



Improving Generative LLMs

DL4DS – Spring 2025

April/May Dates



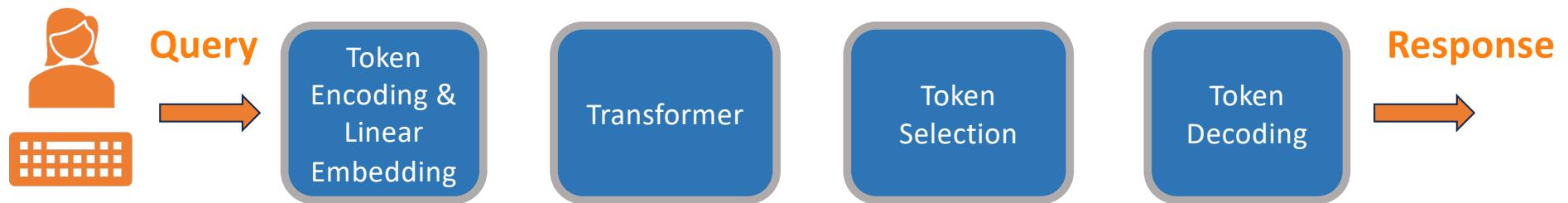
You are here

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		April 1	2	3 Xformers Part 2	4	5
6	7	8 Industry Talk	9	10 Improving LLMs	11	12
13	14	15 LLM RLHF	16	17 GANs	18	19
20	21	22 VAEs	23	24 Diffusion Models	25	26
27	28	29 ★ Project Presentations 1 ★	30	May 1 ★ Project Presentations 2 ★	2	3
4	5 Project Reports Due	6	7	8	8	10
				Finals Week		

