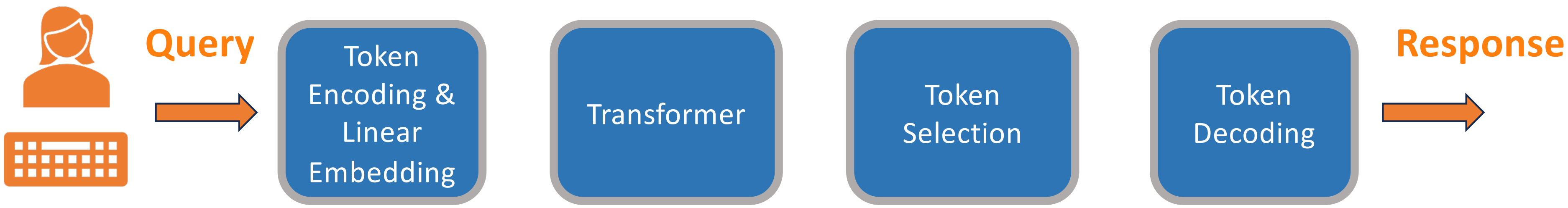
Improving Generative LLMs

DL4DS – Spring 2026

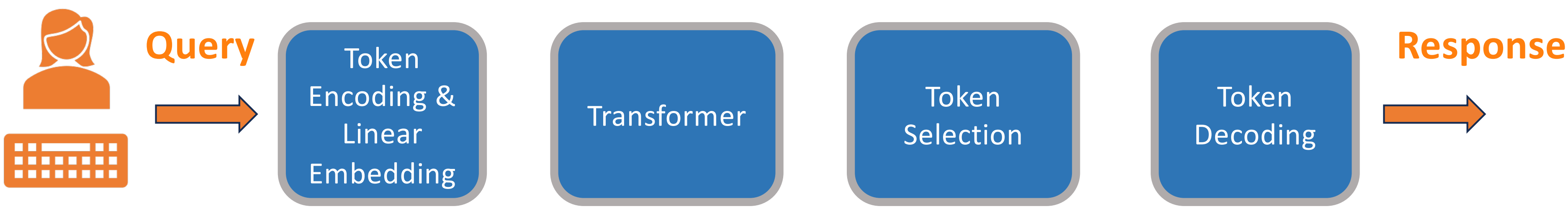
Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- Parameter Efficient Fine-Tuning: Low-Rank Adaptation

LLM Generative Flow

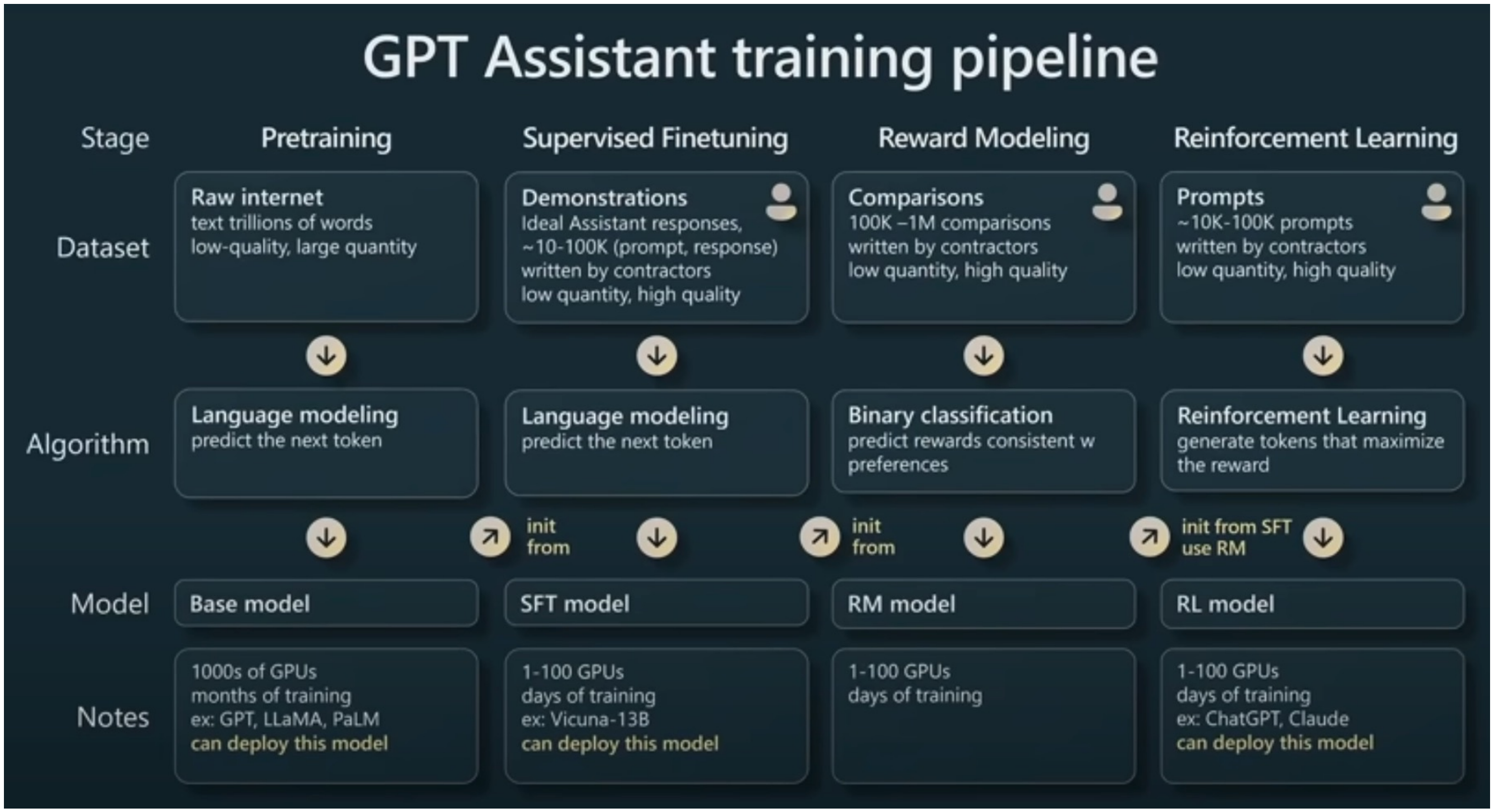


LLM Generative Flow

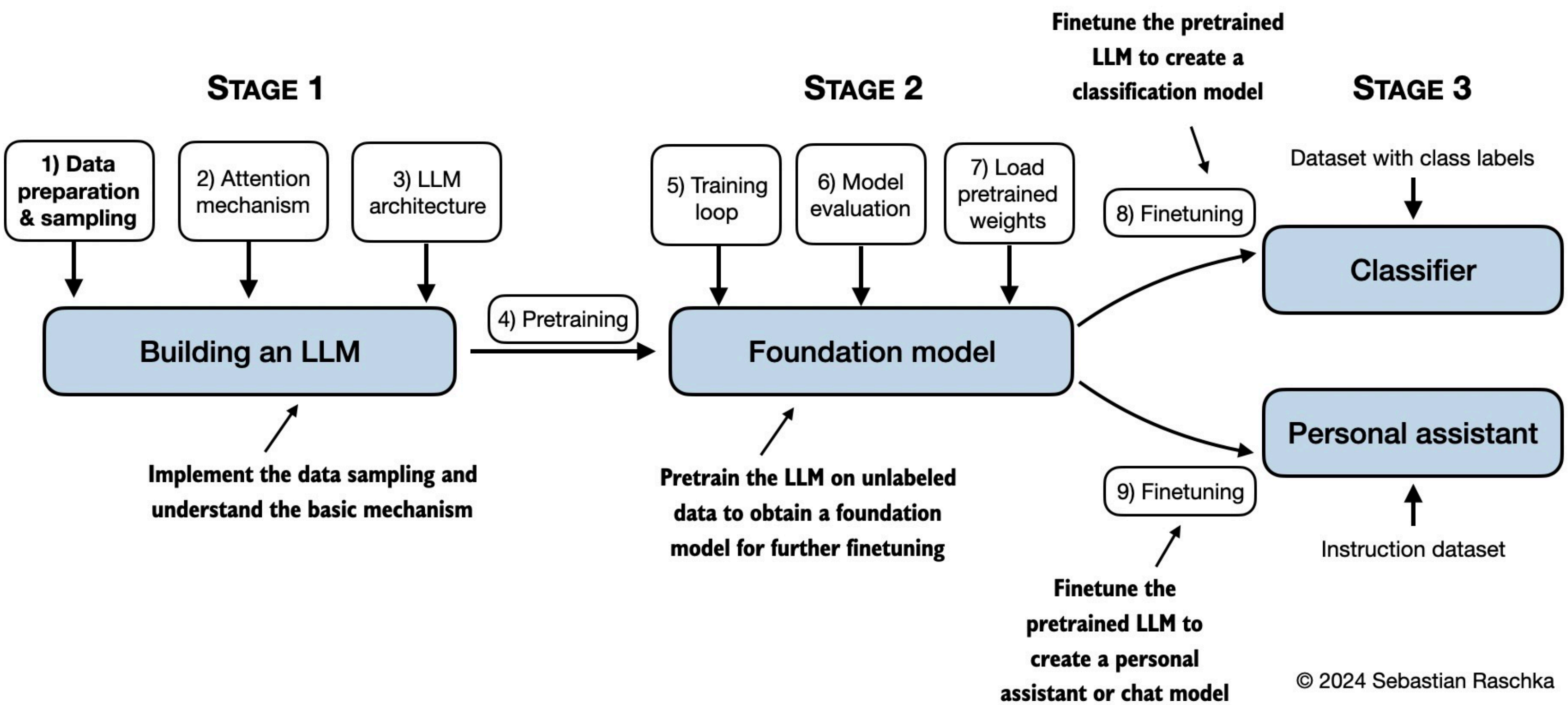


- How do we improve the response?
- How do we evaluate the response?

How do we build a chat model?



State of GPT, Andrej Karpathy, MS Build Keynote, 2023

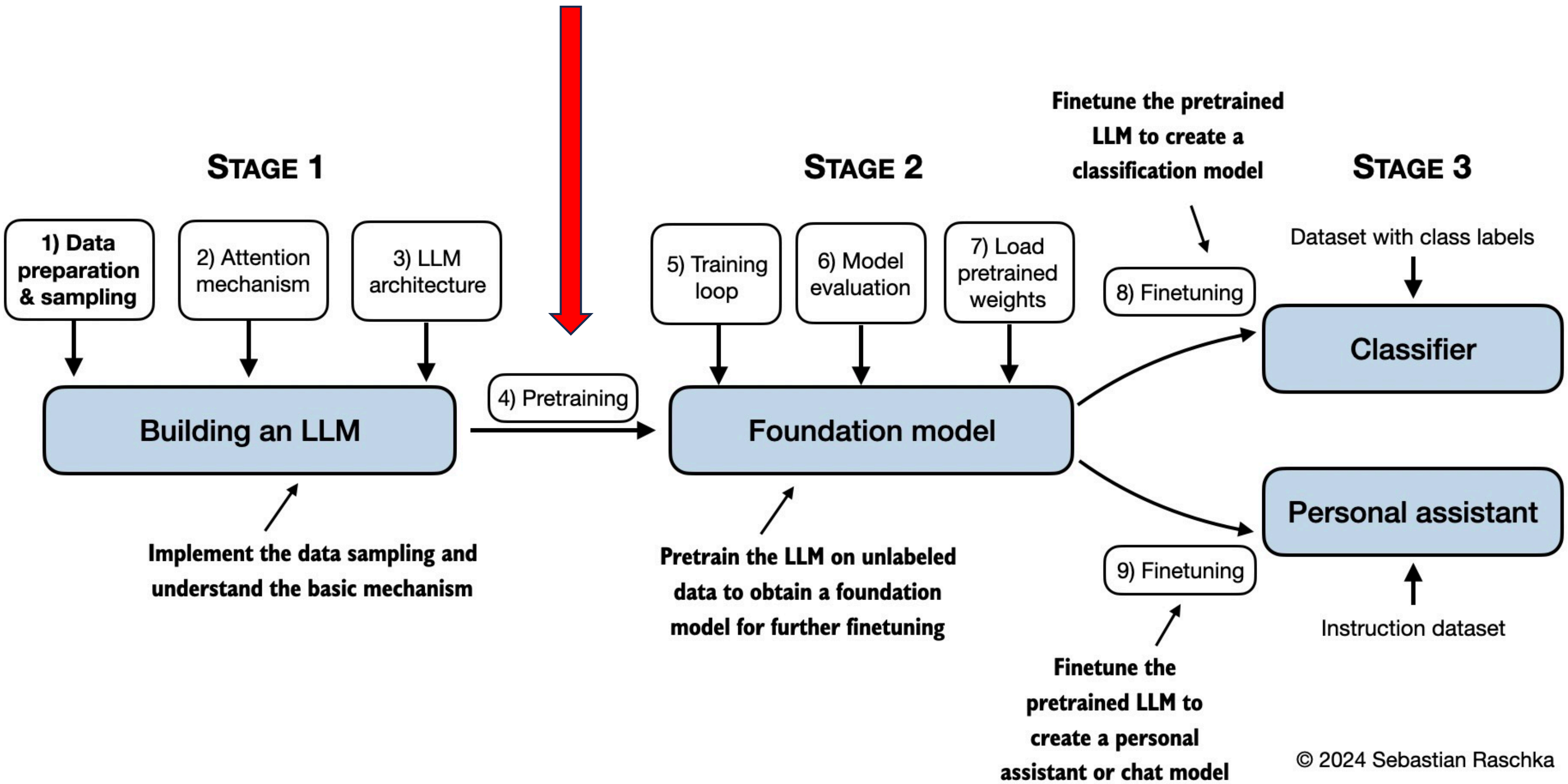


Finetune the pretrained LLM to create a classification model
STAGE 1
STAGE 2
STAGE 3
1) Data preparation & sampling
2) Attention mechanism
3) LLM architecture
5) Training loop
6) Model evaluation
7) Load pretrained weights
8) Finetuning
Dataset with class labels
Classifier
Building an LLM
4) Pretraining
Foundation model
Personal assistant
Implement the data sampling and understand the basic mechanism
Pretrain the LLM on unlabeled data to obtain a foundation model for further finetuning
9) Finetuning
Instruction dataset
Finetune the pretrained LLM to create a personal assistant or chat model
© 2024 Sebastian Raschka

Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- Parameter Efficient Fine-Tuning: Low-Rank Adaptation

Pre-Training



The GPT-3 dataset was 499 billion 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

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

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

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."

