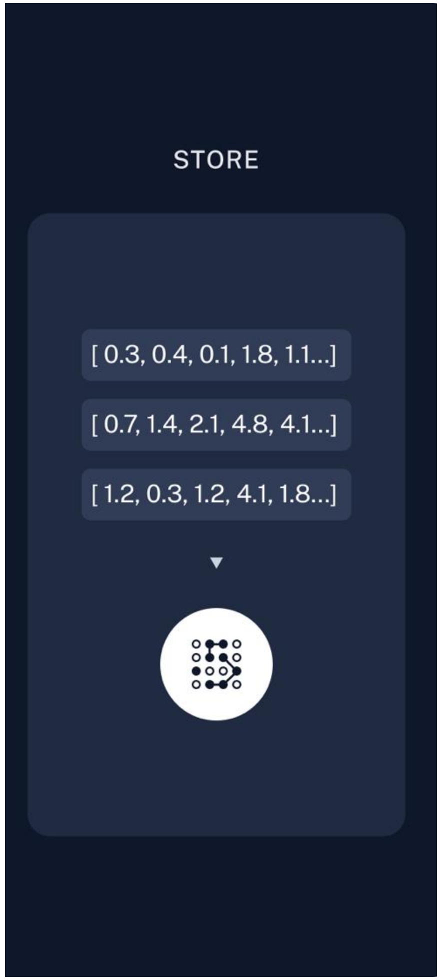
[1.2, 0.3, 1.2, 4.1, 1.8...]

See for example: https://python.langchain.com/docs/modules/data_connection/text_embedding/

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Store

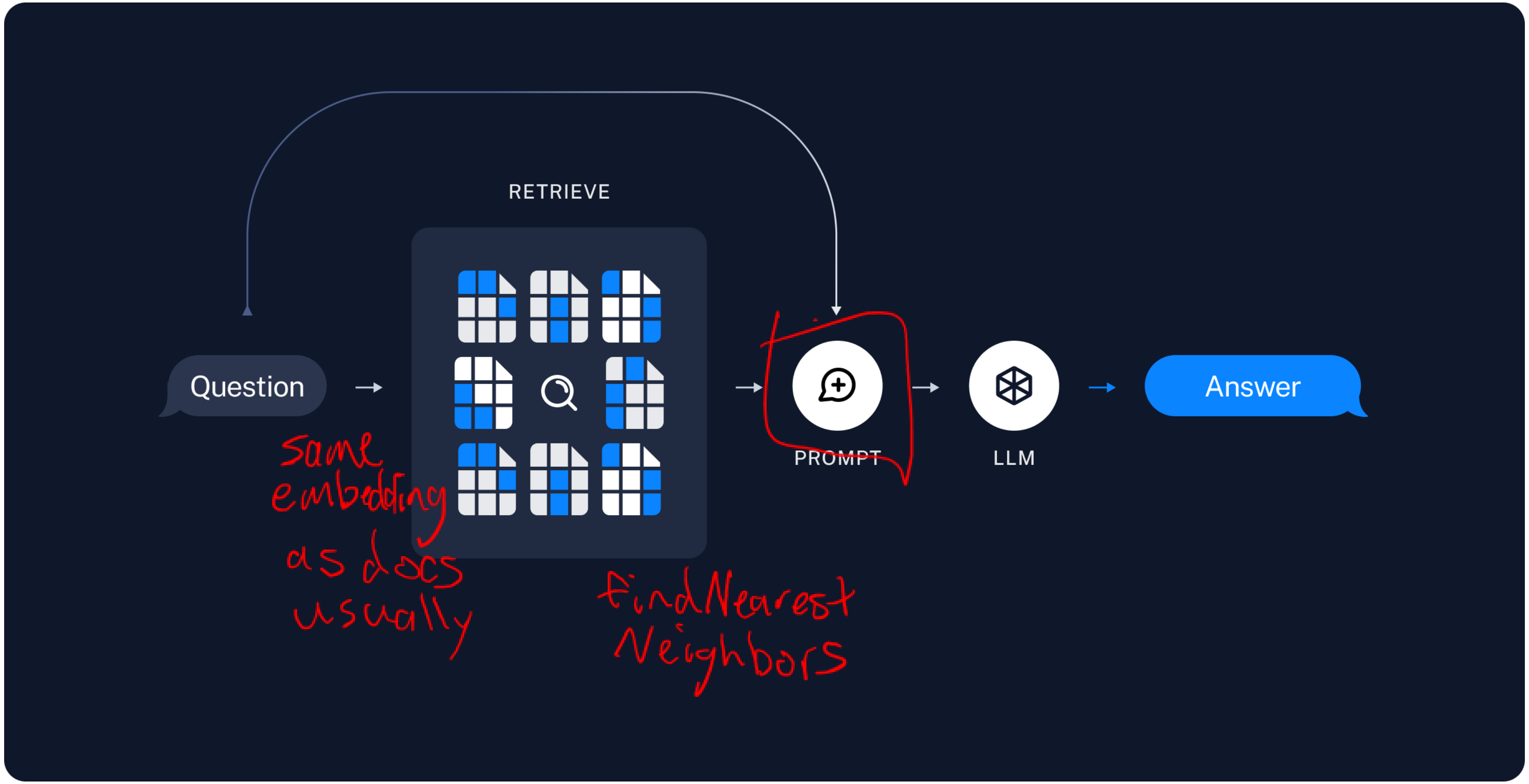
- Store the data in some kind of Vector Store
- e.g. Chroma, FAISS, Lance, Pinecone, etc…



See for example: https://python.langchain.com/docs/modules/data_connection/vectorstores/