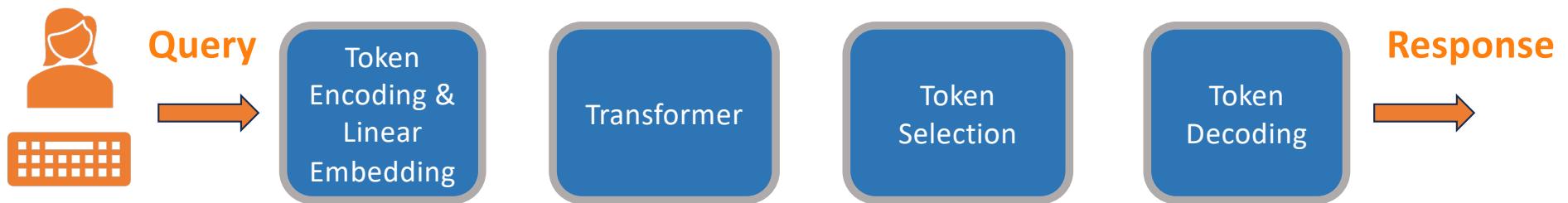
Topics

- 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

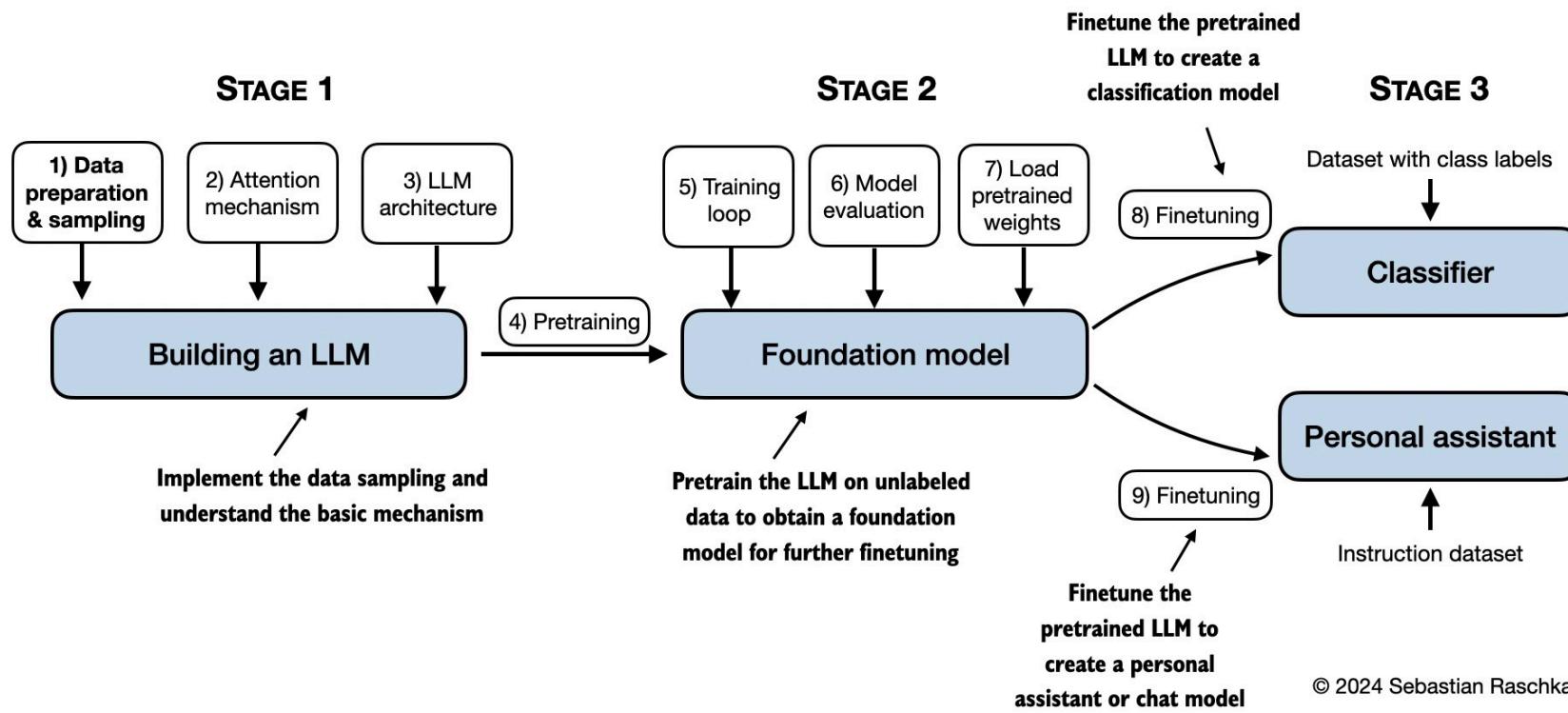
LLM Generative Flow

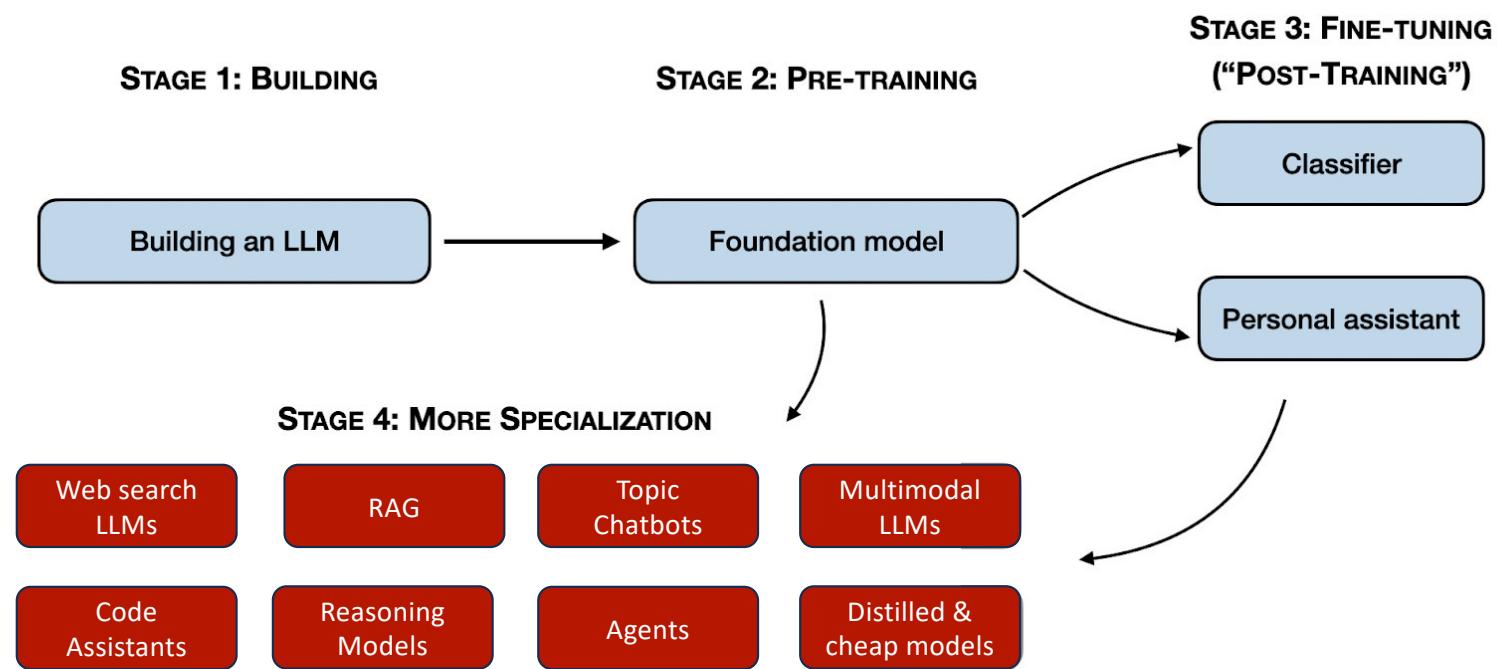


LLM Generative Flow