Why did they stop listing training sources?

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

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/

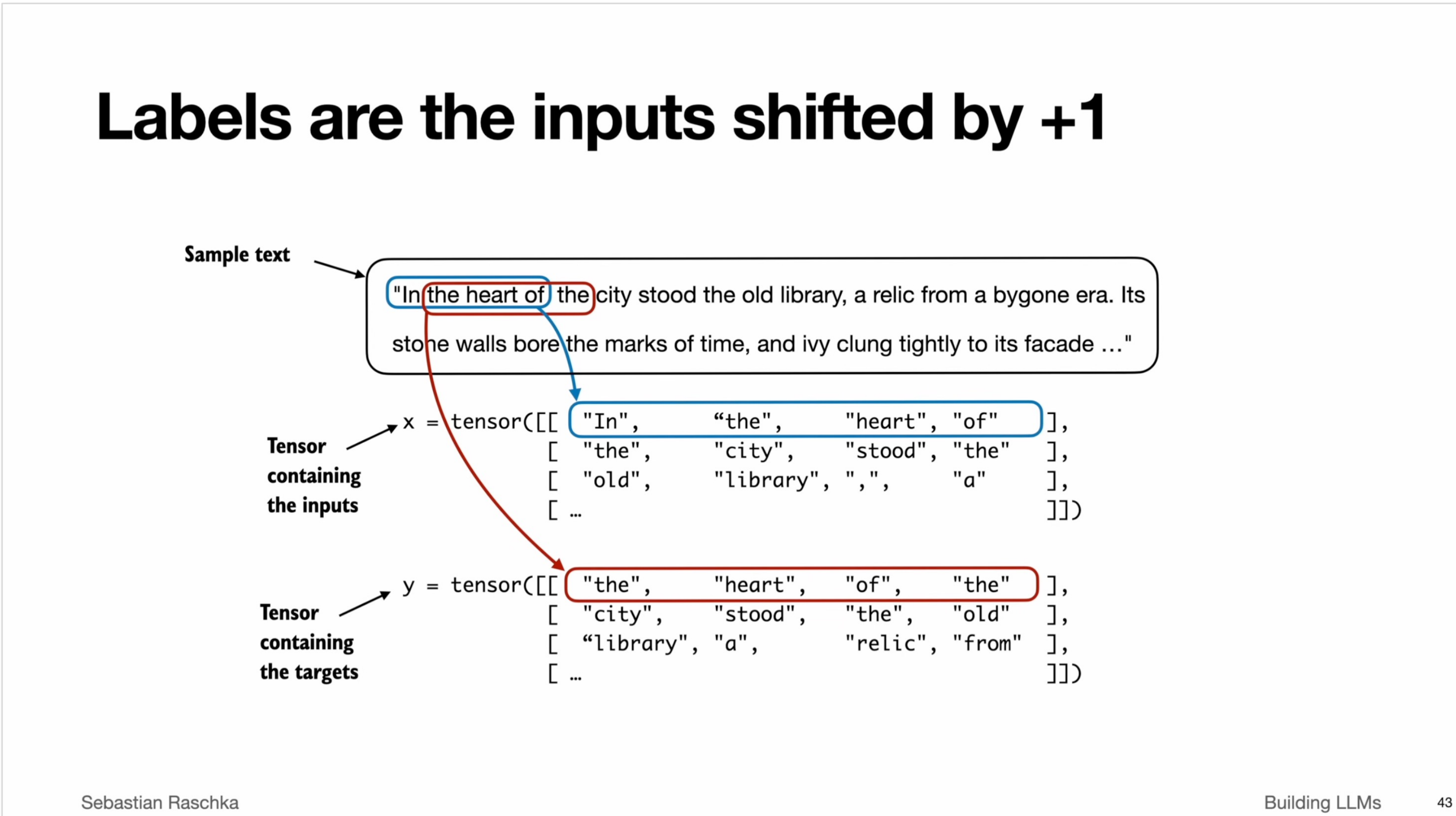
S. Raschka, "Developing an LLM: Building, Training, Finetuning"

Quantity vs quality

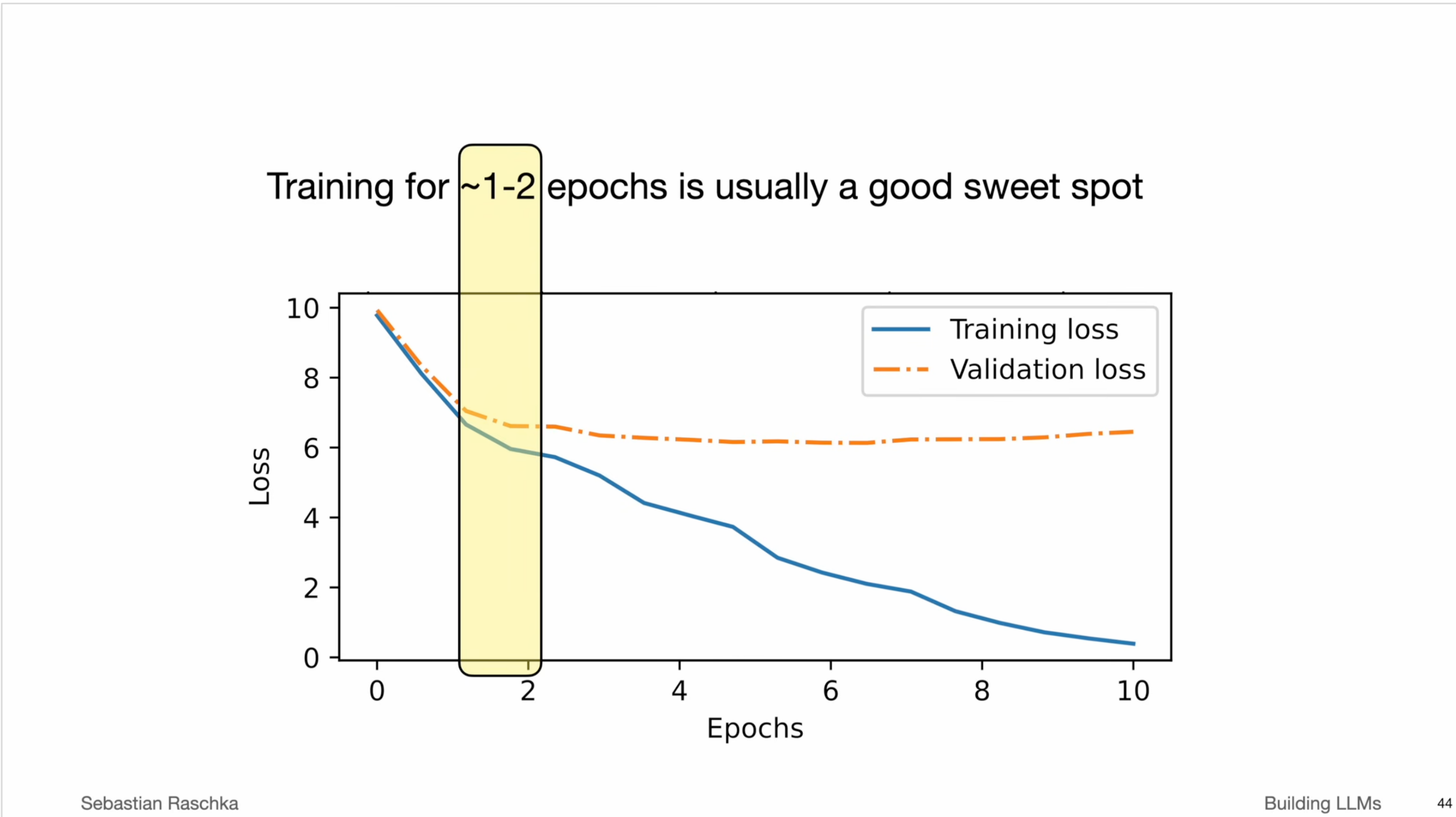
"we mainly focus on the quality of data for a given scale. We try to calibrate the training data to be closer to the "data optimal" regime for small models. In particular, we filter the publicly available web data to contain the correct level of "knowledge" and keep more web pages that could potentially 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

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



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



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

Scaling Laws

N: Model size (# parameters)
D: Data set size (# tokens)
C: Compute budget (TOPS)

- Original Scaling Laws (Kaplan et al., January 2020)
  - Kaplan et al. [2020] established power-law relationships between model performance and three factors:
  - model size (N), dataset size (D), and compute budget (C).

The Scaling Law Tradeoff*: $\boldsymbol{L = A/N^{\alpha} + B/D^{\beta} + L_0}$

where L is the cross-entropy loss

Three terms, three error sources:

- A/N^{α} — approximation error (shrinks with more parameters)
- B/D^{β} — estimation error (shrinks with more training data)
- L_0 — irreducible error (entropy of language itself)

The compute constraint: $\boldsymbol{C \approx 6ND}$

- ~6 FLOPs per parameter per token (2 forward + 2 backward + 2 optimizer)
- Fixed budget C forces a tradeoff: bigger N means smaller $D = C/(6N)$

*Kaplan et al. [2020]

Scaling Laws

N: Model size (# parameters)
D: Data set size (# tokens)
C: Compute budget (TOPS)

- Original Scaling Laws (Kaplan et al., January 2020)
  - Kaplan et al. [2020] established power-law relationships between model performance and three factors:
  - model size (N), dataset size (D), and compute budget (C).
  - **Key conclusion**: $N_{optimal} \propto C^{0.73}$, suggesting compute should be allocated primarily to larger models. Under this framework, GPT-3 (175B parameters) was trained on ~300B tokens (~1.7 tokens/parameter).
- The Chinchilla Paper (Hoffmann et al., March 2022)
  - Hoffmann et al. [2022] trained 400+ models (70M–16B parameters) and found the corrected scaling:
  - model size and training tokens should scale equally ($N_{optimal} \propto C^{0.50}$), yielding ~20 tokens per parameter. Chinchilla (70B parameters, 1.4T tokens) outperformed Gopher (280B parameters, 300B tokens) using the same compute budget.
  - **The core insight**: *most existing LLMs were massively undertrained*—too many parameters, too little data. This paper reshaped the entire field's training strategy.
- Beyond Chinchilla: Inference-Aware Scaling (2023–2024)
  - Sardana and Frankle [2024] modified Chinchilla's framework to account for inference costs. When serving demand is high, it is optimal to train smaller models for much longer than Chinchilla-optimal. This explains the industry trajectory: LLaMA-1 (20:1 token-to-parameter ratio) → LLaMA-2 (30:1) → LLaMA-3 8B (1,875:1, trained on 15T tokens).
  - **The "Chinchilla Trap"**: compute-optimal models are often too large for cost-effective deployment.
- Inference-Time Scaling Laws (2024)
  - A separate paradigm emerged: rather than scaling training compute, allocate more compute at inference time. Snell et al. [2025] showed that a smaller model with optimal test-time compute can outperform a 14× larger model.
  - **This established two independent axes for scaling AI capability**: training-time and inference-time.

The Scaling Law Tradeoff*: $\boldsymbol{L = A/N^{\alpha} + B/D^{\beta} + L_0}$

where L is the cross-entropy loss

Three terms, three error sources:

- A/N^{α} — approximation error (shrinks with more parameters)
- B/D^{β} — estimation error (shrinks with more training data)
- L_0 — irreducible error (entropy of language itself)

The compute constraint: $\boldsymbol{C \approx 6ND}$

- ~6 FLOPs per parameter per token (2 forward + 2 backward + 2 optimizer)
- Fixed budget C forces a tradeoff: bigger N means smaller $D = C/(6N)$

The ratio α/β determines the optimal strategy:

- Kaplan (α > β): $N_{opt} \propto C^{0.73}$ → "spend compute on parameters" → GPT-3: 1.7 tok/param
- Chinchilla ($\alpha \approx \beta$): $N\_opt \propto C$^0.50 → "scale both equally" → Chinchilla: 20 tok/param
- Inference-aware (add serving cost): → "overtrain small models" → LLaMA 3 8B: 1,875 tok/param

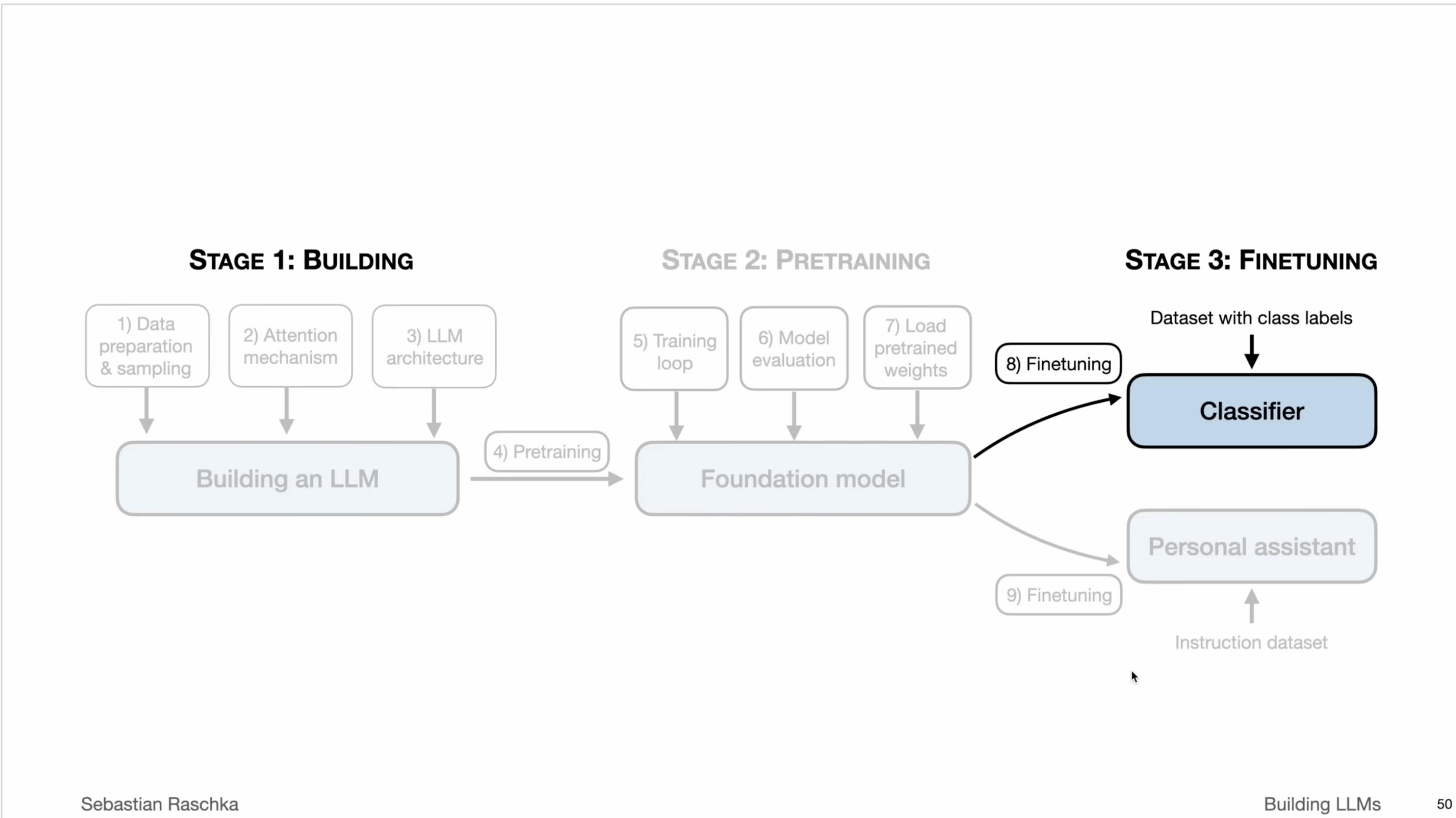
Same equation, same constraint, three different answers — depending on what you measure and what you optimize for---