https://python.langchain.com/docs/use_cases/question_answering/

RAG – Retrieval and Generation

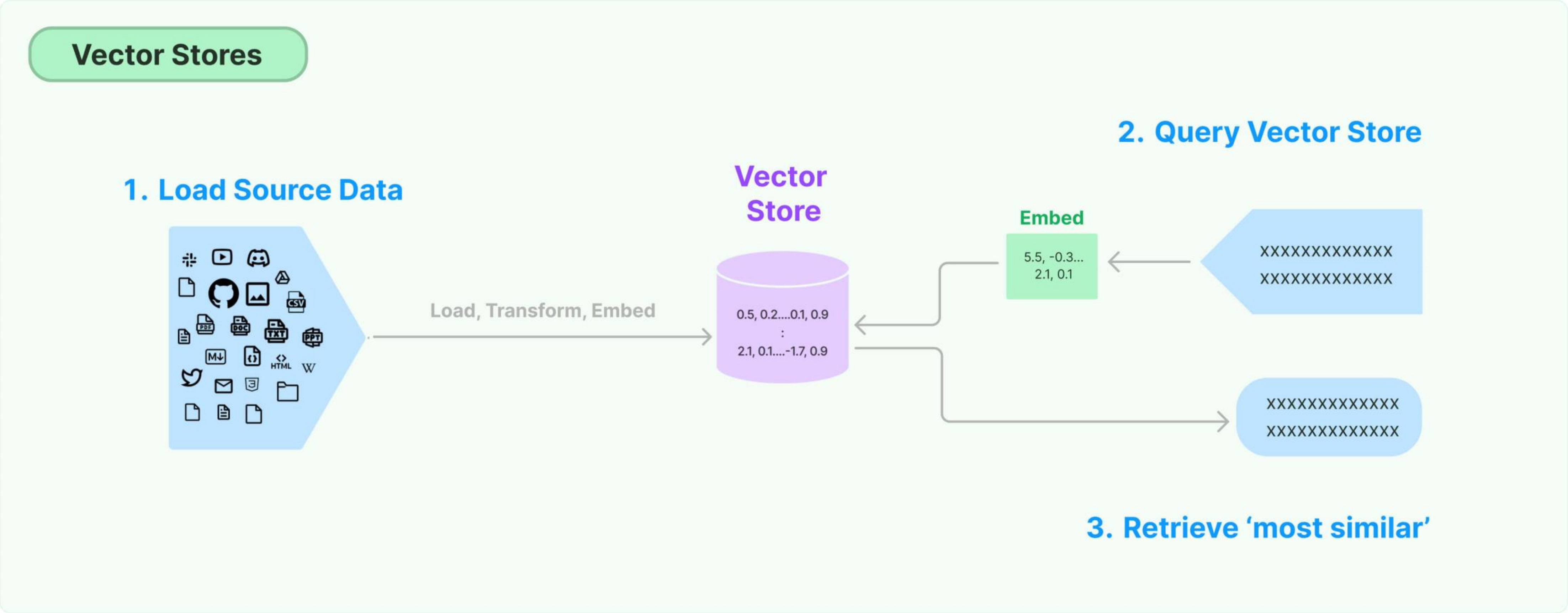


same embedding as docs usually

findNearest Neighbors

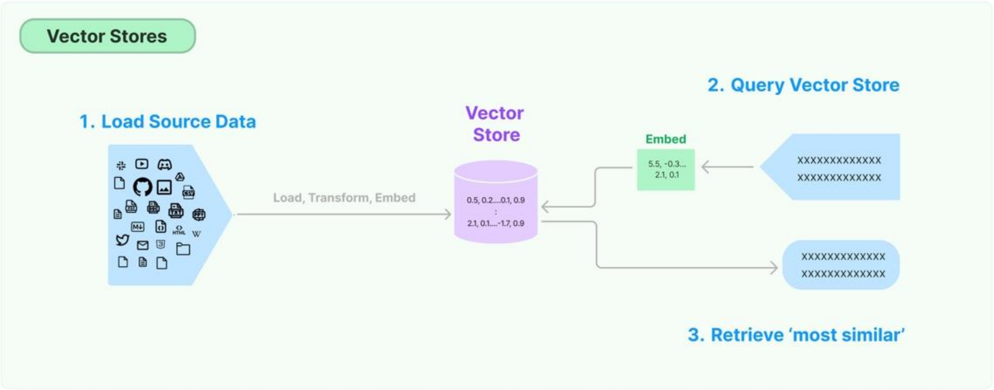
https://python.langchain.com/docs/use_cases/question_answering/

RAG – Retrieval



https://python.langchain.com/docs/modules/data_connection/vectorstores/

RAG – Retrieval Similarity Measure



L2 Norm*: $d = \sum_i (A_i - B_i)^2$

Inner Product: $d = 1 - \sum_i (A_i \times B_i)$

Cosine Similarity: $1 - \frac{\sum_i (A_i \times B_i)}{\sqrt{\sum_i (A_i^2)}\sqrt{\sum_i (B_i^2}}$

same when unit length

Most common:
Normalize to unit length

Nearest Neighbor

* Default on Chroma Vector Database

https://docs.trychroma.com/usage-guide#changing-the-distance-function

RAG – Other Query-Document Matching Approaches

1. BERT and Variants for Query-Document Matching

BERT:

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. *This foundational paper introduces BERT and its methodology for language understanding, which has been widely applied to information retrieval tasks*.

Application in Information Retrieval:

Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv:1901.04085. *This work explores how BERT can be used for re-ranking search results, demonstrating its effectiveness in improving information retrieval systems*. https://arxiv.org/abs/1901.04085

2. Fine-tuning for Specific Tasks

Fine-Tuning BERT for Search:

MacAvaney, S., Cohan, A., & Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. SIGIR. *This paper discusses fine-tuning BERT with contextual embeddings specifically for document ranking, providing insights into adapting Transformer modelsfor search tasks*. https://dl.acm.org/doi/abs/10.1145/3331184.3331317

3. Dual-encoder and Cross-encoder Architectures

Dual-Encoders for Efficient Retrieval:

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP. This paper introduces a method using dense vector representations for passages and questions to improve open-domain question answering. https://arxiv.org/abs/2004.04906

Cross-Encoders for Detailed Similarity Scoring:

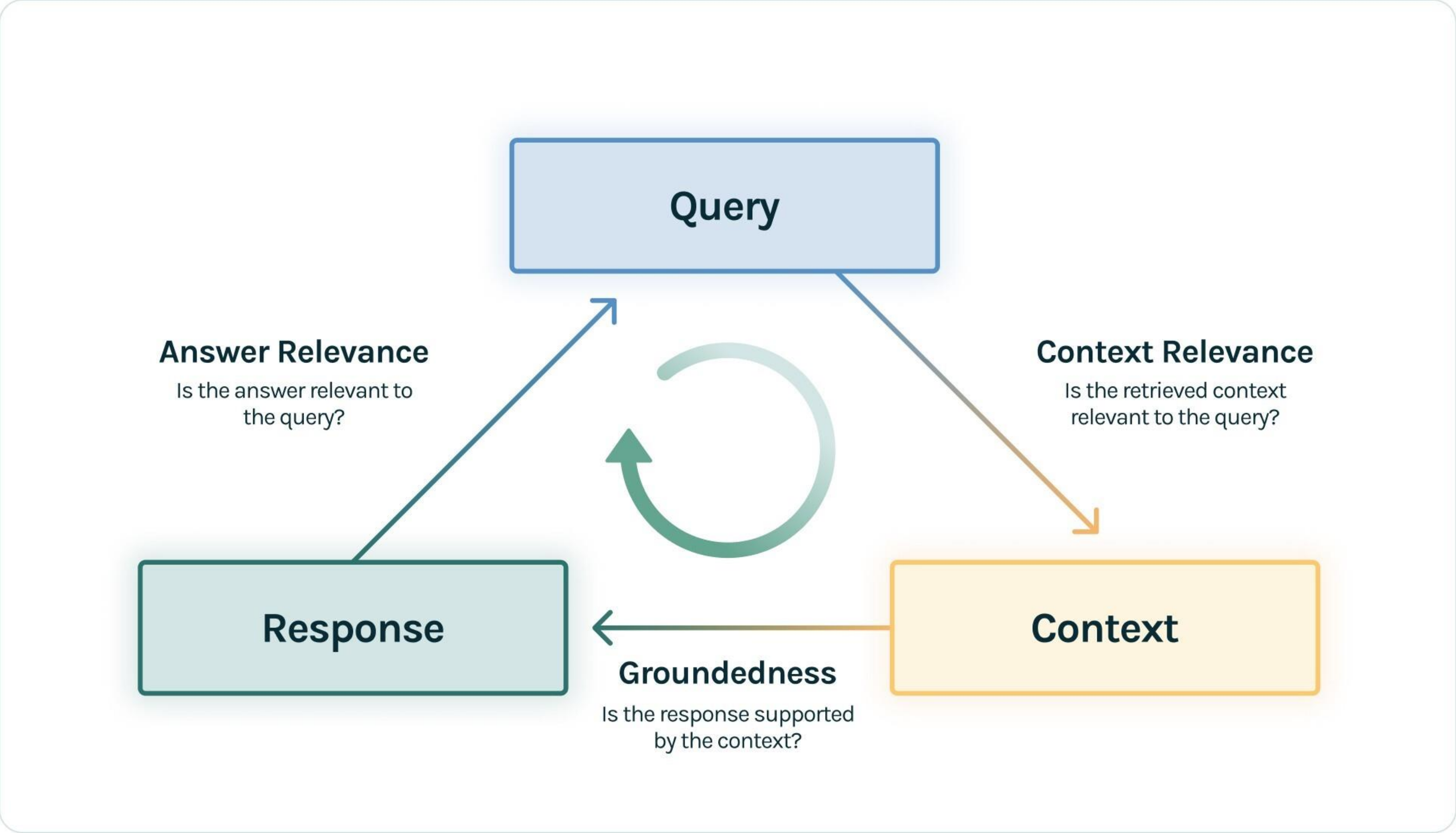
Humeau, S., Shuster, K., Lachaux, M. A., & Weston, J. (2019). Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. arXiv:1905.01969. The poly-encoder architecture introduced here incorporates aspects of both dual and cross-encoders, offering a balance between speed and accuracy for matching tasks. https://arxiv.org/abs/1905.01969