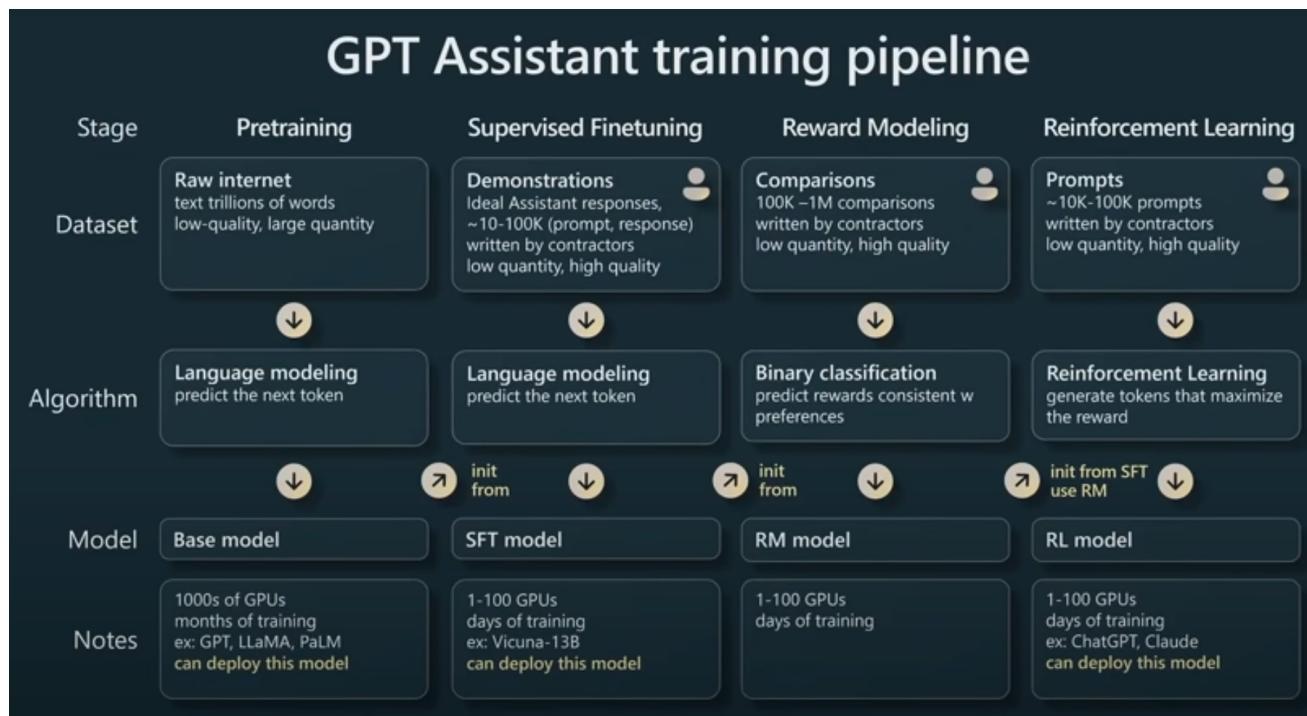
- How do we improve the response?
- How do we evaluate the response?





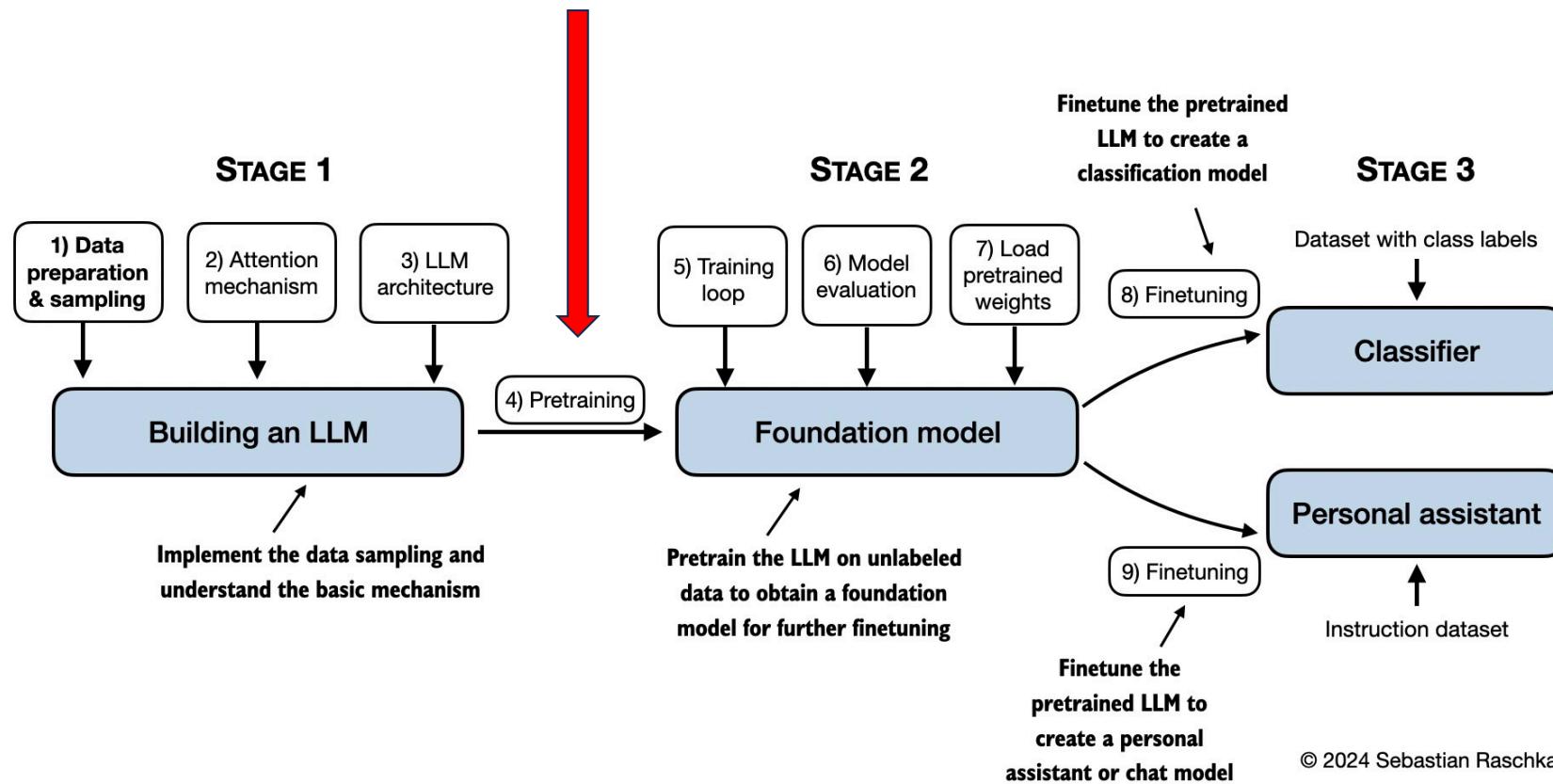
<https://magazine.sebastianraschka.com/p/understanding-reasoning-langs>

How do we build a chat model?