*Kaplan et al. [2020]

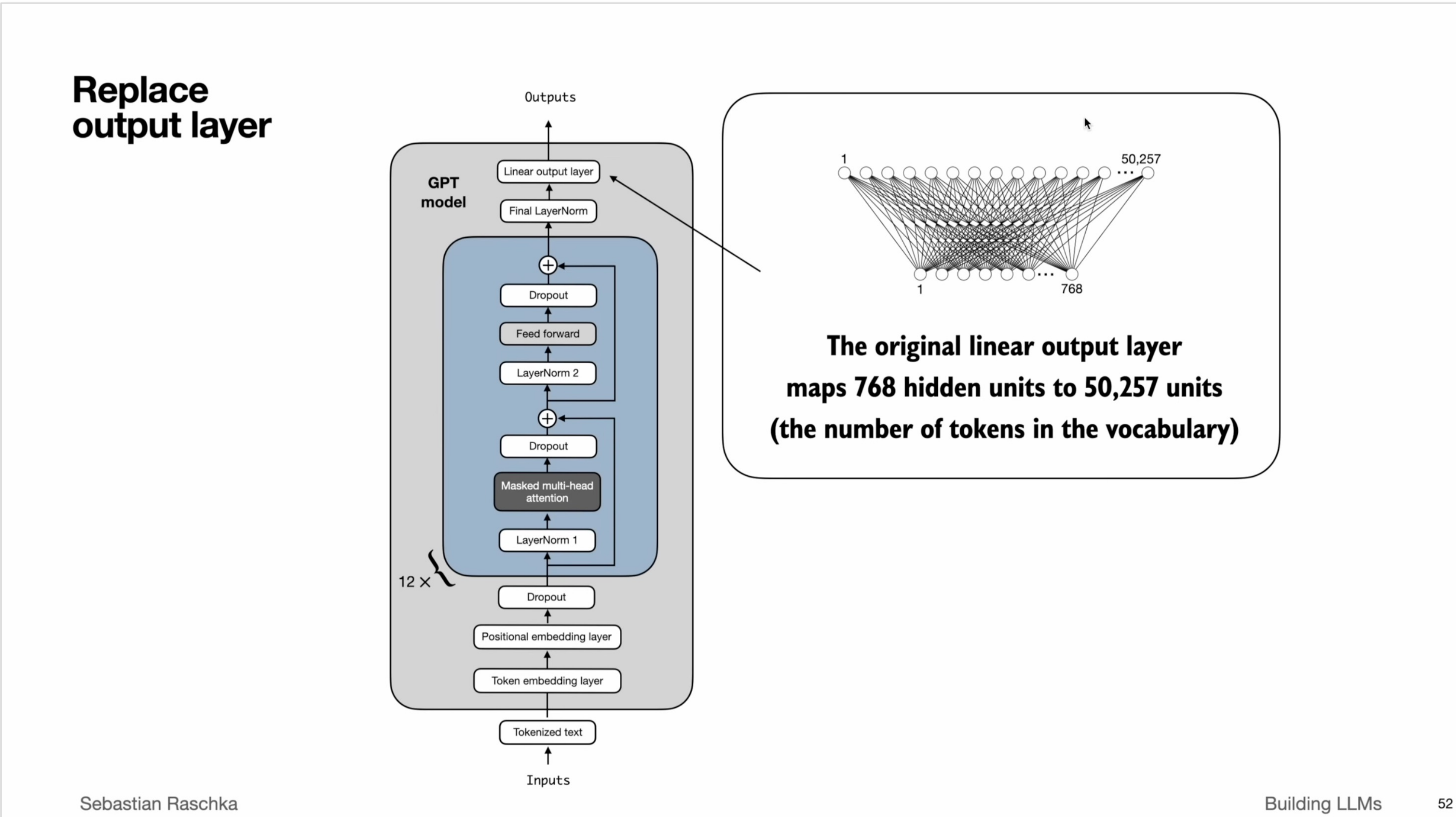
Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- Parameter Efficient Fine-Tuning: Low-Rank Adaptation

Classifier Finetuning

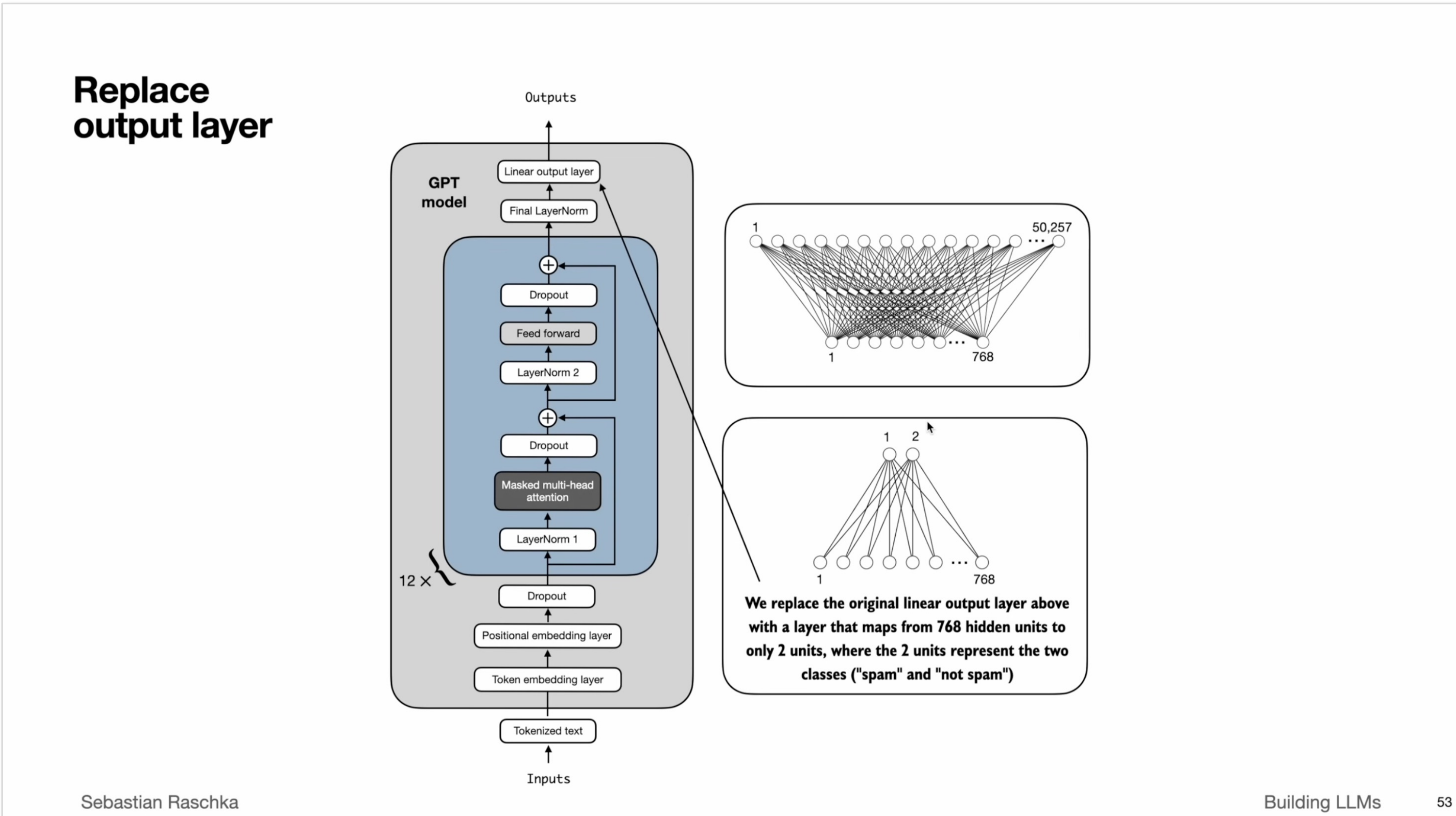


S. Raschka, "Developing an LLM: Building, Training, Finetuning"



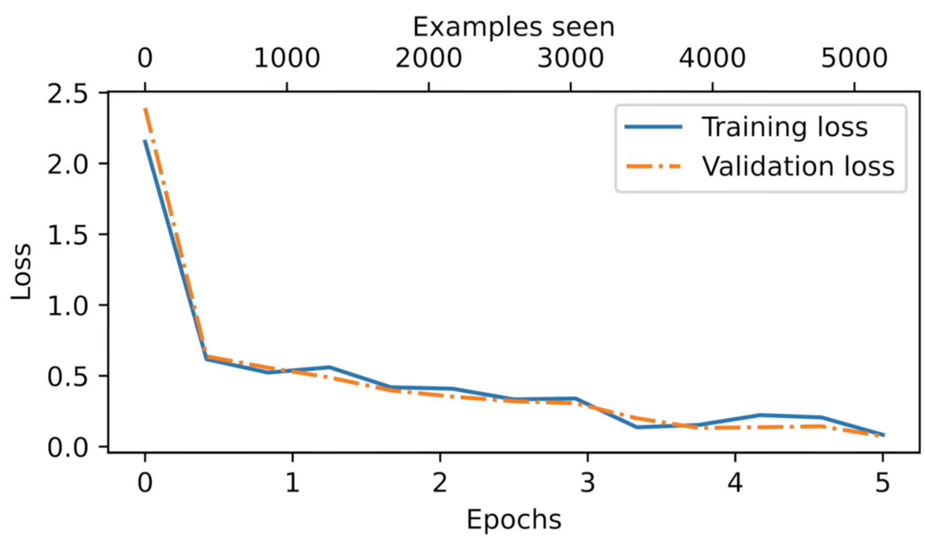
[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]

Example: Spam/Ham Classifier



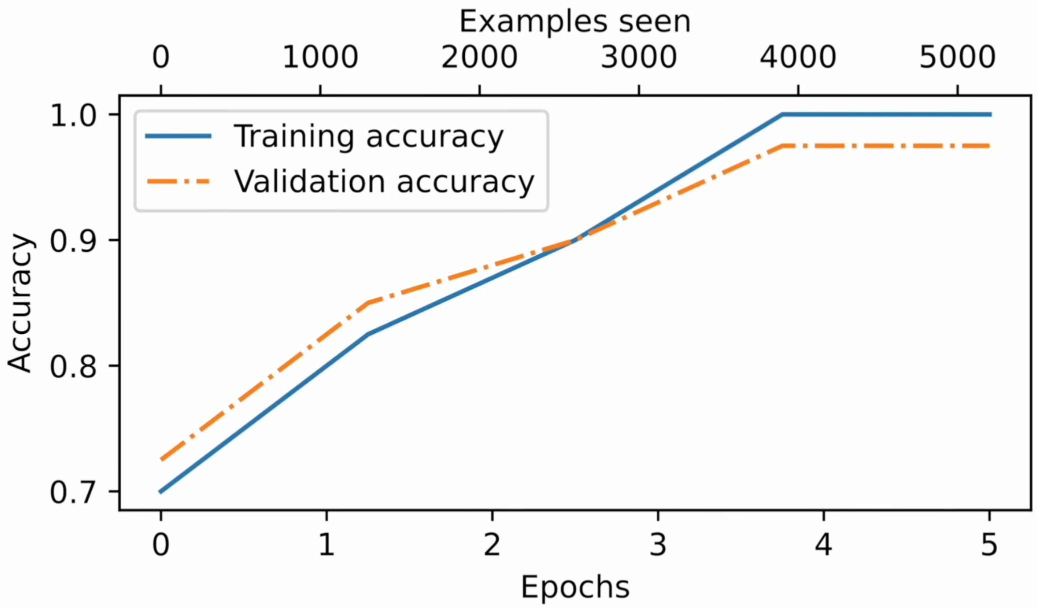
[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]