4. Semantic Search Systems

Semantic Search with Transformers:

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., & Cheng, X. (2020). A Deep Look into Neural Ranking Models for Information Retrieval. Information Processing & Management. This review covers deep learning approaches to information retrieval, including the use of Transformer models for understanding query intent and document relevance in a semantic search context. https://www.sciencedirect.com/science/article/pii/S0306457319302390

Evaluating RAG-based LLMs



Query

Context Relevance
Is the retrieved context relevant to the query?

Context

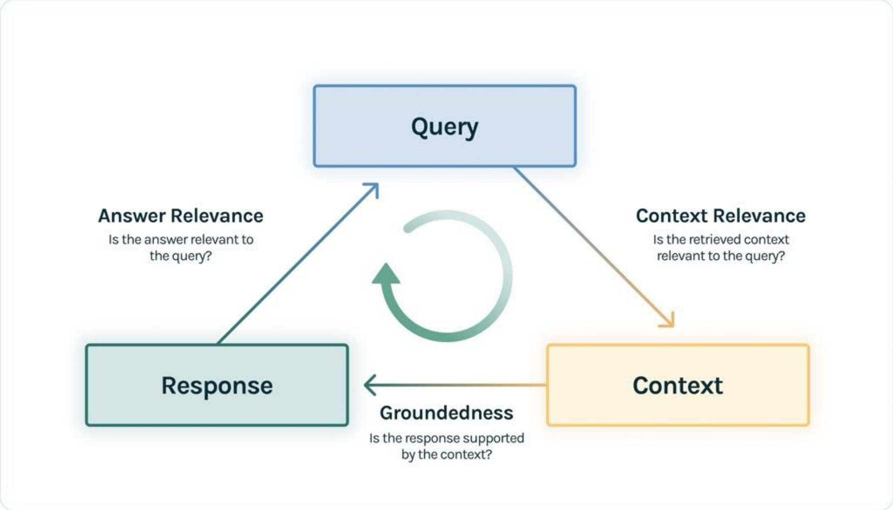
Groundedness
Is the response supported by the context?

Response

Answer Relevance
Is the answer relevant to the query?

https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/

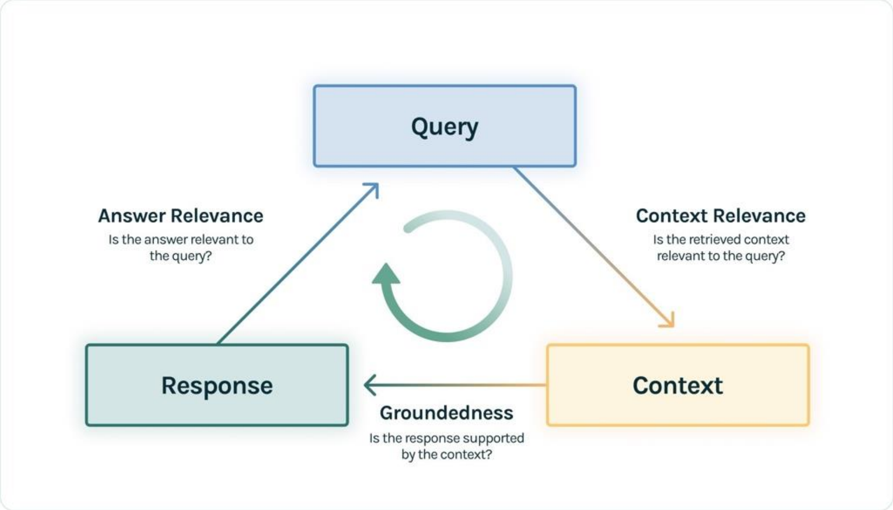
Evaluating RAG: Context Relevance


Query
Answer Relevance
Is the answer relevant to the query?
Context Relevance
Is the retrieved context relevant to the query?
Response
Context
Groundedness
Is the response supported by the context?

- Is the content retrieved from the vector database relevant to the query?
- Irrelevant information will be likely integrated into the response, contributing to hallucinations

https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/

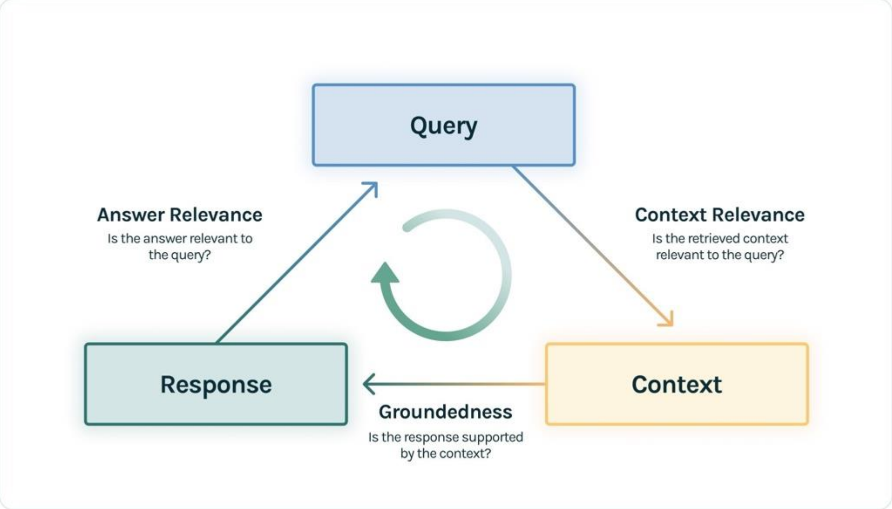
Evaluating RAG: Groundedness


Query
Answer Relevance
Is the answer relevant to the query?
Context Relevance
Is the retrieved context relevant to the query?
Response
Context
Groundedness
Is the response supported by the context?