[State of GPT, Andrej Karpathy, MS Build Keynote](#)

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

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

Llama 1 was trained on 1.4T tokens

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

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

LLaMA: Open and Efficient Foundation Language Models (2020), <https://arxiv.org/abs/2302.13971>

Llama 2 was trained on 2T tokens

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

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

Llama 3 was trained on 15T tokens

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

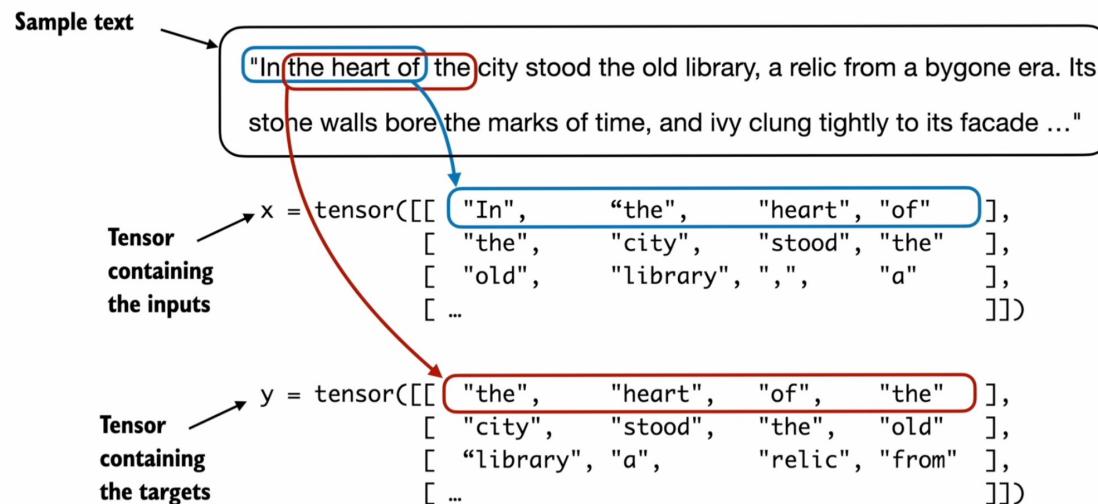
Introducing Meta Llama 3: The most capable openly available LLM to date (2024), <https://ai.meta.com/blog/meta-llama-3/>

Quantity vs quality

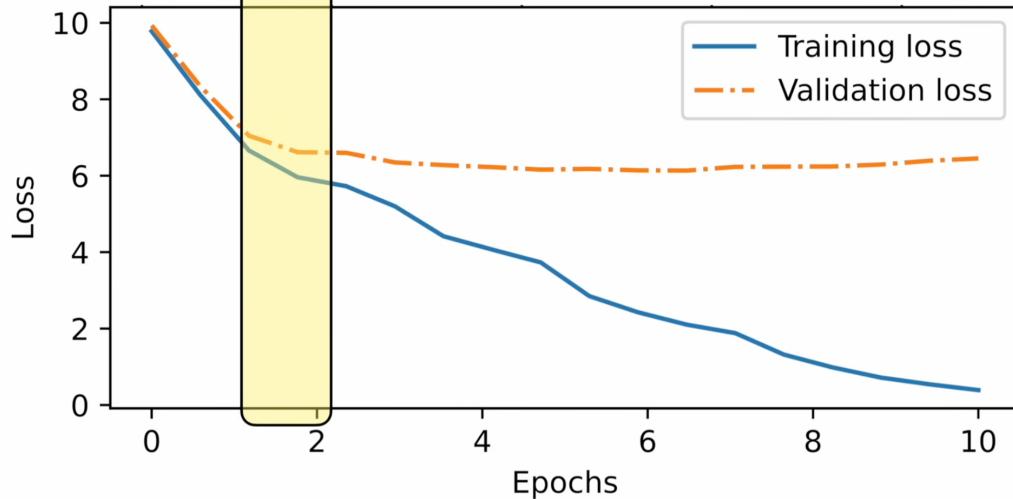
“we mainly focus on the **quality of data** for a given scale. We try to calibrate the training data to be closer to the “data optimal” regime for small models. In particular, we filter the publicly available web data to contain the correct level of “knowledge” and keep more web pages that could potentially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