Track loss values as usual



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

In addition, look at task performance



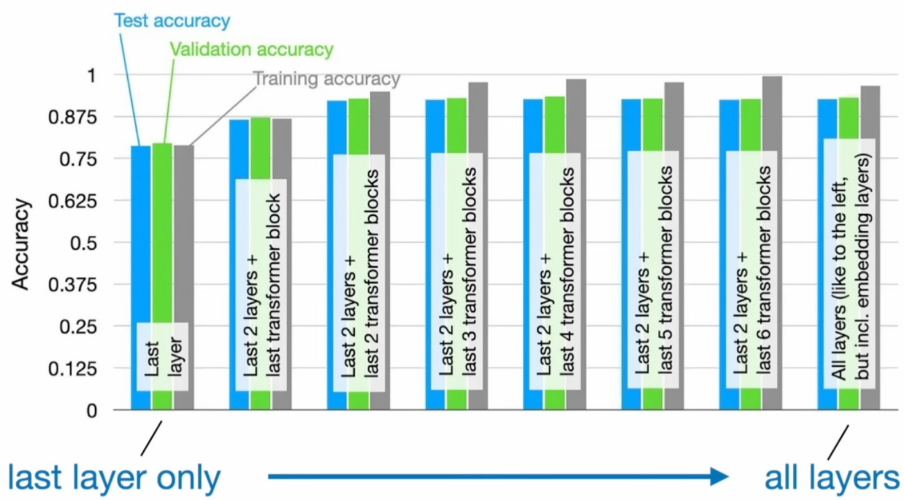
Sebastian Raschka

Building LLMs

55

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

We don't need to finetune all layers

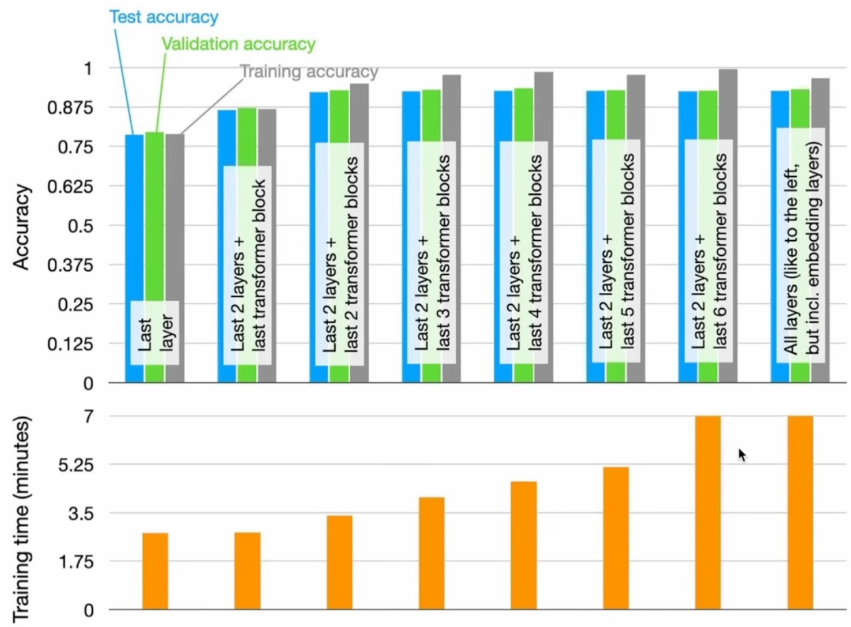


https://magazine.sebastianraschka.com/p/finetuning-large-language-models

Sebastian Raschka Building LLMs 56

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

Training more layers takes more time



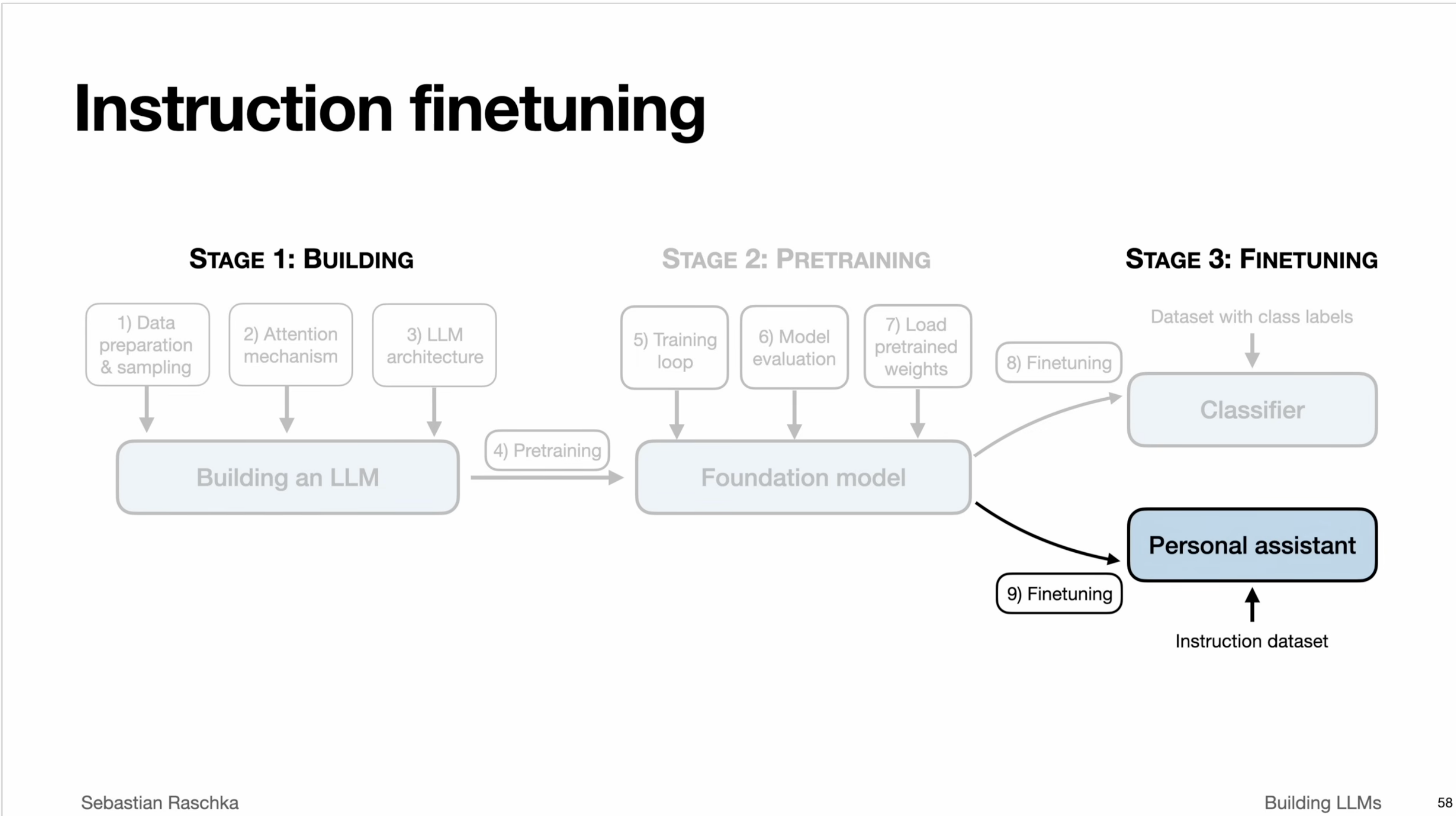
https://magazine.sebastianraschka.com/p/finetuning-large-language-models

Sebastian Raschka Building LLMs 57

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

Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- Parameter Efficient Fine-Tuning: Low-Rank Adaptation

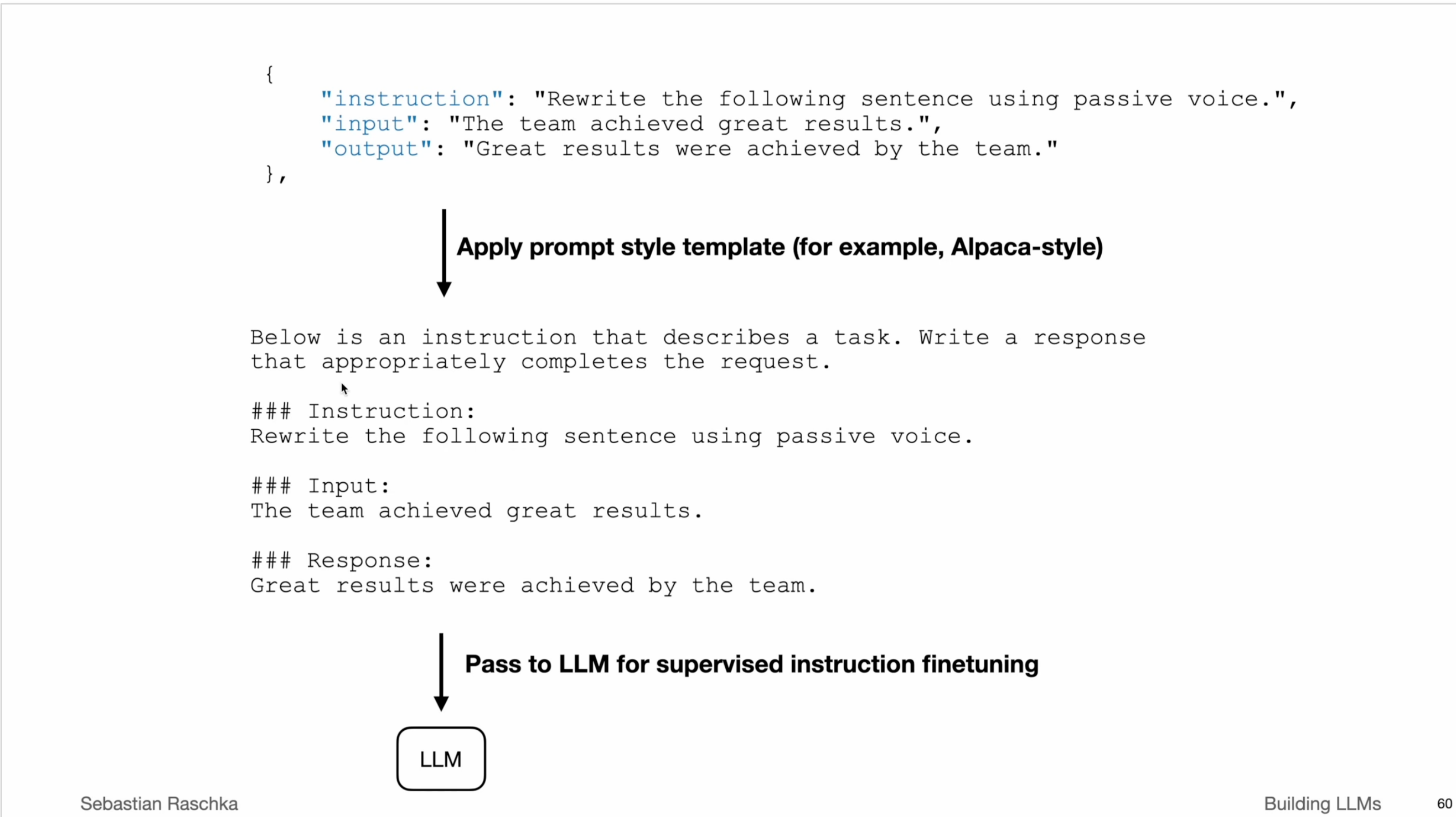


[S. Raschka, "Developing an LLM: Building, Training, Finetuning"]

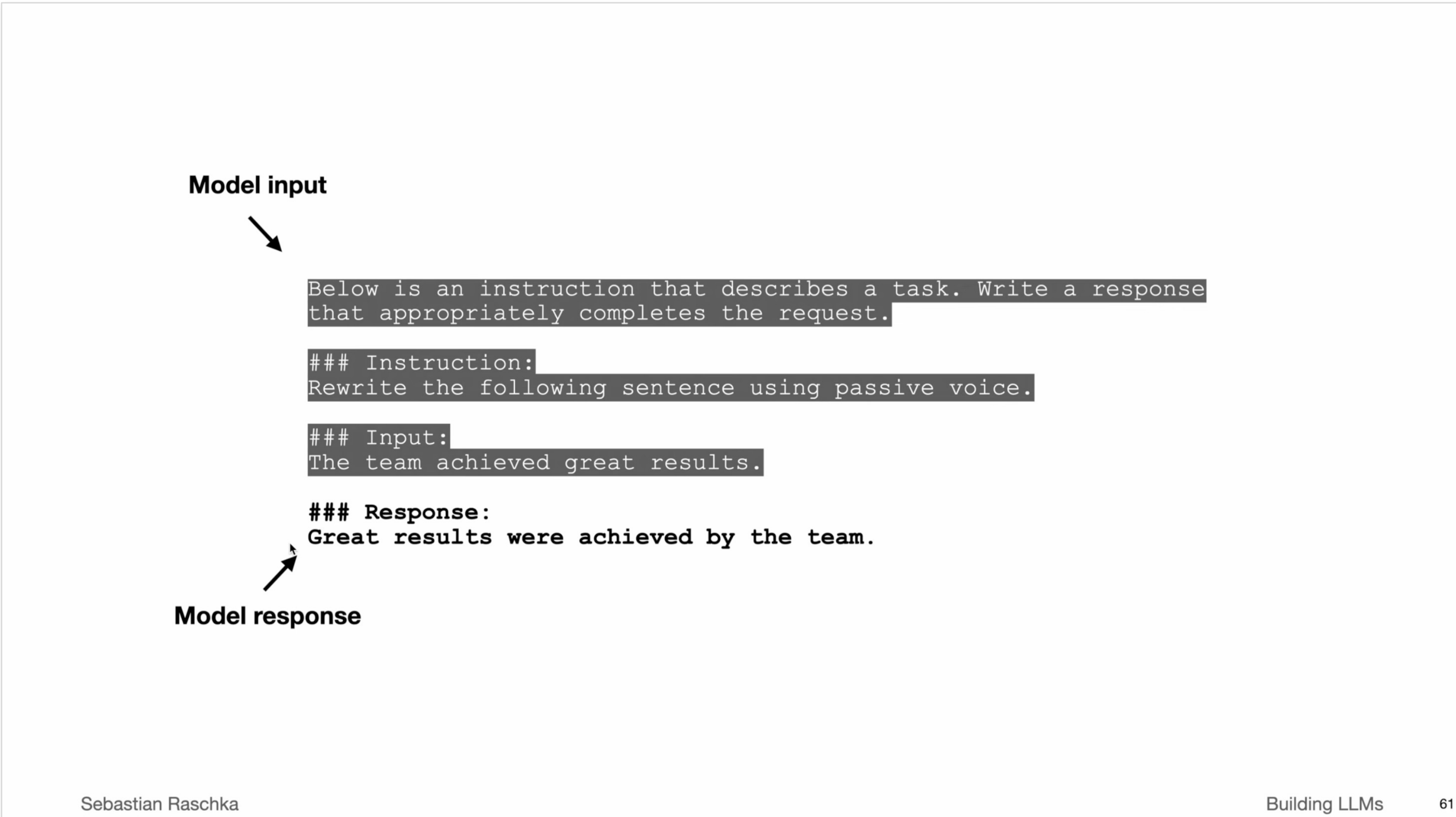
Instruction finetuning datasets

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


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



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



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

Alpaca Instruction Tuning Dataset

Datasets: tatsu-lab/alpaca | like 745 | Follow Tatsu Lab 49

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 card | Data Studio | Files and versions xet | Community 9

Dataset Viewer | Auto-converted to Parquet | API | Embed | Data Studio

Split (1)
train 52k rows

Search this dataset

instruction string · lengths 9 – 489	input string · lengths 0 – 2.47k	output string · lengths 0 – 4.18k	text string · lengths 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

Use this dataset

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

< Papers | arxiv:2305.11206

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.