- The context was provided to the LLM as part of the prompt
- Did the LLM response incorporate the context appropriately?
- Can we support each claim in the response from the context?

https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/

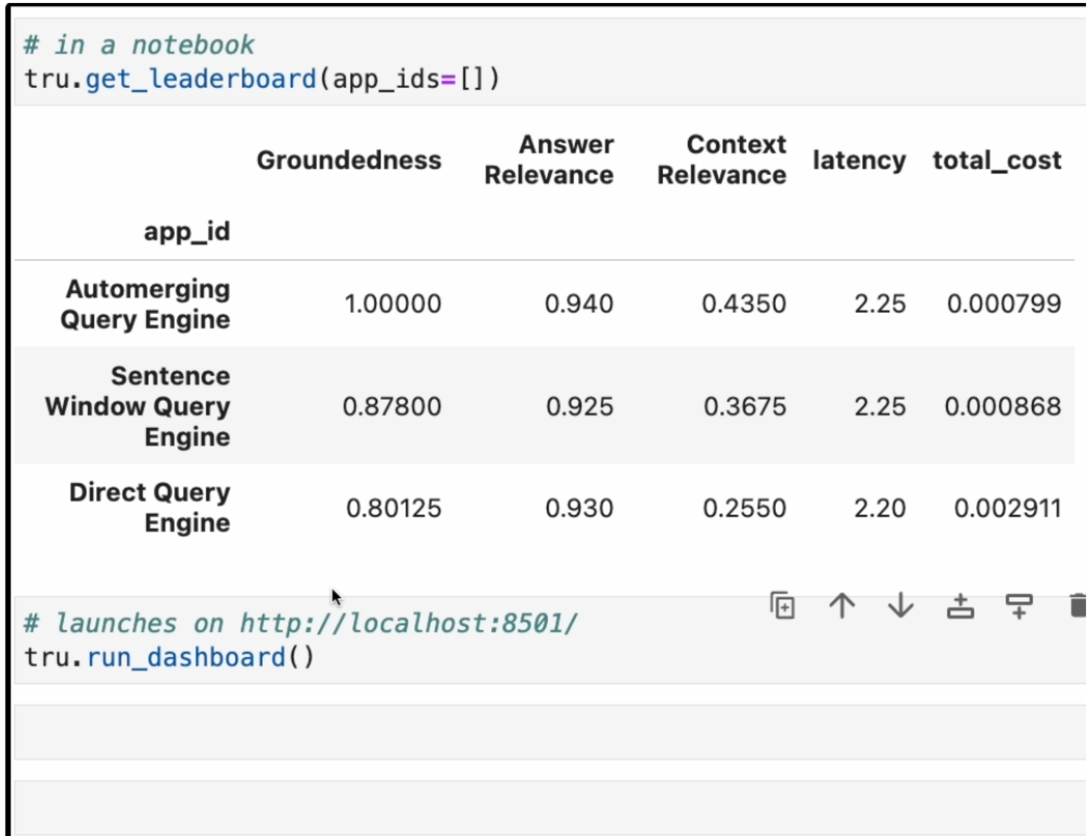
Evaluating RAG: Answer Relevance


Query
Answer Relevance
Is the answer relevant to the query?
Context Relevance
Is the retrieved context relevant to the query?
Response
Context
Groundedness
Is the response supported by the context?

- Is the answer relevant to the original question?
- Prompt is augmented with context.
- Did the context cause the LLM to stray away from the question?

https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/

Growing ecosystem of tools to do evaluation



```
# in a notebook
tru.get_leaderboard(app_ids=[])
```

app_id	Groundedness	Answer Relevance	Context Relevance	latency	total_cost
Automerging Query Engine	1.00000	0.940	0.4350	2.25	0.000799
Sentence Window Query Engine	0.87800	0.925	0.3675	2.25	0.000868
Direct Query Engine	0.80125	0.930	0.2550	2.20	0.002911

```
# launches on http://localhost:8501/
tru.run_dashboard()
```

trulens

Evaluate and Track LLM Applications

Evaluate, iterate faster, and select your best LLM app with TruLens.

Retrieval-Augmented Generation (RAG)

RAG systems have evolved from Naive RAG to Advanced RAG and Modular RAG. This evolution has occurred to address certain limitations around performance, cost, and efficiency.

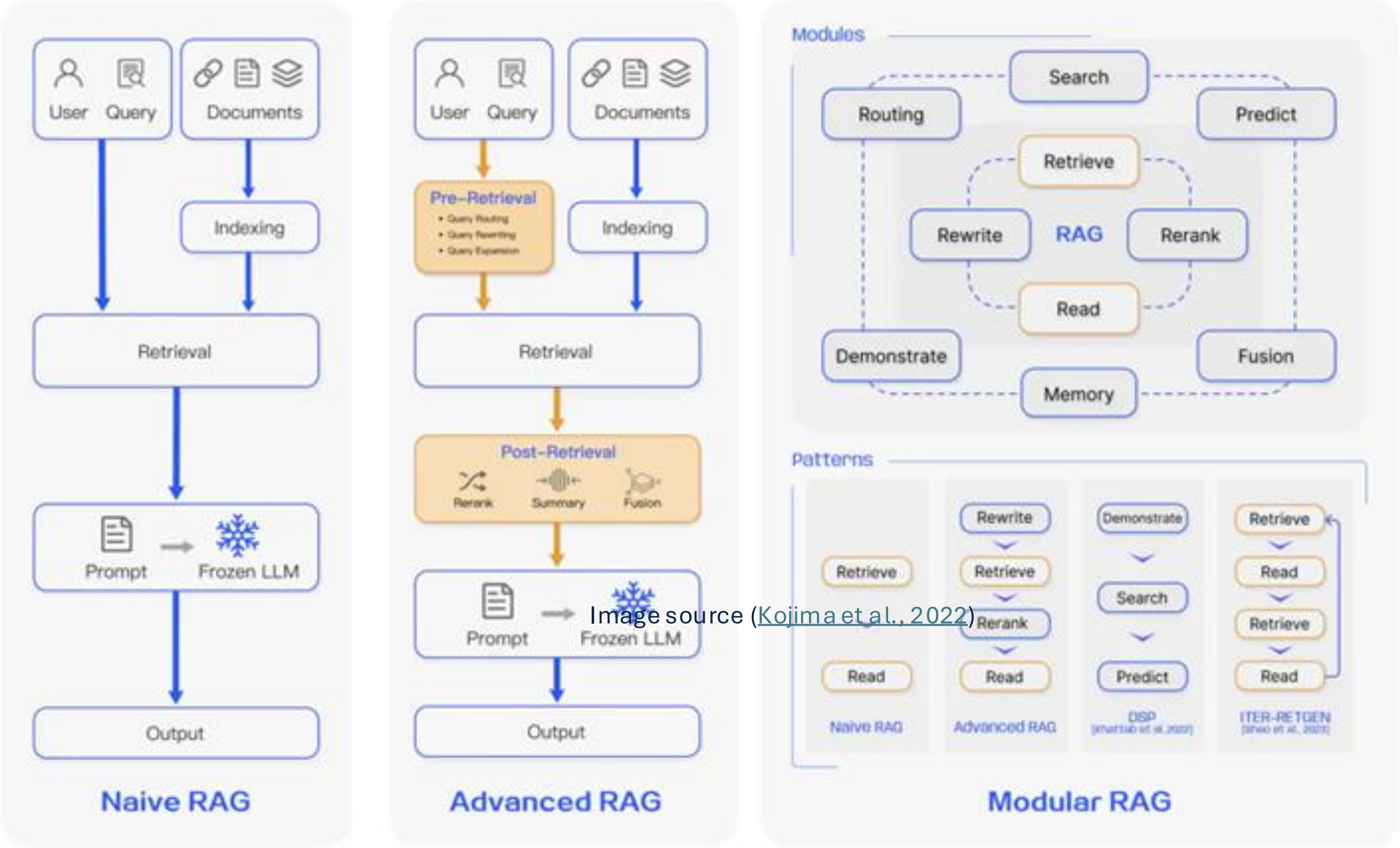


Image source (Kojima et al., 2022)

Pre-Retrieval Improvements
- Enhance indexed data quality, optimize chunk size and overlap.
- Rewrite user queries for better match in vector database.
- Use metadata and pronoun replacement to maintain context in chunks.