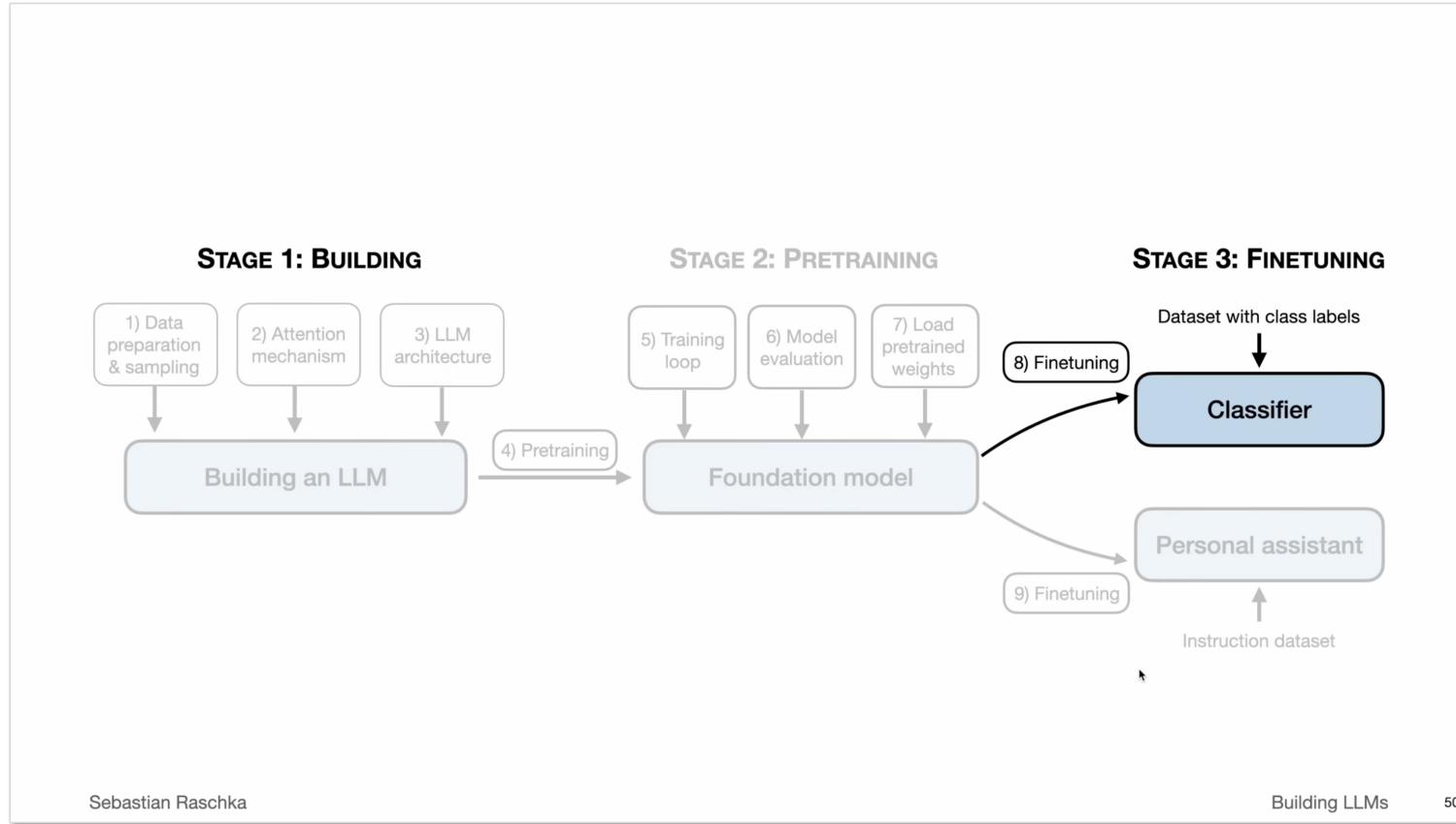
Labels are the inputs shifted by +1



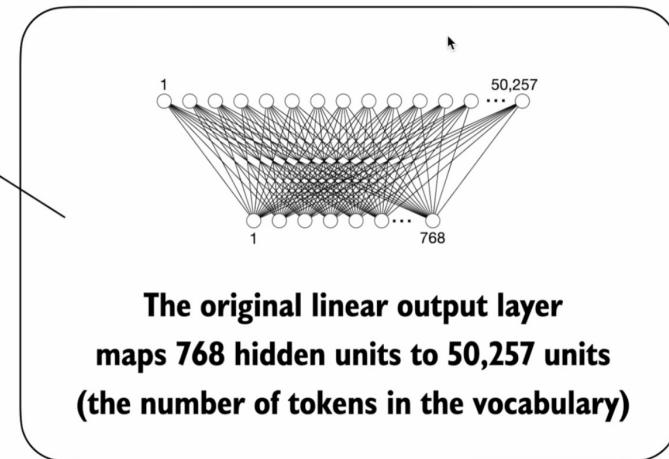
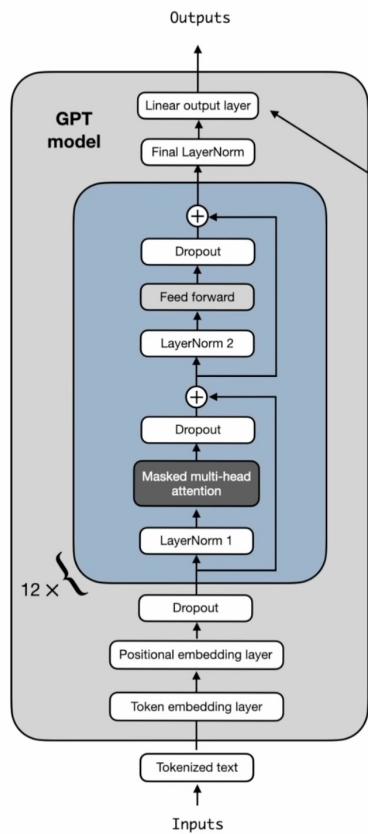
Training for ~1-2 epochs is usually a good sweet spot