Datasets: GAIR/lima | like 430 | Follow SII - GAIR 151

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

Dataset card | Data Studio | Files and versions | Community 8

Dataset Viewer — Auto-converted to Parquet | API | Data Studio

Split (2)
train · 1.03k rows

Search this dataset

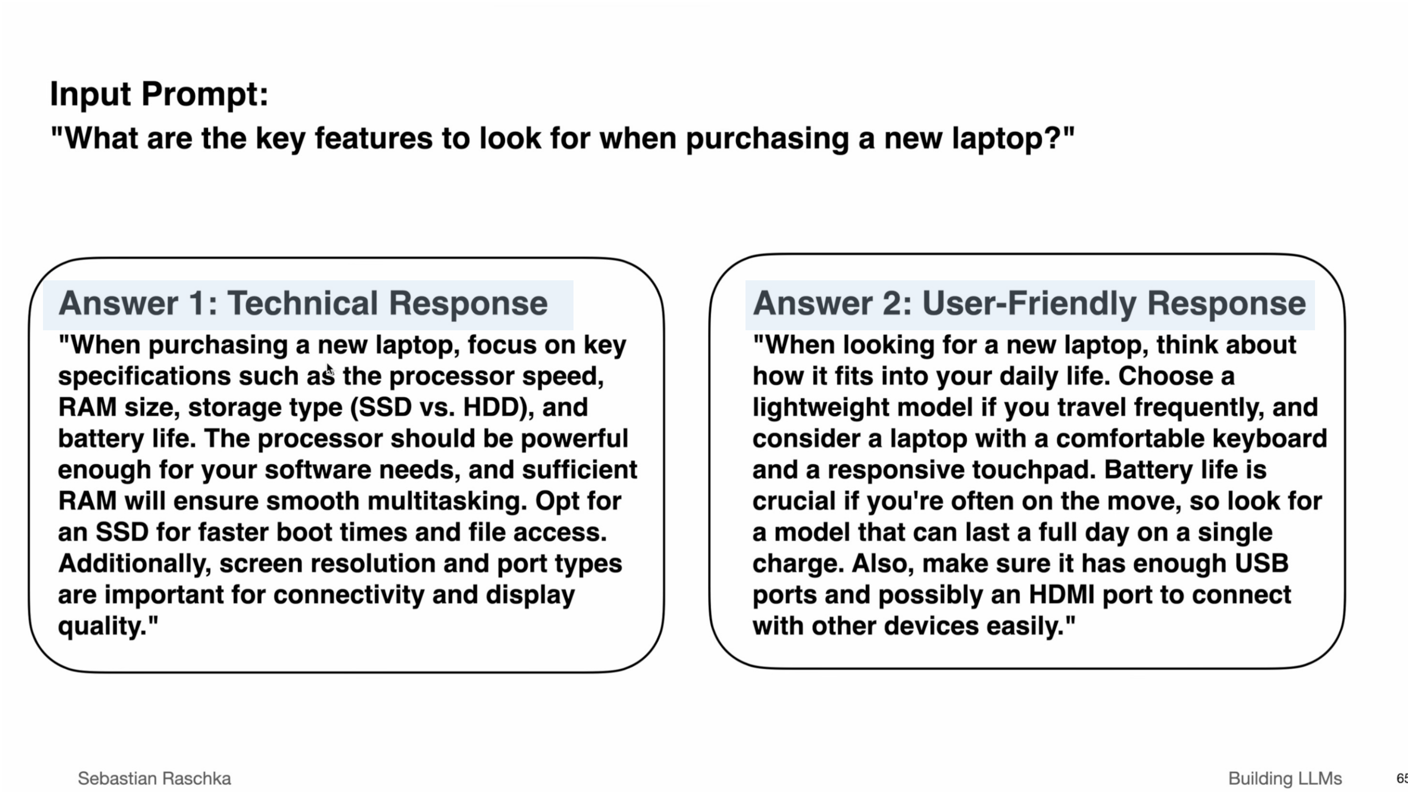
conversations (sequence)	source (string · classes; stackexcha… 38.8%)
["Can brain cells move? By movement I mean long distance migration (preferably within the brain only).", "The question is relatively broad and…	stackexchange
["In our computer systems lecture we were introduced to the MIPS processor. 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

< Previous 1 2 3 ... 11 Next >

Preference Fine Tuning

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

Refine responses for style or safety
Reward preferred responses



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

Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- Parameter Efficient Fine-Tuning: Low-Rank Adaptation

Generative LLM Evaluations

Evaluate for

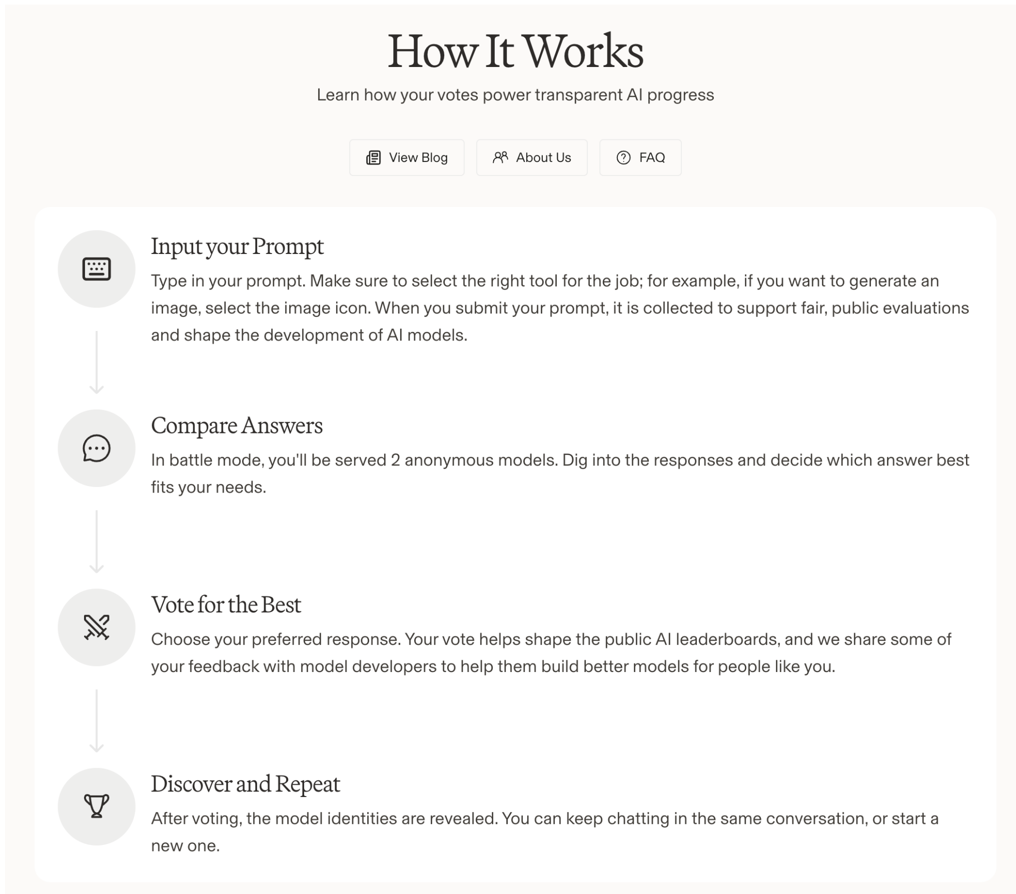
- Accuracy (is it factual or hallucinated?)
- Relevance (is it answering the question?)
- Bias, Toxicity (Is it fair? Or even worse is it racist or toxic?)
- Diversity of Response (does it always give same response? or equally useful diverse responses?)

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

Chatbot Arena -- Crowdsourcing



https://arena.ai

Arena.ai Leaderboard (March, 2026)

Overview | Text | Code | Vision | Document | Text-to-Image | Image Edit | Search | Text-to-Video | Image-to-V

Start Voting

Leaderboard Overview

See how leading AI models stack up across text, image, vision, and more. This page provides a high-level snapshot of each Arena. Explore dedicated tabs for deeper insights. Learn more here.

Text

4 days ago

Rank	Model	Score	Votes
1	claude-opus-4-6-th...	1504	12,730
2	claude-opus-4-6	1500	13,553
3	gemini-3.1-pro-pre...	1493	15,809
4	grok-4.20-beta1	1491	7,378
5	gemini-3-pro	1486	41,631
6	gpt-5.4-high	1484	5,570
7	grok-4.20-beta-030...	1483	5,702
8	gpt-5.2-chat-lates...	1480	11,405
9	gemini-3-flash	1474	30,962
10	claude-opus-4-5-20...	1474	37,448

View all

Code

4 days ago

Rank	Model	Score	Votes
1	claude-opus-4-6	1549	4,264
2	claude-opus-4-6-th...	1545	3,495
3	claude-sonnet-4-6	1523	6,391
4	claude-opus-4-5-20...	1491	13,247
5	claude-opus-4-5-20...	1465	13,559
6	gpt-5.4-high (code...	1457	1,488
7	gemini-3.1-pro-pre...	1455	4,733
8	glm-5	1445	4,265
9	glm-4.7	1439	4,877
10	gemini-3-pro	1438	17,152

View all

Vision

4 days ago

Rank	Model	Score	Votes
1	gemini-3-pro	1286	13,520

Document

19 days ago

Rank	Model	Score	Votes
1	claude-opus-4-6	1524	4,336

https://arena.ai/leaderboard/

LiveBench

A Challenging, Contamination-Free LLM Benchmark

LiveBench appeared as a Spotlight Paper in ICLR 2025.

This work is sponsored by Abacus.AI

Leaderboard | Details | Code | Data | Paper

Introduction

Introducing **LiveBench**: a benchmark for LLMs designed with test set contamination and objective evaluation in mind. It has the following properties:

- LiveBench limits potential contamination by releasing new questions regularly.
- Each question has verifiable, objective ground-truth answers, eliminating the need for an LLM judge.
- LiveBench currently contains a set of 23 diverse tasks across 7 categories, and we will release new, harder tasks over time.

We will evaluate your model on LiveBench! Open a github issue or email us at livebench@livebench.ai!

Leaderboard

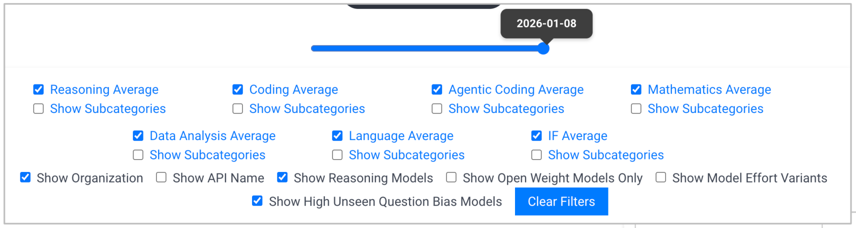
We update questions regularly so that the benchmark completely refreshes every 6 months. Some questions for previous releases are available here. The most recent version is **LiveBench-2026-01-08**. This version features a new mathematical task and a new data analysis task.

https://livebench.ai

March, 2026

LiveBench

A Challenging, Contamination-Free LLM Benchmark

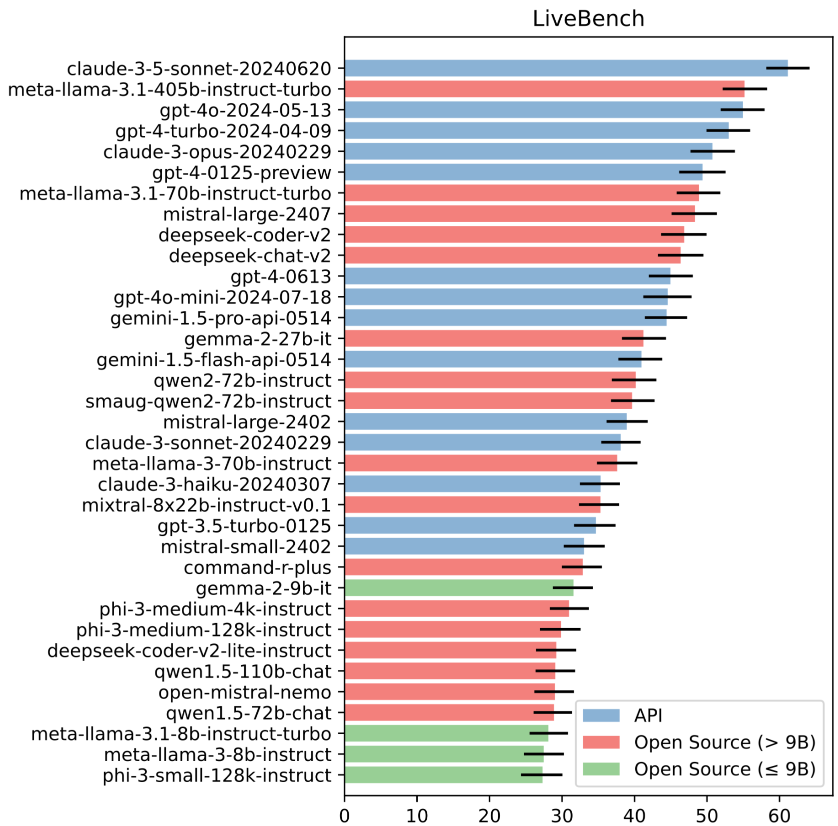


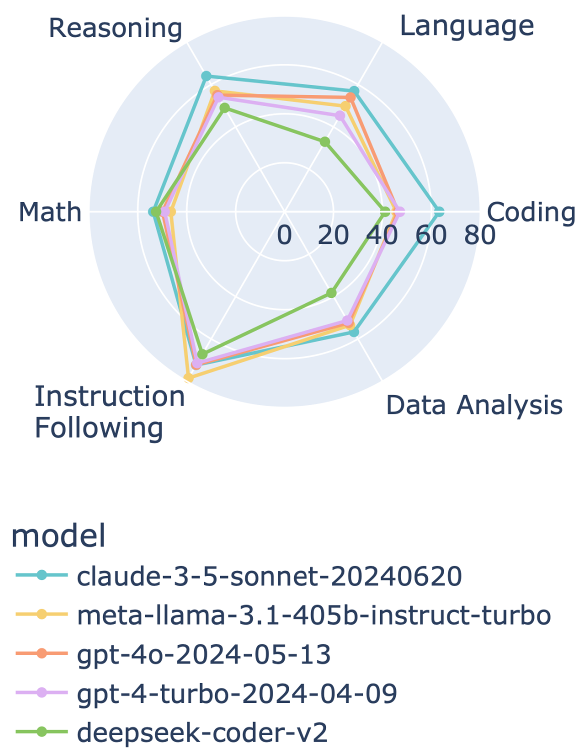
Model	Organization	Global Average	Reasoning Average	Coding Average	Agentic Coding Average	Mathematics Average	Data Analysis Average	Language Average	IF Average
GPT-5.4 Thinking xHigh Effort	OpenAI	80.28	88.12	77.54	70.00	94.15	79.31	82.63	70.22
Gemini 3.1 Pro Preview High* *5th rank in unseen questions across all categories	Google	79.93	84.00	76.45	65.00	91.04	78.54	85.38	79.10
Claude 4.6 Opus Thinking High Effort	Anthropic	76.33	88.67	78.18	61.67	89.32	69.89	83.27	63.31
Claude 4.5 Opus Thinking High Effort	Anthropic	75.96	80.09	79.65	63.33	90.39	74.44	81.26	62.55
Claude 4.6 Sonnet Thinking Medium Effort	Anthropic	75.47	84.77	79.27	60.00	86.99	77.95	76.10	63.22
GPT-5.2 High	OpenAI	74.84	83.21	76.07	51.67	93.17	78.16	79.81	61.77
GPT-5.2 Codex	OpenAI	74.30	77.71	83.62	51.67	88.77	78.20	73.68	66.45
GPT-5.1 Codex Max High	OpenAI	73.98	83.65	80.68	53.33	83.22	70.12	76.48	70.38
Gemini 3 Pro Preview High	Google	73.39	77.42	74.60	55.00	81.84	74.39	84.62	65.85
GPT-5.3 Codex High	OpenAI	72.76	80.15	78.18	55.00	87.84	62.69	80.09	65.38
Gemini 3 Flash	Google	72.40	74.55	73.90	40.00	84.17	74.77	84.56	74.86