Retrieval Enhancements
- Explore alternative search methods (e.g., full-text, graph-based).
- Experiment with different embedding models for task suitability.
- Implement hierarchical and recursive search for precision.

Post-Retrieval Optimization
- Re-rank or score chunks for relevance; compress information from multiple chunks.
- Employ smaller, faster models for specific steps to reduce latency.
- Parallelize intermediate steps and use caching for common queries.

Balancing Quality and Latency
- Opt for parallel processing, smaller models, and caching strategies.
- Tailor RAG approach based on the complexity of user queries and the nature of tasks.

https://www.promptingguide.ai/research/rag

Any questions?

???

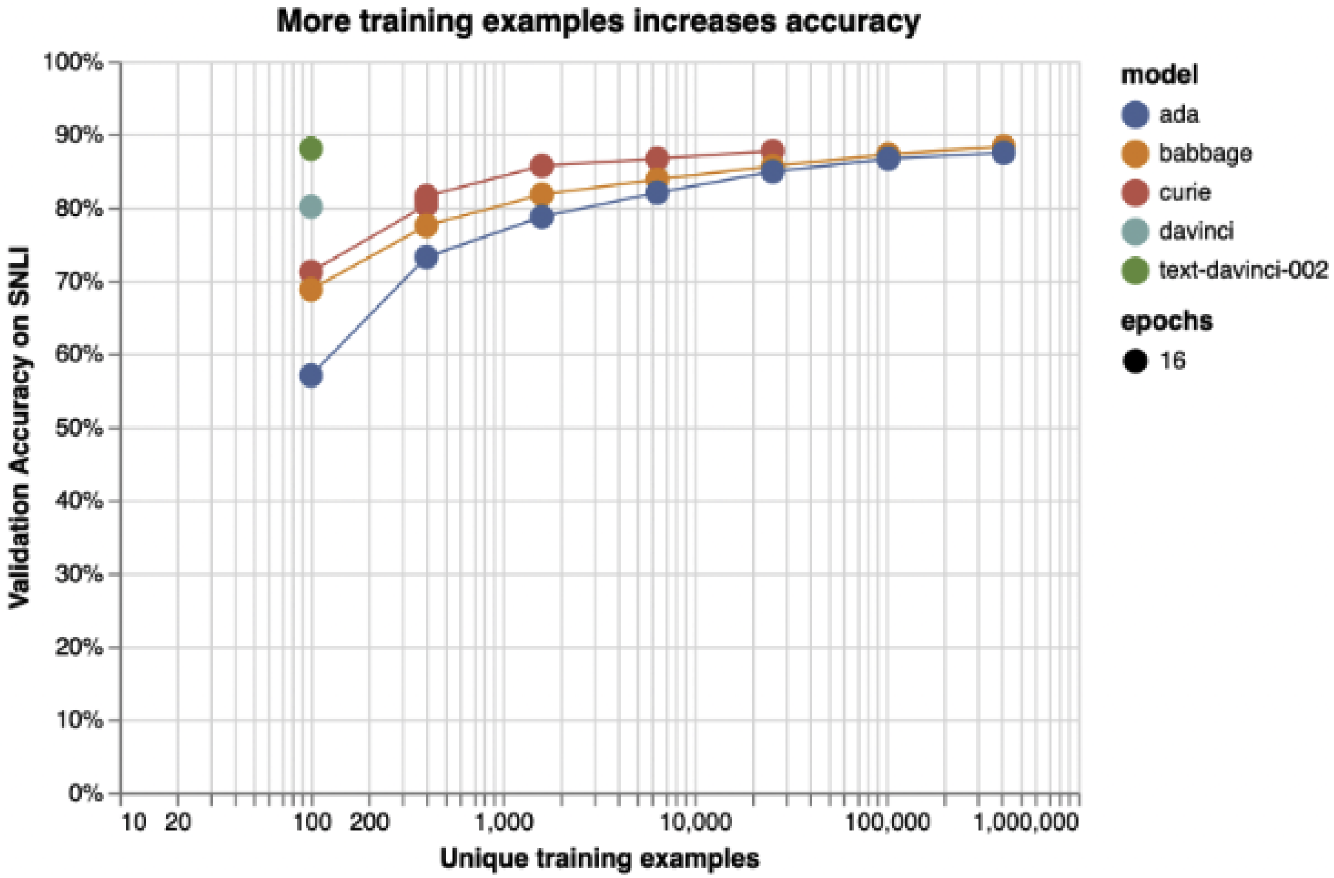
Moving on

- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)

Model Finetuning

- Large foundation models are pre-trained on general tasks
- Might not do as well on specialized tasks
  - Try prompt engineering and retrieval augmentation first
- Good news: can fine tune model with much smaller dataset to adapt to downstream tasks
- Fine tuned model is same size as original.
  - Resource Intensive: Can take very large memory and compute resources to fine tune
  - Storage Demands: If you have n downstream tasks, you will have n copies of your large model.

Full Finetuning Example



Text classification performance on the Stanford Natural Language Inference (SNLI) Corpus. Ordered pairs of sentences are classified by their logical relationship: either contradicted, entailed (implied), or neutral. Default fine-tuning parameters were used when not otherwise specified.

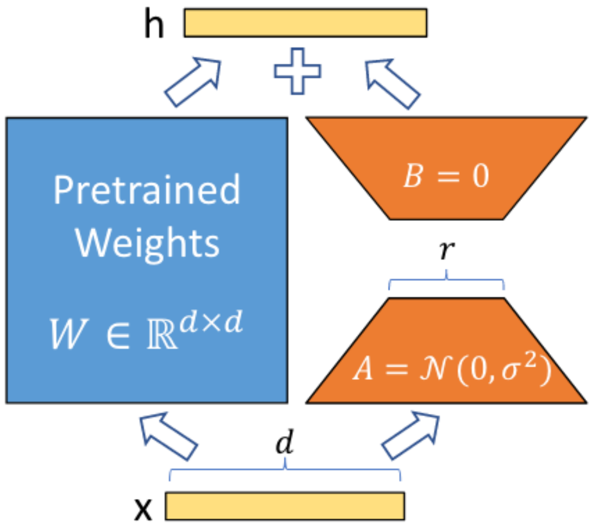
https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning