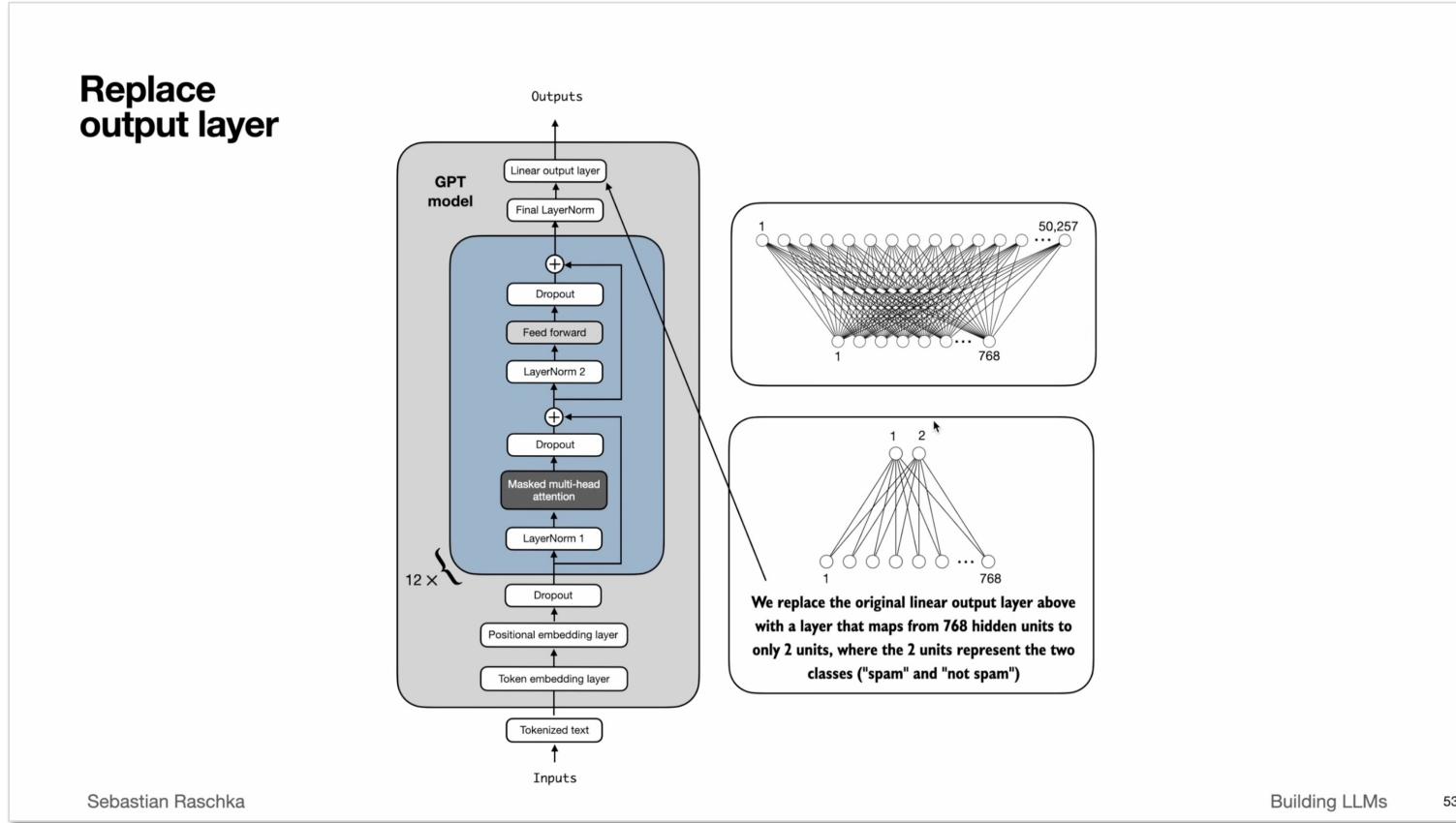
Classifier Finetuning



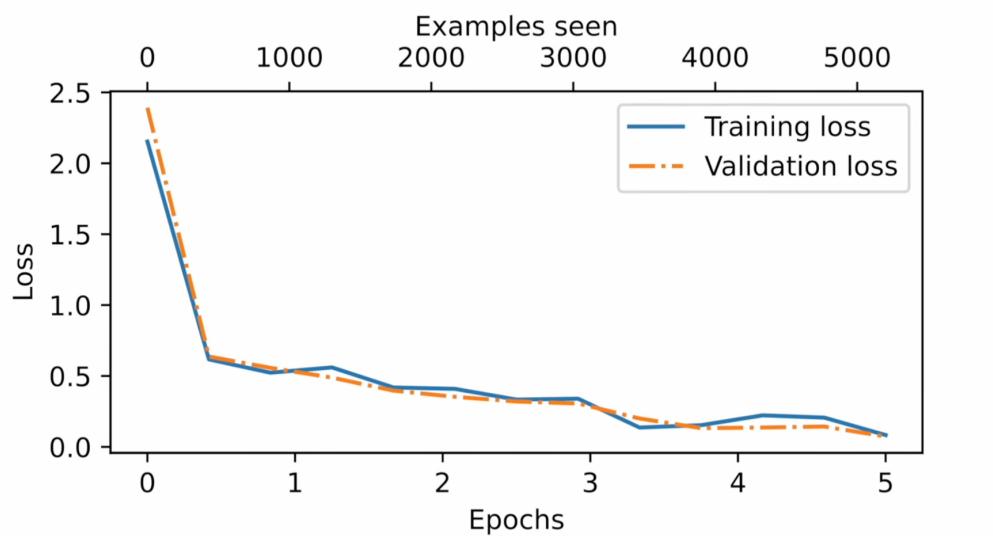
Replace output layer



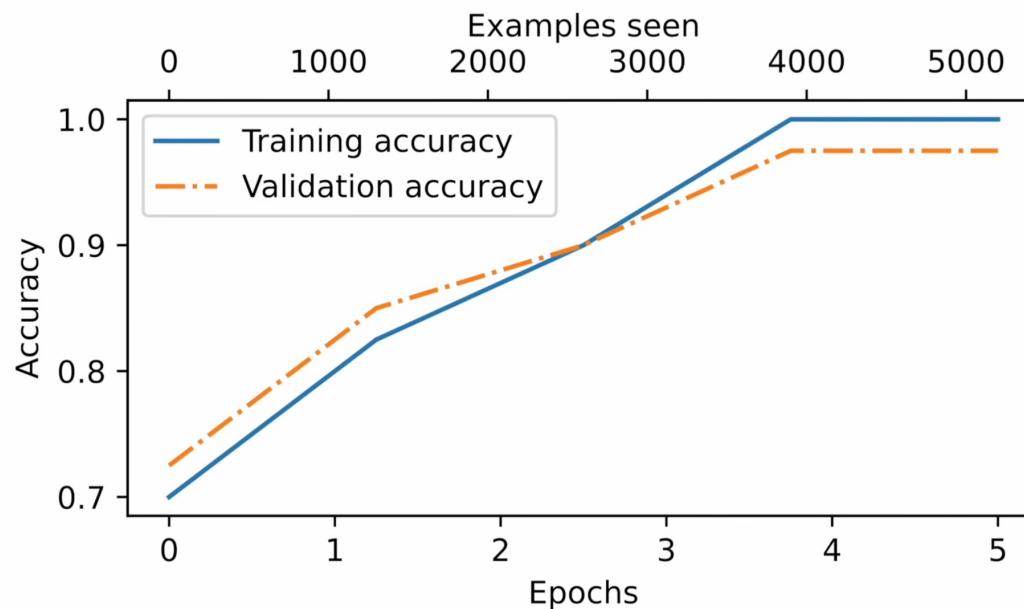
Example: Spam/Ham Classifier



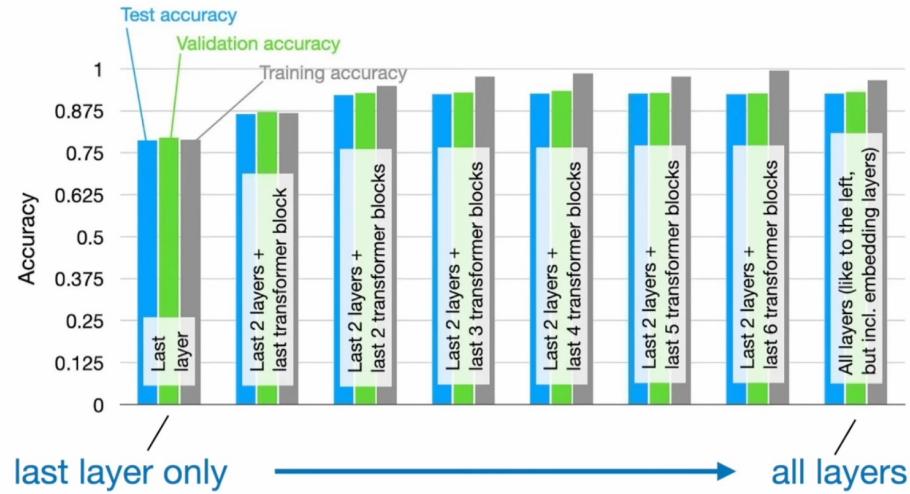
Track loss values as usual



In addition, look at task performance

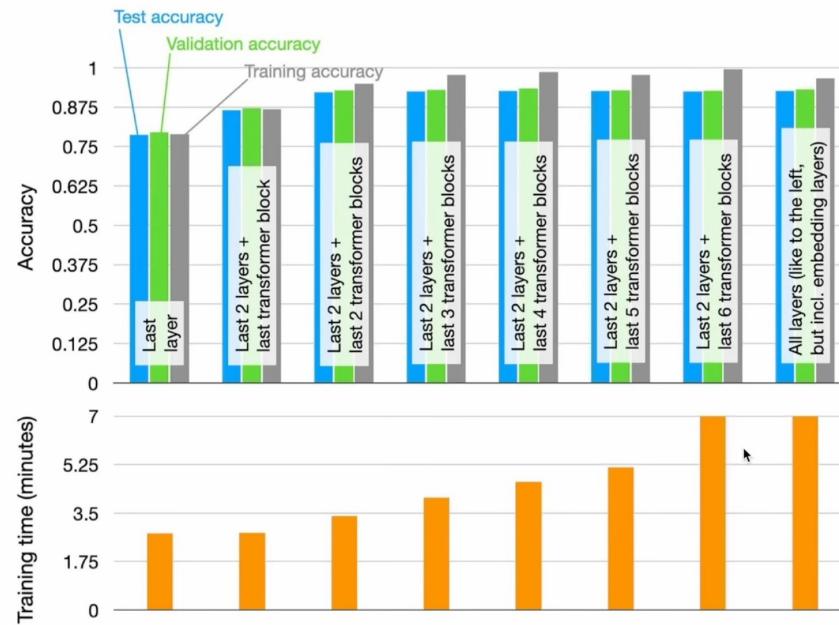


We don't need to finetune all layers



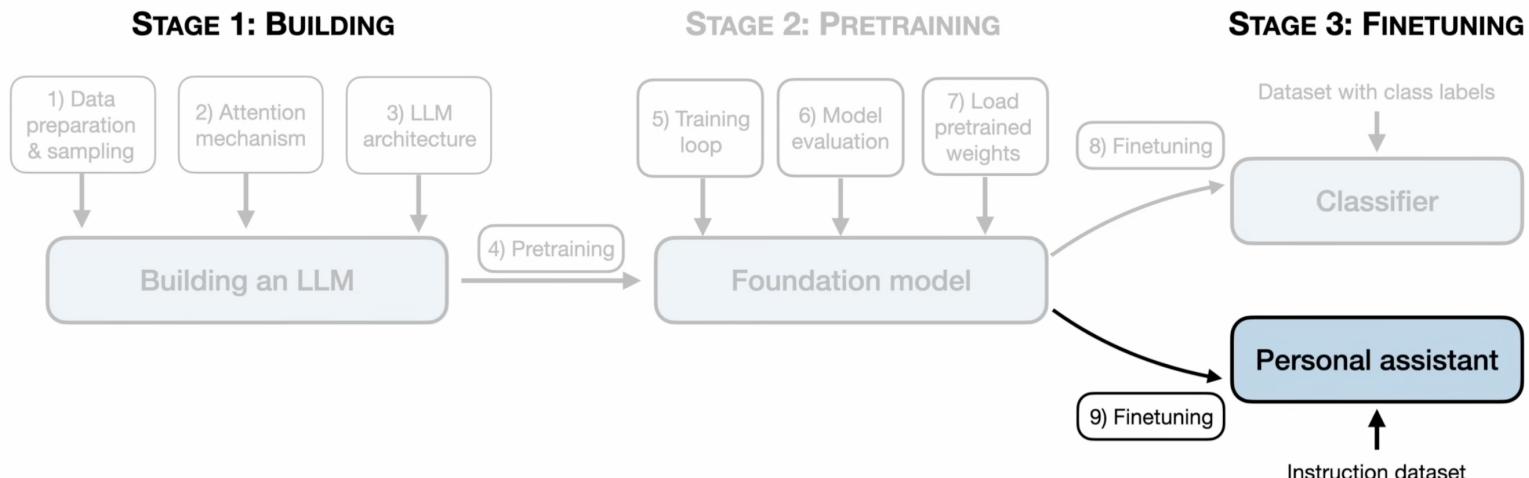
<https://magazine.sebastianraschka.com/p/finetuning-large-language-models>

Training more layers takes more time



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

Instruction 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."  
,
```

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

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

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

```
### Instruction:  
Rewrite the following sentence using passive voice.  
  
### Input:  
The team achieved great results.  
  
### Response:  
Great results were achieved by the team.
```

↓ Pass to LLM for supervised instruction finetuning

LLM