https://livebench.ai

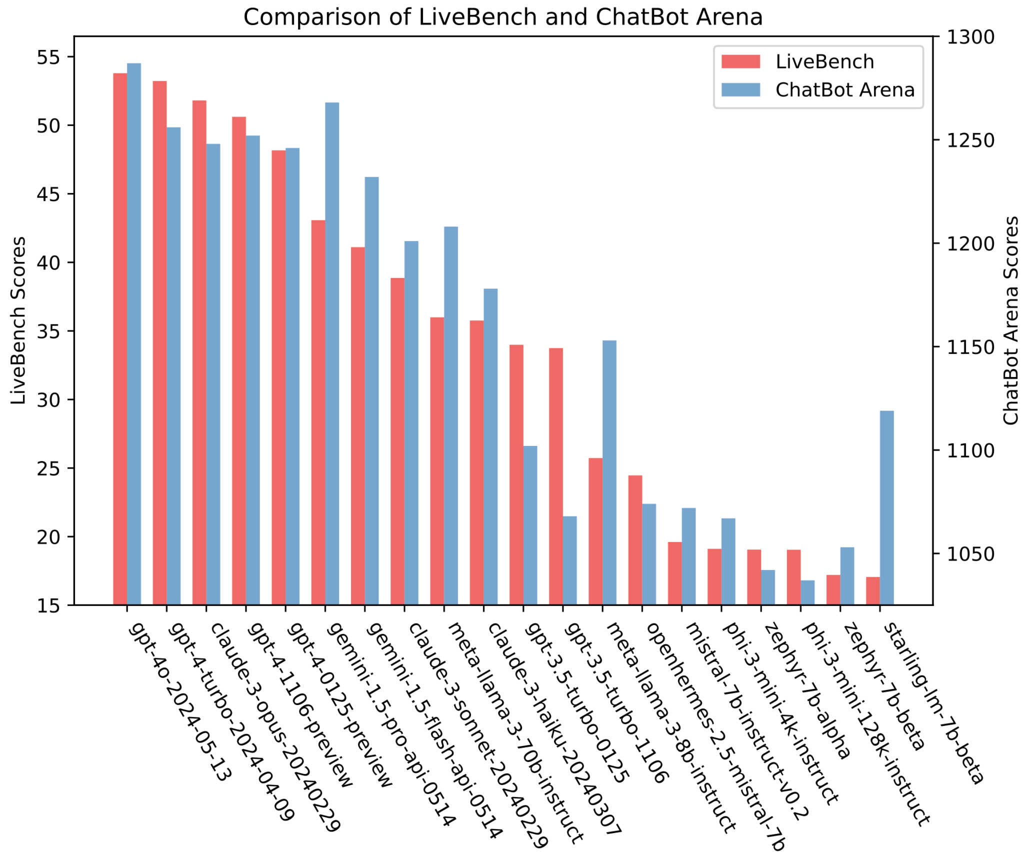
LiveBench

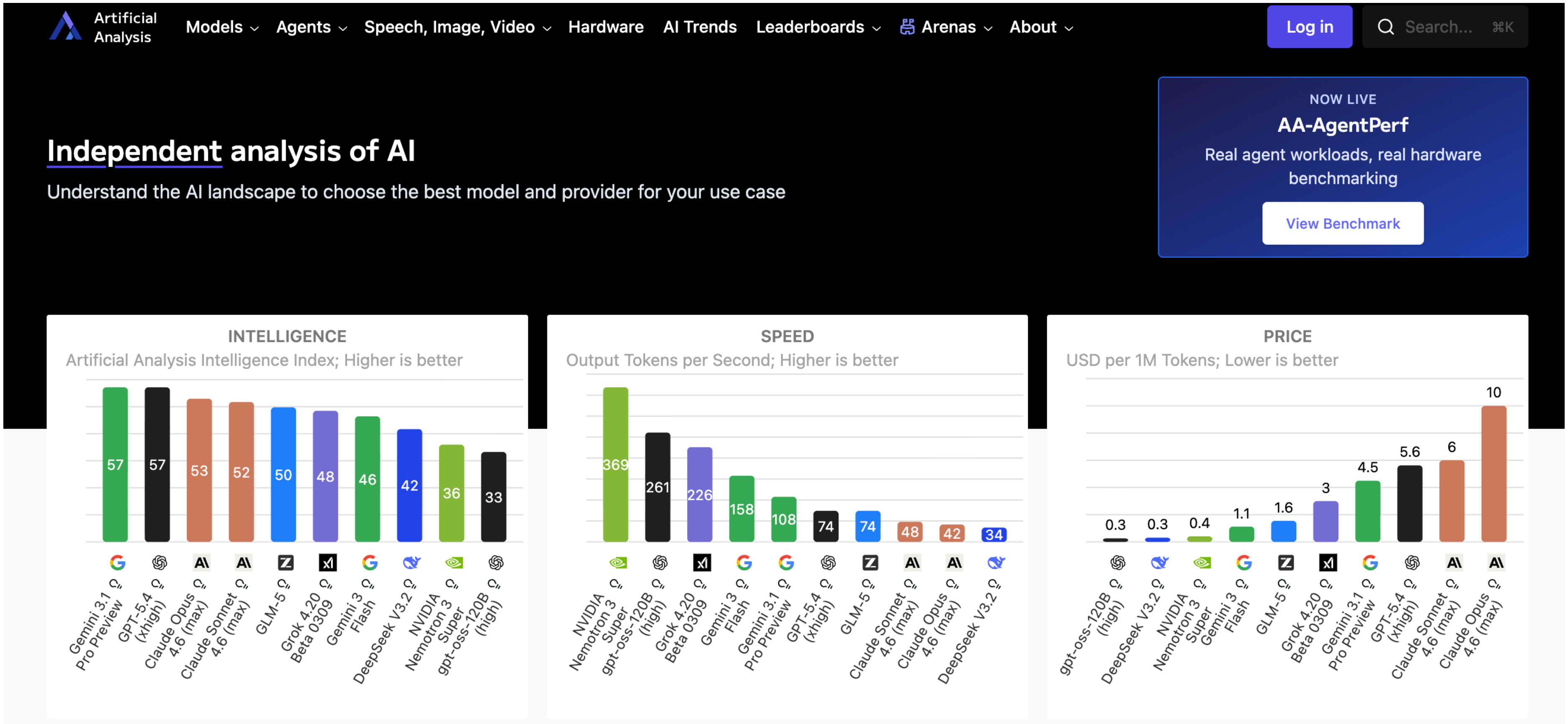
A Challenging, Contamination-Free LLM Benchmark





LiveBench
A Challenging, Contamination-Free LLM Benchmark

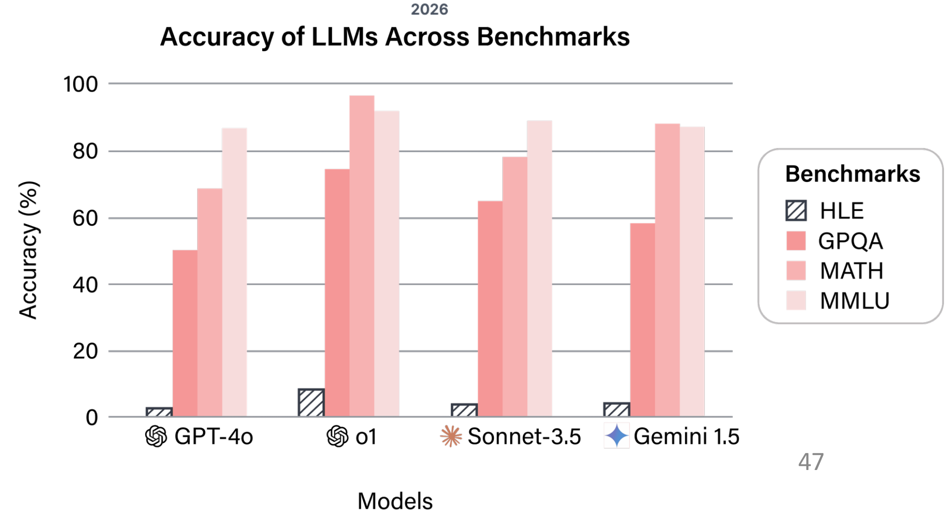




https://artificialanalysis.ai/

Humanity's Last Exam

Source: CAIS AI Dashboard



Introduction

Benchmarks are important tools for tracking the rapid advancements in large language model (LLM) capabilities. However, benchmarks are not keeping pace in difficulty: LLMs now achieve over 90% accuracy on popular benchmarks like MMLU, limiting informed measurement of state-of-the-art LLM capabilities. In response, we introduce Humanity's Last Exam, a multi-modal benchmark at the frontier of human knowledge, designed to be the final closed-ended academic benchmark of its kind with broad subject coverage. The dataset consists of 2,500 challenging questions across over a hundred subjects. We publicly release these questions, while maintaining a private test set of held out questions to assess model overfitting.

Humanity's Last Exam

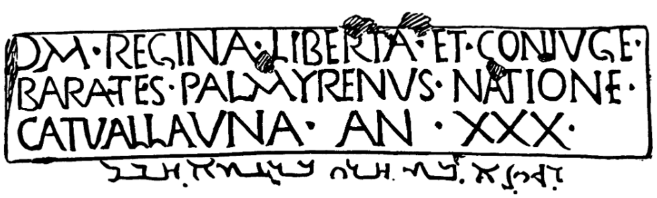
Dataset

Humanity's Last Exam (HLE) is a global collaborative effort, with questions from nearly 1,000 subject expert contributors affiliated with over 500 institutions across 50 countries – comprised mostly of professors, researchers, and graduate degree holders.

Examples 1-2/8

Classics

Question:



Here is a representation of a Roman inscription, originally found on a tombstone. Provide a translation for the Palmyrene script. A transliteration of the text is provided: RGYNᵓ BT ḤRY BR ʿTᵓ ḤBL

Henry T
Merton College, Oxford

Ecology

Question:
Hummingbirds within Apodiformes uniquely have a bilaterally paired oval bone, a sesamoid embedded in the caudolateral portion of the expanded, cruciate aponeurosis of insertion of m. depressor caudae. How many paired tendons are supported by this sesamoid bone? Answer with a number.

Edward V
Massachusetts Institute of Technology

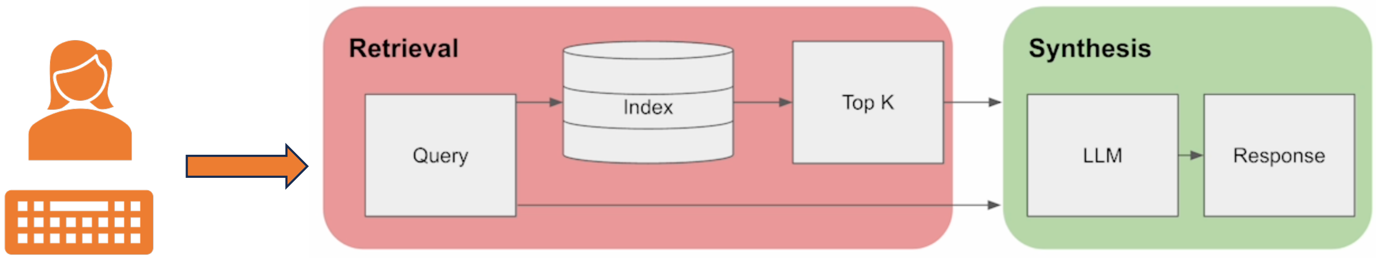
Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- 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)