Model Finetuning Drawbacks

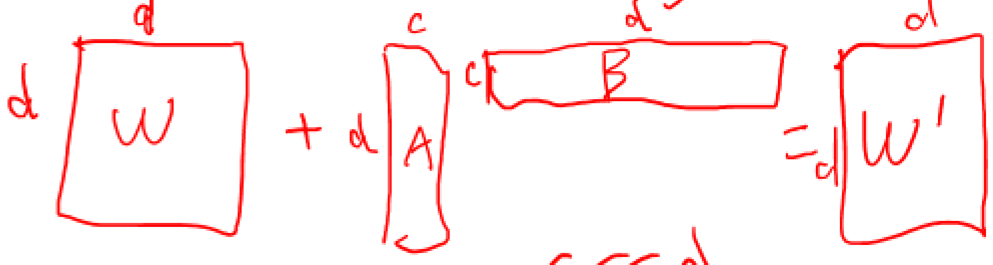
- Fine tuned model is same size as original.
  - Resource Intensive: Can take very large memory and compute resources to fine tune.
  - Storage Demands: If you have n downstream tasks, you will have n copies of your large model.
- Solution is to update aspects of the model, rather than entire model
  - Low Rank Adaptation of the weight updates -- LoRA
  - Train and concatenated soft prompts -- Prompt Tuning

Low Rank Adaptation

- Deploying independent instances of downstream fine-tuned models can be prohibitive (e.g. GPT3, 175B params, 700GB@fp32)
- Instead, freeze the pre-trained model and inject *trainable rank decomposition matrices* into each layer
- Reduce trainable parameters by 10,000x!!
- On-par or better than finetuning on RoBERTa, DeBERTa, GPT-2 and GPT-3


h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

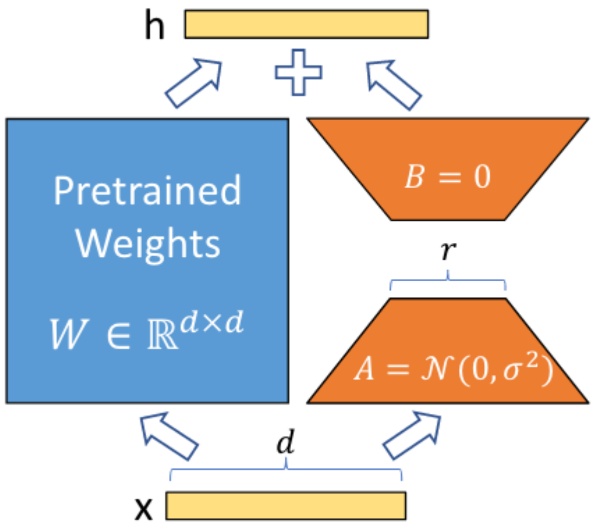
Instead of finetuning W



$c \ll d$

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

Low Rank Adaptation


h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

- Aghajanyan et al show that pretrained language models have a low "intrinsic dimension" often approximately low rank
- Updates to weight matrices likely have a low "intrinsic rank" during training
- Found that even very low rank (e.g. r=1 or2) with GPT-3 175B is effective where full rank (embedding dimension) is 12,288
- Many groups had anecdotally tried but failed to directly train low rank models.
- DeepSeek did this in late 2024.

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685
A. Aghajanyan et al., "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning". arXiv:2012.13255 [cs], December 2020. URL http://arxiv.org/abs/2012.13255.

Reminder: Rank of a Matrix

- The number of linearly independent rows or columns of a matrix
- Determines the dimension of the vector space spanned by the column vectors
- A measure of “dimensionality”

LoRA: Method

Say you have pre-trained weights,

$$W_0 \in \mathbb{R}^{d \times k}$$

Represent update with a low rank decomposition

$$W_0 + \Delta W = W_0 + BA \,,$$

where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ and the rank $r \ll \min(d, k)$, is much less than the full rank.

For updates,

$$h = (W_0 + \Delta W)x = W_0 x + \Delta W x = W_0 x + BAx$$

Initialize A to random gaussian and B to zero

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

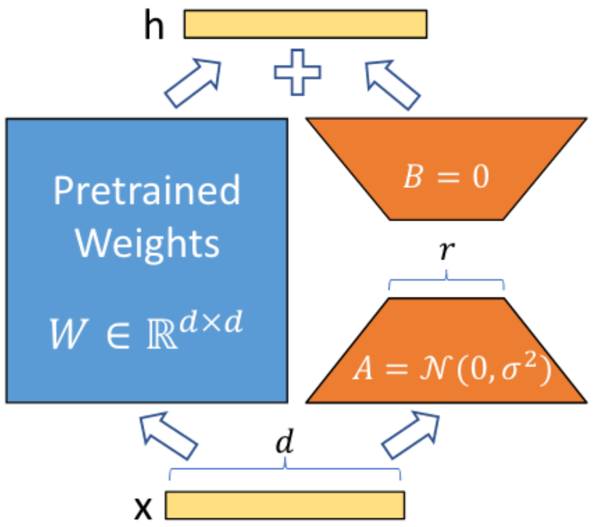
LoRA: Method

LoRA can be viewed as a generalization of full finetuning, since using full rank = full finetuning

Updates:

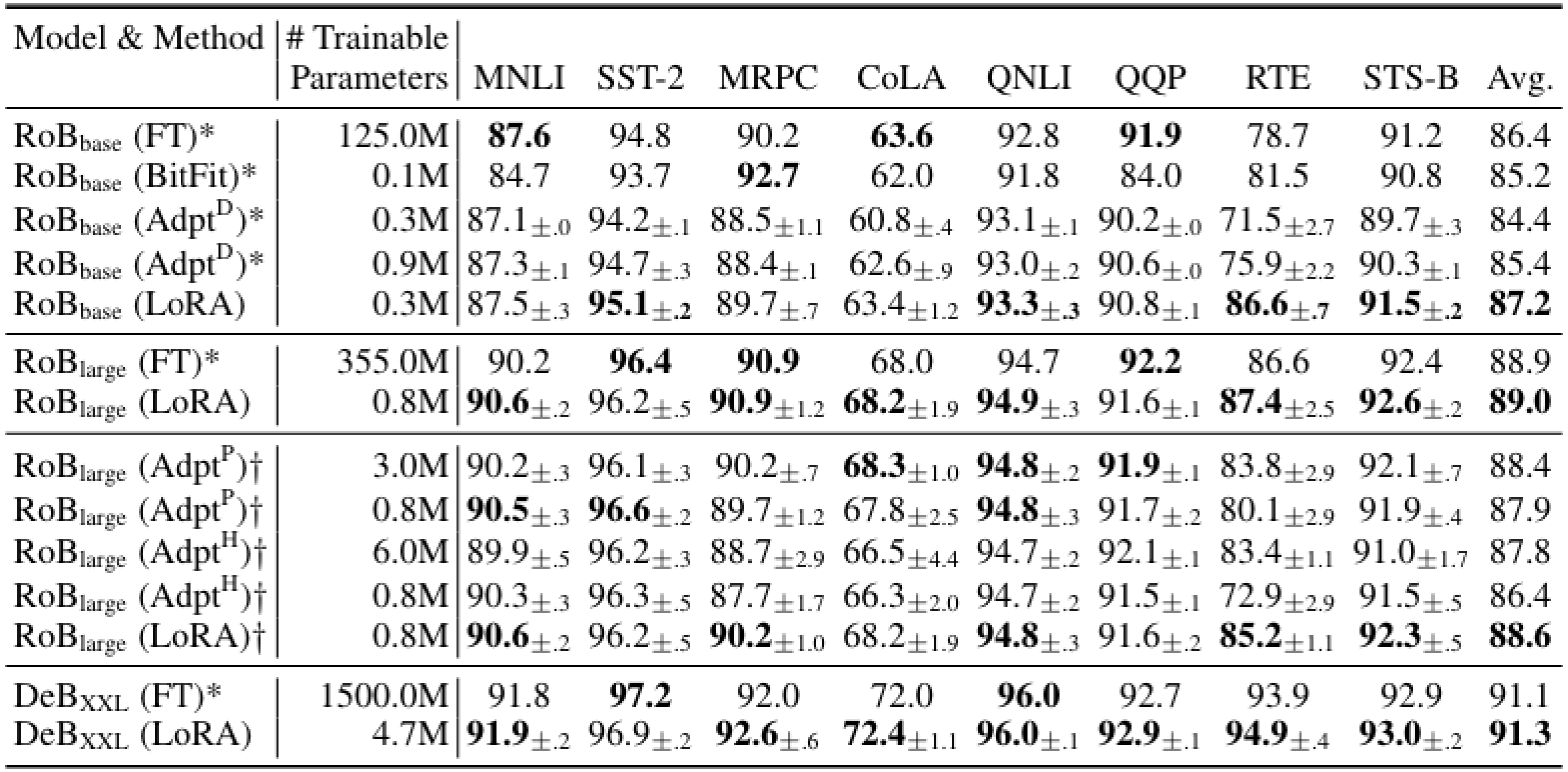
$$h = (W_0 + \Delta W)x = W_0 x + \Delta W x = W_0 x + BAx$$

Generally, only applied to W_q and W_v matrices.


h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

LoRA Results / Comparisons



Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB$_{\text{base}}$ (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB$_{\text{base}}$ (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB$_{\text{base}}$ (Adpt^{D})*	0.3M	87.1$_{\pm.0}$	94.2$_{\pm.1}$	88.5$_{\pm1.1}$	60.8$_{\pm.4}$	93.1$_{\pm.1}$	90.2$_{\pm.0}$	71.5$_{\pm2.7}$	89.7$_{\pm.3}$	84.4
RoB$_{\text{base}}$ (Adpt^{D})*	0.9M	87.3$_{\pm.1}$	94.7$_{\pm.3}$	88.4$_{\pm.1}$	62.6$_{\pm.9}$	93.0$_{\pm.2}$	90.6$_{\pm.0}$	75.9$_{\pm2.2}$	90.3$_{\pm.1}$	85.4
RoB$_{\text{base}}$ (LoRA)	0.3M	87.5$_{\pm.3}$	**95.1**$_{\pm.2}$	89.7$_{\pm.7}$	63.4$_{\pm1.2}$	**93.3**$_{\pm.3}$	90.8$_{\pm.1}$	**86.6**$_{\pm.7}$	**91.5**$_{\pm.2}$	**87.2**
RoB$_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
RoB$_{\text{large}}$ (LoRA)	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.9**$_{\pm1.2}$	**68.2**$_{\pm1.9}$	**94.9**$_{\pm.3}$	91.6$_{\pm.1}$	**87.4**$_{\pm2.5}$	**92.6**$_{\pm.2}$	**89.0**
RoB$_{\text{large}}$ (Adpt^{P})†	3.0M	90.2$_{\pm.3}$	96.1$_{\pm.3}$	90.2$_{\pm.7}$	**68.3**$_{\pm1.0}$	**94.8**$_{\pm.2}$	**91.9**$_{\pm.1}$	83.8$_{\pm2.9}$	92.1$_{\pm.7}$	88.4
RoB$_{\text{large}}$ (Adpt^{P})†	0.8M	**90.5**$_{\pm.3}$	**96.6**$_{\pm.2}$	89.7$_{\pm1.2}$	67.8$_{\pm2.5}$	**94.8**$_{\pm.3}$	91.7$_{\pm.2}$	80.1$_{\pm2.9}$	91.9$_{\pm.4}$	87.9
RoB$_{\text{large}}$ (Adpt^{H})†	6.0M	89.9$_{\pm.5}$	96.2$_{\pm.3}$	88.7$_{\pm2.9}$	66.5$_{\pm4.4}$	94.7$_{\pm.2}$	92.1$_{\pm.1}$	83.4$_{\pm1.1}$	91.0$_{\pm1.7}$	87.8
RoB$_{\text{large}}$ (Adpt^{H})†	0.8M	90.3$_{\pm.3}$	96.3$_{\pm.5}$	87.7$_{\pm1.7}$	66.3$_{\pm2.0}$	94.7$_{\pm.2}$	91.5$_{\pm.1}$	72.9$_{\pm2.9}$	91.5$_{\pm.5}$	86.4
RoB$_{\text{large}}$ (LoRA)†	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.2**$_{\pm1.0}$	68.2$_{\pm1.9}$	**94.8**$_{\pm.3}$	91.6$_{\pm.2}$	**85.2**$_{\pm1.1}$	**92.3**$_{\pm.5}$	**88.6**
DeB$_{\text{XXL}}$ (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB$_{\text{XXL}}$ (LoRA)	4.7M	**91.9**$_{\pm.2}$	96.9$_{\pm.2}$	**92.6**$_{\pm.6}$	**72.4**$_{\pm1.1}$	**96.0**$_{\pm.1}$	**92.9**$_{\pm.1}$	**94.9**$_{\pm.4}$	**93.0**$_{\pm.2}$	**91.3**