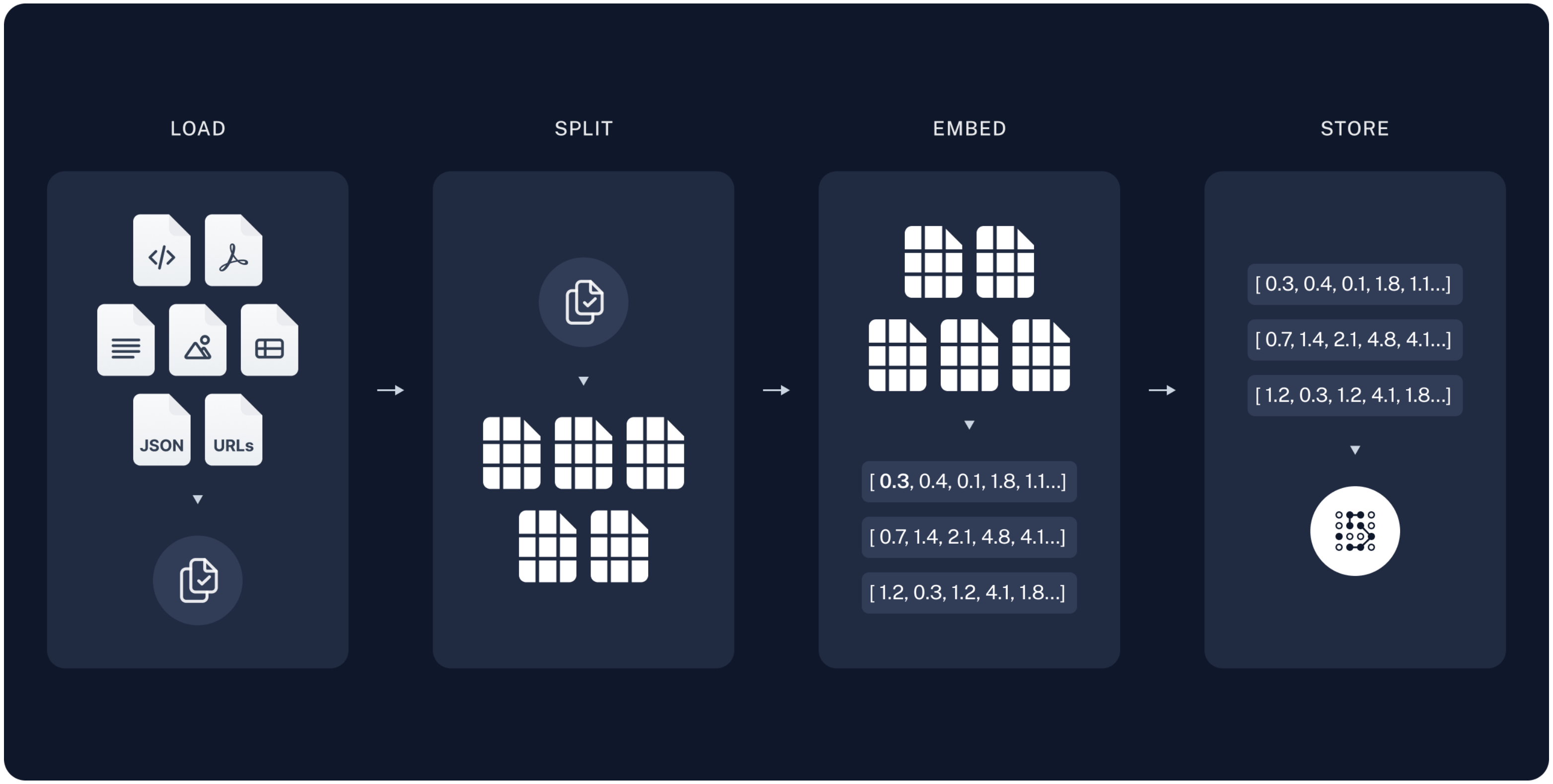
RAG Architecture

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

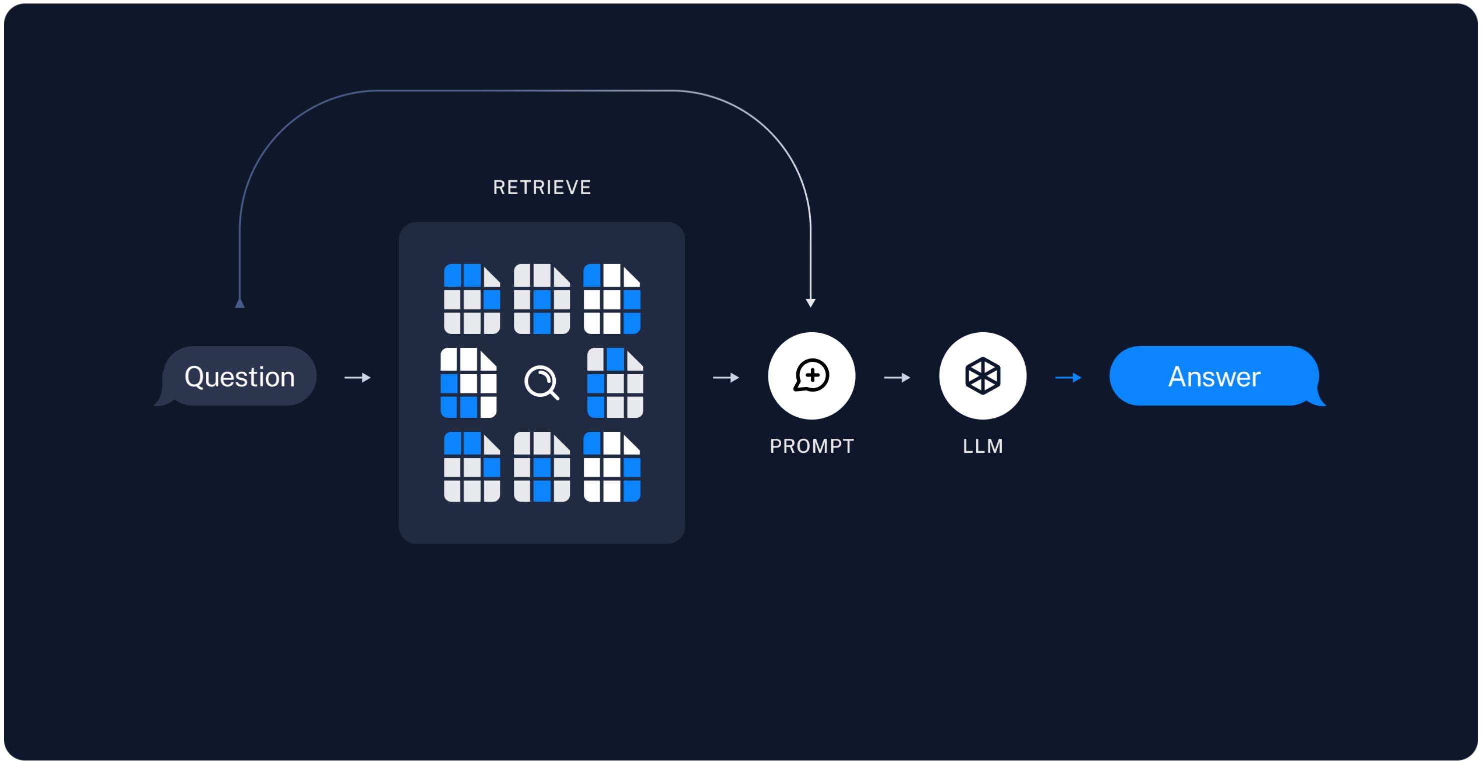
https://python.langchain.com/docs/use_cases/question_answering/

RAG – Loading and Indexing



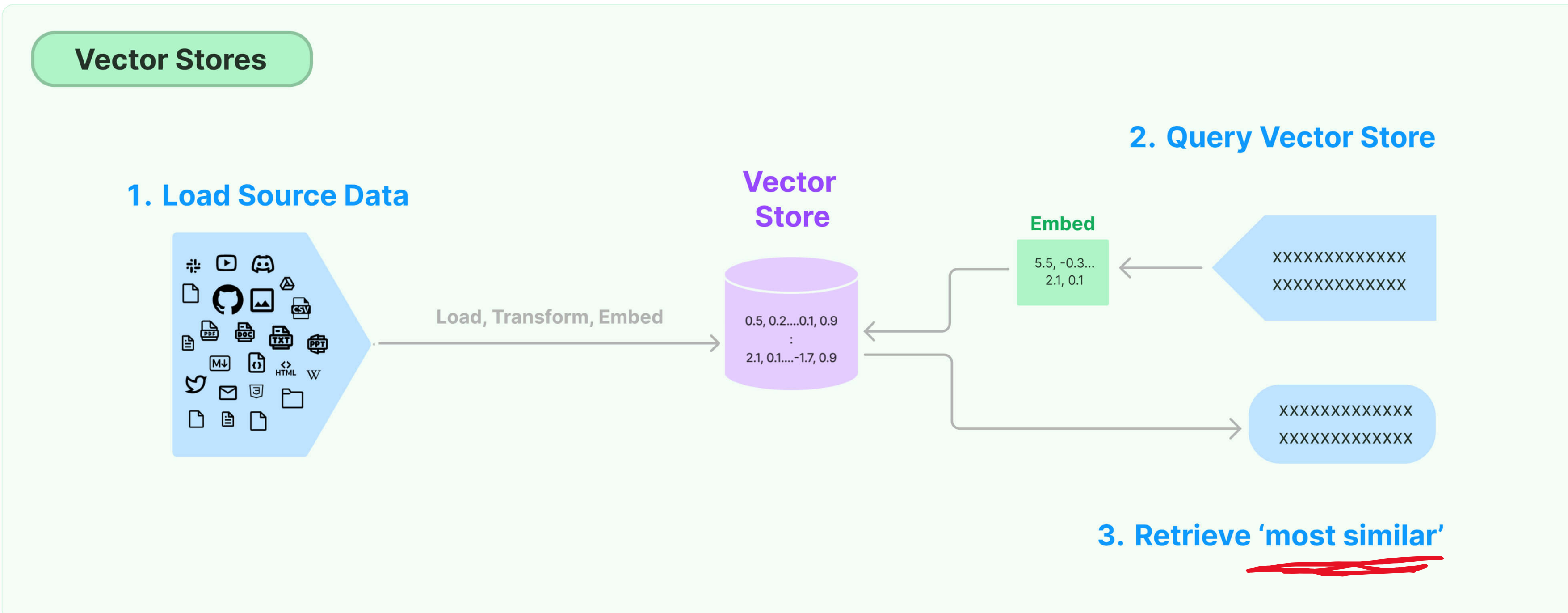
https://python.langchain.com/docs/use_cases/question_answering/

RAG – Retrieval and Generation



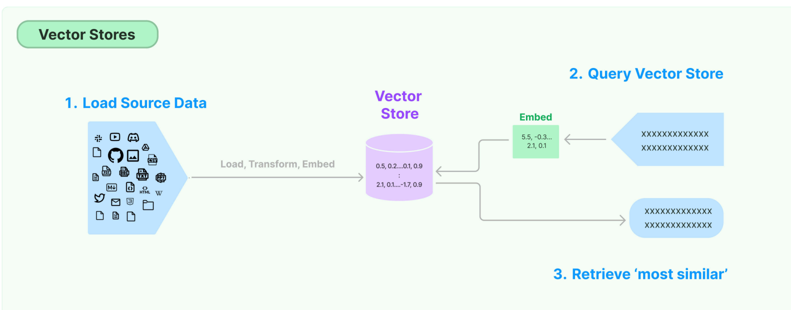
https://python.langchain.com/docs/use_cases/question_answering/

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

* Default on Chroma Vector Database

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

Is simple similarity measure between query and document the best approach?

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 models for 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:

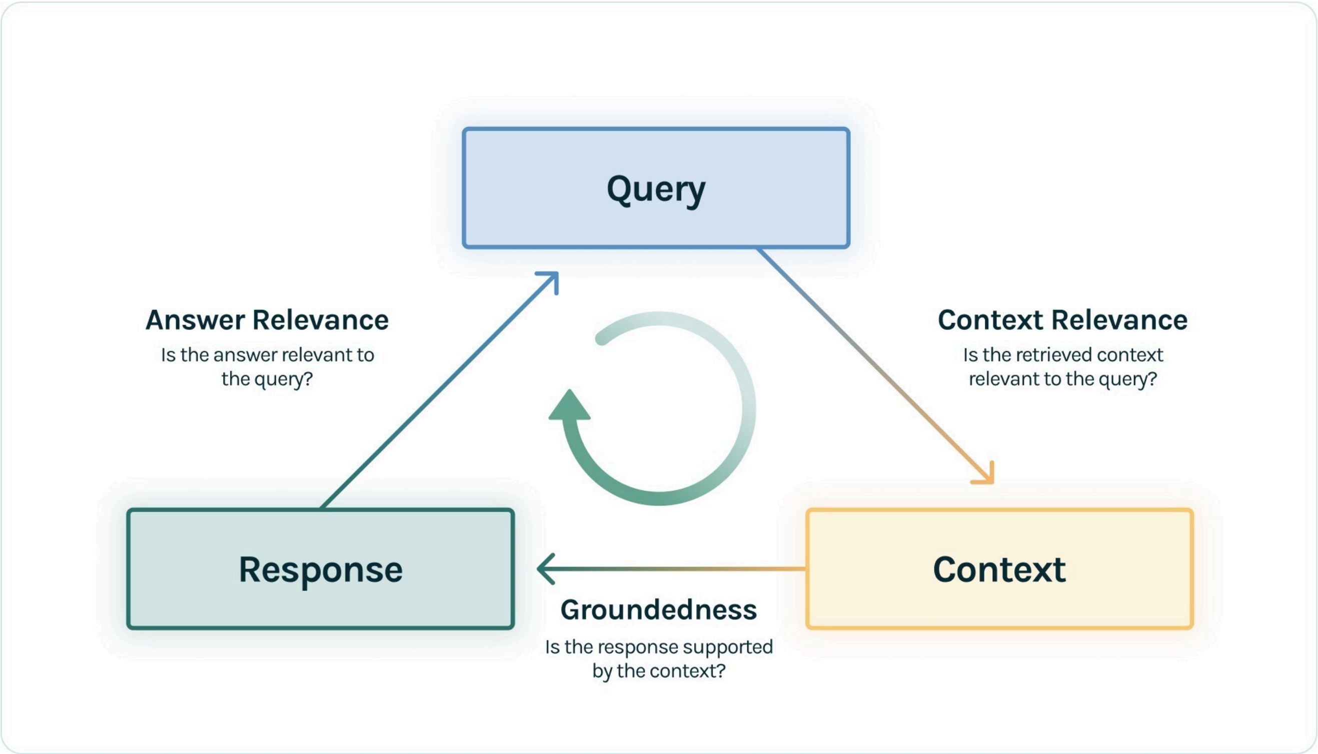
Humeau, S., Shuster, K., Lachaux, M. A., & Weston, J. (2019). Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. arXiv:1905.01969. The poly-encoder architecture introduced here incorporates aspects of both dual and cross-encoders, offering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

4. Semantic Search Systems

Semantic Search with Transformers:

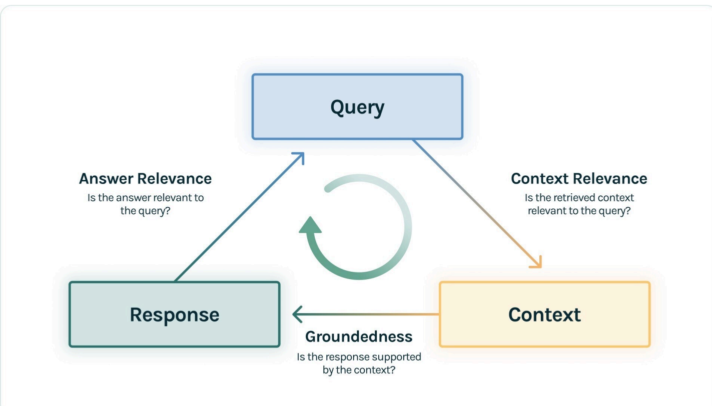
Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., & Cheng, X. (2020). A Deep Look into Neural Ranking Models for Information Retrieval. Information Processing & Management. This review covers deep learning approaches to information retrieval, including the use of Transformer 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



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

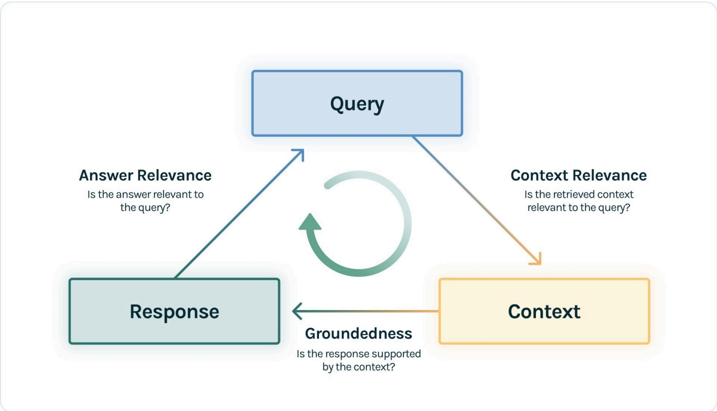
Evaluating RAG: Context Relevance



- 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/

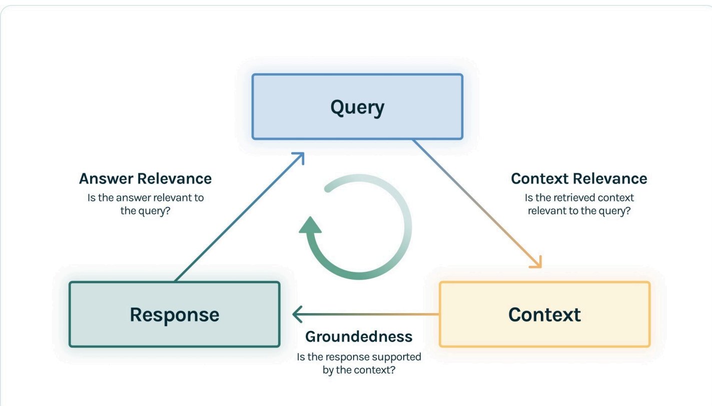
Evaluating RAG: Groundedness



- 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



- 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.

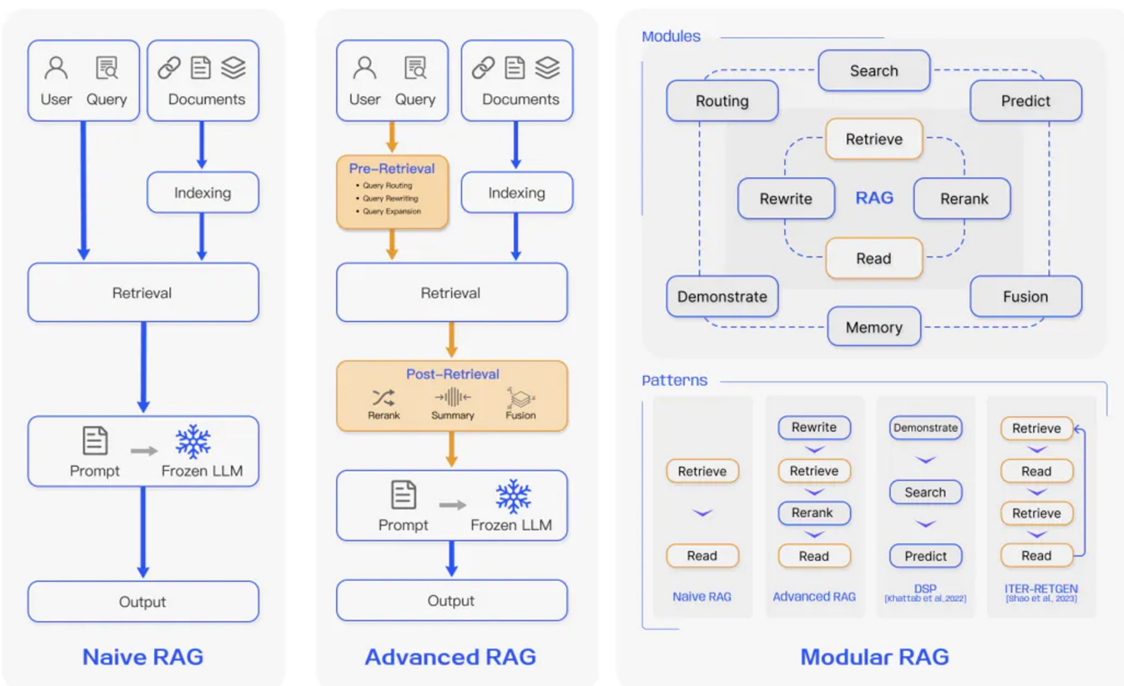


Image source (Kojima et al., 2022)

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.

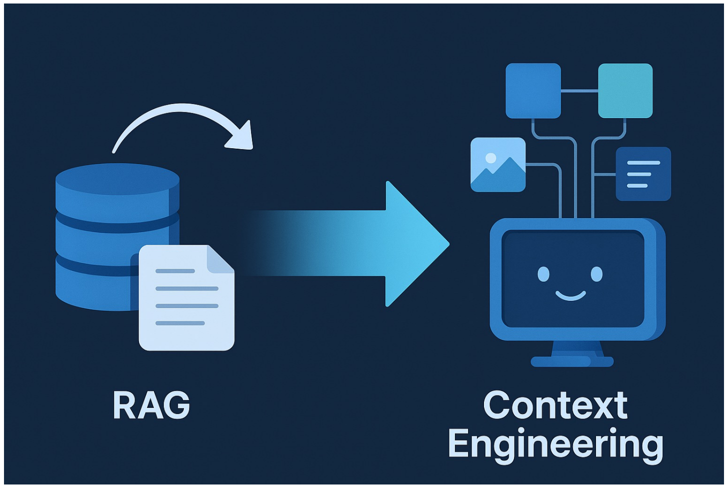
Retrieval Enhancements
- Explore alternative search methods (e.g., full-text, graph-based).
- Experiment with different embedding models for task suitability.
- Implement hierarchical and recursive search for precision.

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

From RAG to Context Engineering



The shift from **Retrieval-Augmented Generation (RAG)** to **Context Engineering** marks an evolution from simply "finding and dumping" data to "curating and orchestrating" the entire state of an AI model.

While RAG is a technique for adding external knowledge, context engineering is the broader architectural discipline of managing everything a model "sees" before generating a response.

Key Pillars of Context Engineering

Instead of just increasing the "Top-K" results, context engineering uses several advanced strategies to improve performance:

- **Context Orchestration**: Deciding which combination of data, tools, and memories are relevant for a specific reasoning step.
- **Precision Filtering**: Using various techniques to remove irrelevant noise from retrieved chunks before they reach the model.
- **Dynamic Routing**: A "context engine" or router that plans the task and picks the right tools or data sources (e.g., Model Context Protocol (MCP)) before execution.
- **Memory Management**: Distilling facts from past conversations into long-term memory while keeping recent, relevant notes in short-term context.
- **Contextual RAG**: A method popularized by Anthropic that adds specific context to chunks during ingestion to ensure the retriever understands the "big picture" of a single snippet.

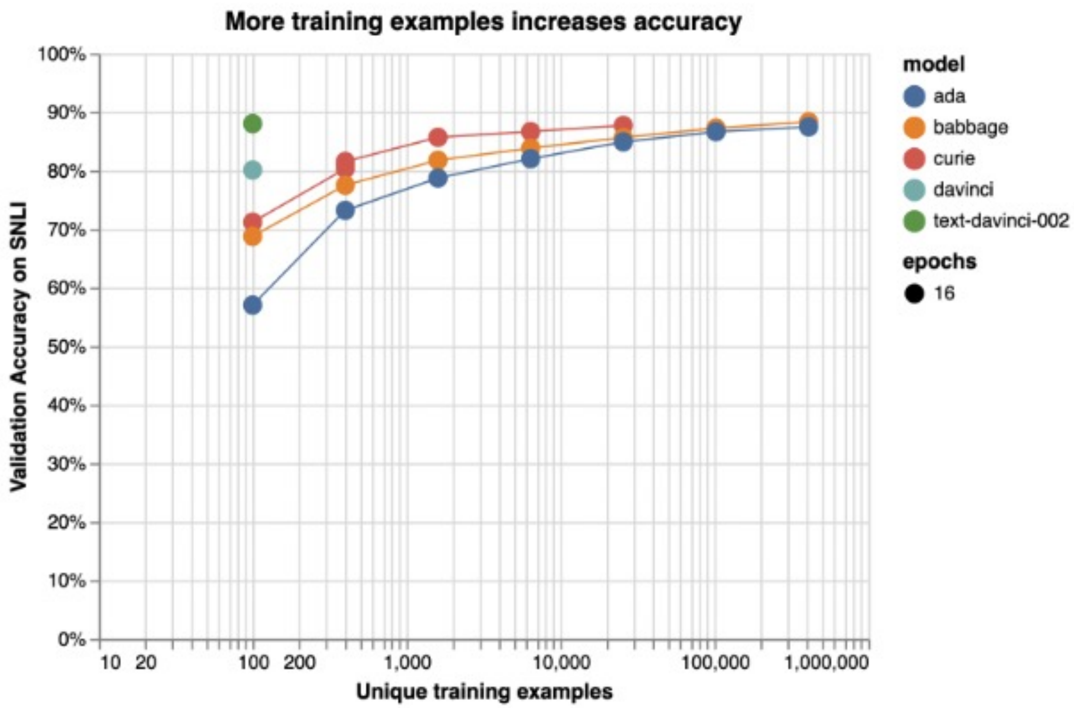
Topics

- LLM Training Process
  - Pre-training
  - Classifier Fine-Tuning
  - Instruction (Chat) Fine-Tuning
  - Preference Tuning
- Evaluating LLMs
- Improving LLMs with RAG
  - Evaluating LLMs
- 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

🤗 HuggingFace – Fine-tune Pretrained Model Tutorials

- Finetune for Sentiment Analysis Example
  - https://huggingface.co/docs/transformers/training
  - Finetune bert-base-cased (109M params, FP32, 436MB) on Yelp review dataset (650K reviews, 323 MB)
- Finetune for text classification example
  - https://github.com/huggingface/notebooks/blob/main/examples/text_classification.ipynb
  - preprocess the data and fine-tune a pretrained model on any GLUE task
- Finetune for question answering
  - https://github.com/huggingface/notebooks/blob/main/examples/question_answering.ipynb
  - preprocess the data and fine-tune a pretrained model on SQUAD

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

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

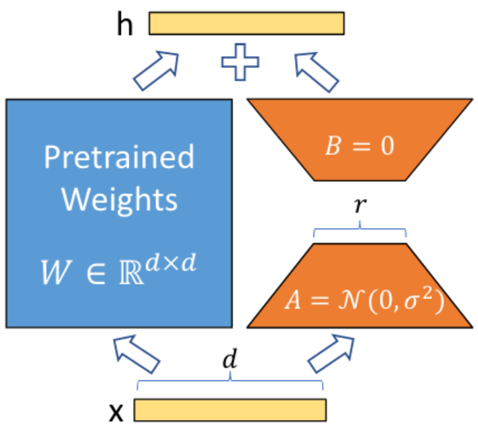
- Low rank adaptation of the weight updates -- LoRA
- Train and concatenated soft prompts -- Prompt Tuning

Topics

- Full finetuning
- Low rank adaptation

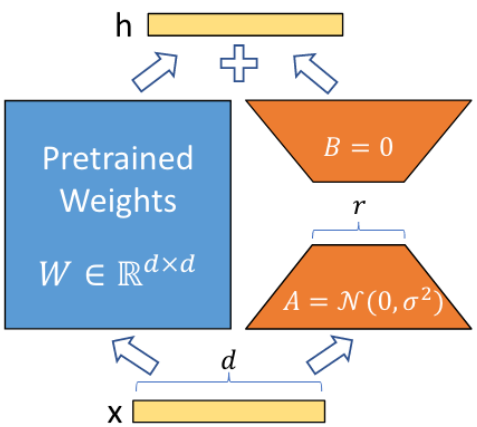
Low Rank Adaptation

- 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



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



- Aghajanyan et al show that pretrained language models have a low "intrinsic dimension"
- 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

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

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

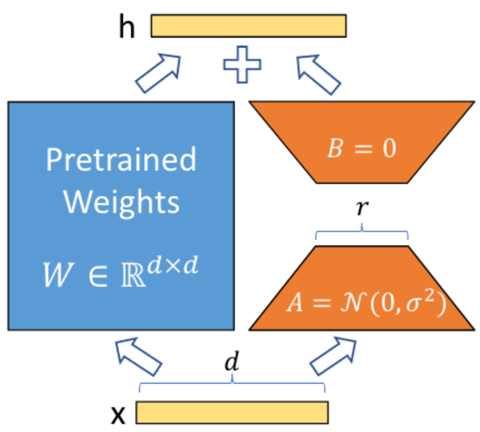
E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

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.

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

RoBERTa {base,large}
DeBERTa XXL

Full Fine-Tune

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

Full Fine-Tune

Model & Method	# Trainable Parameters	BLEU	NIST	MET	ROUGE-L	CIDEr
		E2E NLG Challenge				
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$^{\mathrm{Top2}}$)*	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	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
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	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

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 Rank r	W_q 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	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

To Dive Deeper

From Sebastian Raschka

- LLM Training: RLHF and its Alternatives
- LLMs from Scratch book and repo
- Understanding Reasoning LLMs (CoT, DeepSeek, etc.)

Next Time

- Image, vision and multimodal transformers

Feedback

https://forms.gle/pXHM5nx1Ti9aFmpw6