GLUE benchmark – measure across 9 language tasks

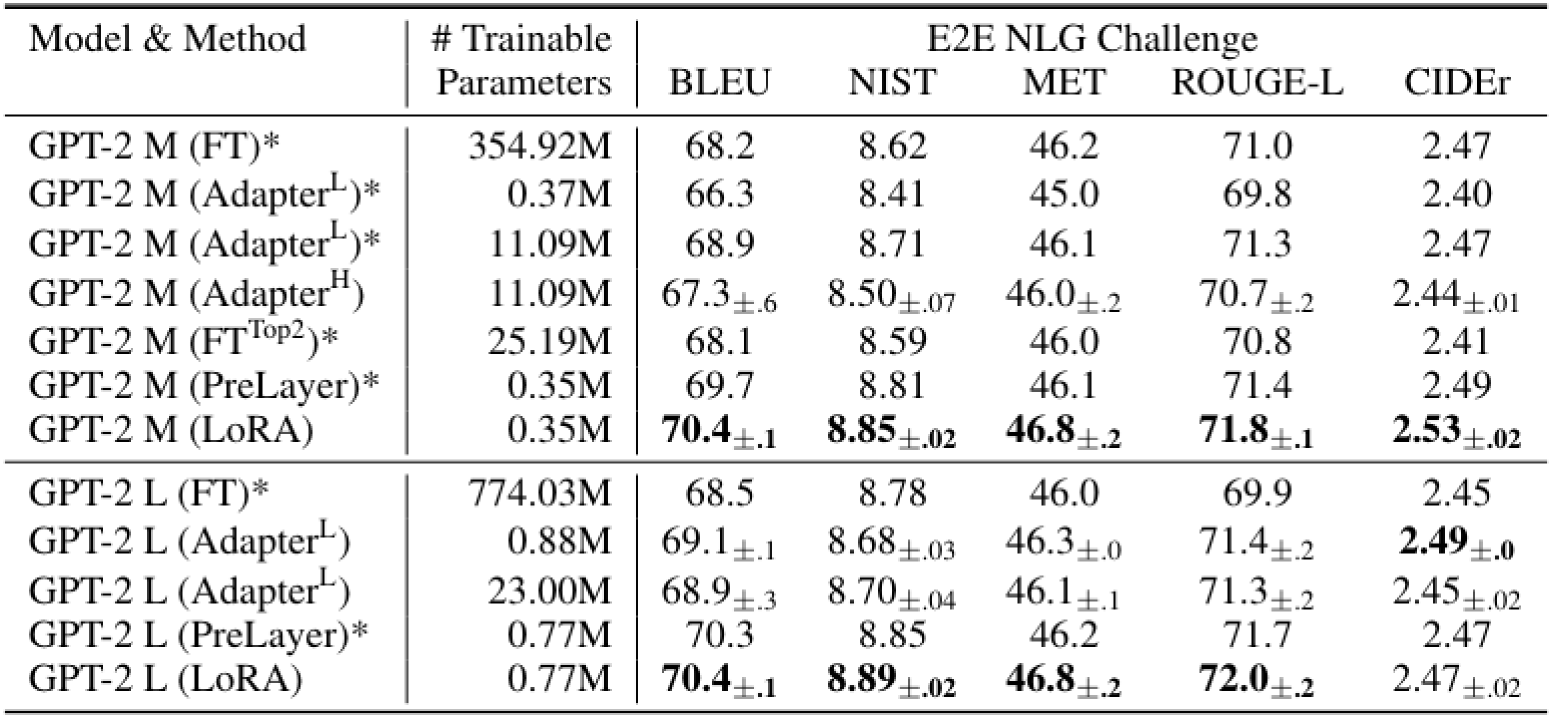
BitFit – train only the bias vectors

Adpt – Inserts adaptation layer between self-attention and MLP module

E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

† indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

LoRA Results / Comparisons



Model & Method	# Trainable Parameters	E2E NLG Challenge				
		BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	**70.4**$_{\pm.1}$	**8.85**$_{\pm.02}$	**46.8**$_{\pm.2}$	**71.8**$_{\pm.1}$	**2.53**$_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	**2.49**$_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	**70.4**$_{\pm.1}$	**8.89**$_{\pm.02}$	**46.8**$_{\pm.2}$	**72.0**$_{\pm.2}$	$2.47_{\pm.02}$

GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA outperforms several baselines with comparable or fewer trainable parameters. Confidence intervals are shown for experiments we ran. * indicates numbers published in prior works.

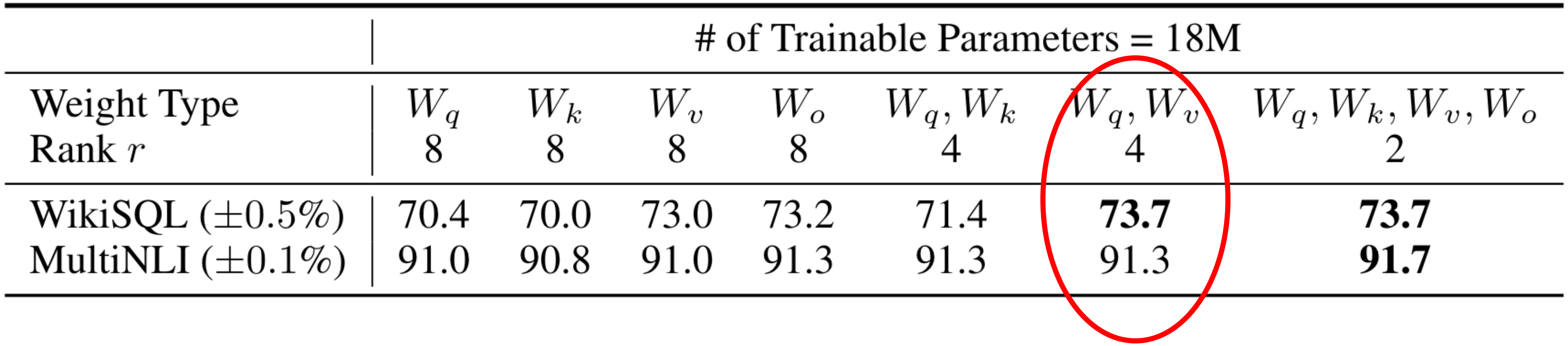
E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

Understanding the Low-Rank Updates

1. Given a parameter budget constraint, which subset of weight matrices in a pre-trained Transformer should we adapt to maximize downstream performance?
2. Is the "optimal" adaptation matrix ΔW really rank-deficient? If so, what is a good rank to use in practice?

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

1) Which weight matrices to target?



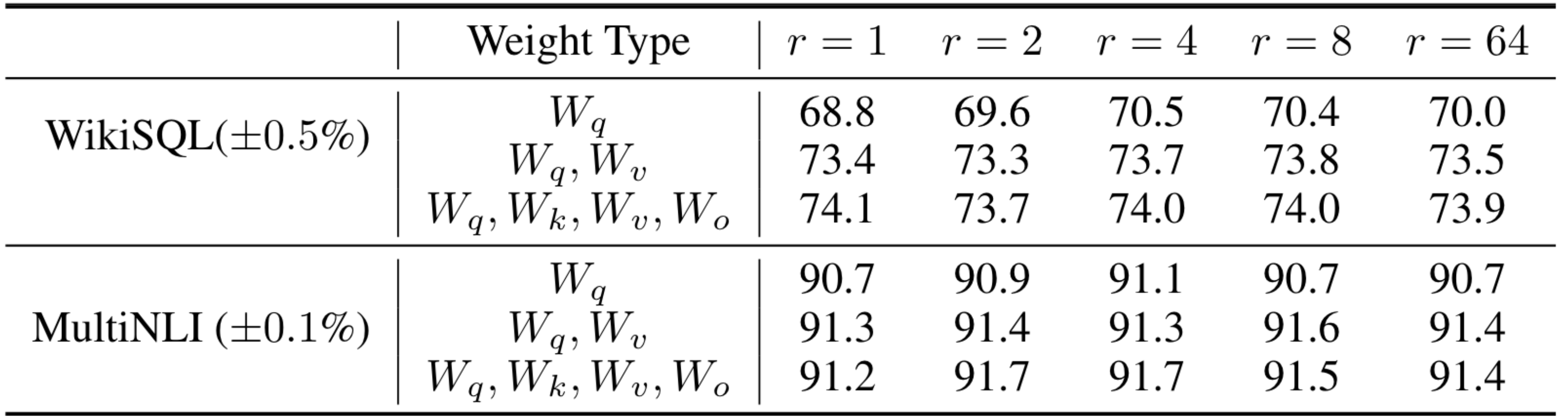
	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL (±0.5%)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI (±0.1%)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

Validation accuracy on WikiSQL and MultiNLI after applying LoRA to different types of attention weights in GPT-3, given the same number of trainable parameters. Adapting both Wq and Wv gives the best performance overall. We find the standard deviation across random seeds to be consistent for a given dataset, which we report in the first column.

Rank of 16 on 2 matrices or even 4 on 4 matrices is sufficient.

E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

2) What is the optimal rank?



	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

“Validation accuracy on WikiSQL and MultiNLI with different rank r. To our surprise, a rank as small as one suffices for adapting both Wq and Wv on these datasets while training Wq alone needs a larger r.”

E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

Any questions?

???

- Sub-quadratic attention follow up
- LLM training
- LLM evaluation
- Retrieval-augmented generation
- Parameter efficient fine-tuning (low-rank adaptation)