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

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	input	output	text
string · lengths 9	string · lengths 489	string · lengths 0	string · lengths 2.47k
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: cfrm.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/alpaca...

- mosaicml/mpt-7b-chat Text Generation · Updated Mar ... ↓ 87.6k ❤ 514
- PKU-Alignment/alpaca-7b-reproduced Updated May 9, 2024 ↓ 11.3k ❤ 5

Alpaca instruction tuning dataset: 50K, LIMA instruction tuning: 1K

LIMA: Finetuning with only 1K instructions

< Papers arxiv:2305.11206

LIMA: Less Is More for Alignment

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

Authors: Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivas 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.

The screenshot shows the Datasets page for the LIMA dataset. At the top, there are links for 'Papers' and 'arxiv:2305.11206'. The main title is 'Datasets: GAIR/lima' with a like count of 430 and a follow button. Below the title, there are filters for 'Modalities: Text', 'Size: 1K - 10K', 'ArXiv: arxiv:2305.11206', 'Libraries: Datasets, Croissant', and 'License: other'. There are also tabs for 'Dataset card', 'Data Studio', 'Files and versions', and 'Community'. The 'Dataset Viewer' section shows a table with a single row labeled 'conversations' under the 'source' column, which is a 'string · classes' type. The table contains several rows of text prompts and their sources, mostly from stackexchange. The bottom of the viewer has navigation buttons for 'Previous', '1', '2', '3', '...', '11', 'Next'.

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."

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

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%
```

GPT-4 scoring

```
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

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

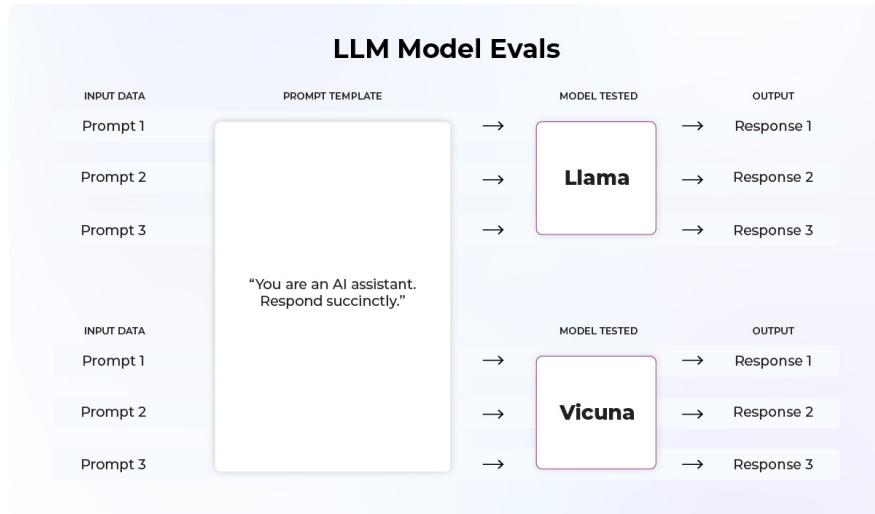
    scores = generate_model_scores(json_data, model, client)
    print(f"\n{model}")
    print(f"Number of scores: {len(scores)} of {len(json_data)}")
    print(f"Average score: {sum(scores)/len(scores):.2f}\n")

    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

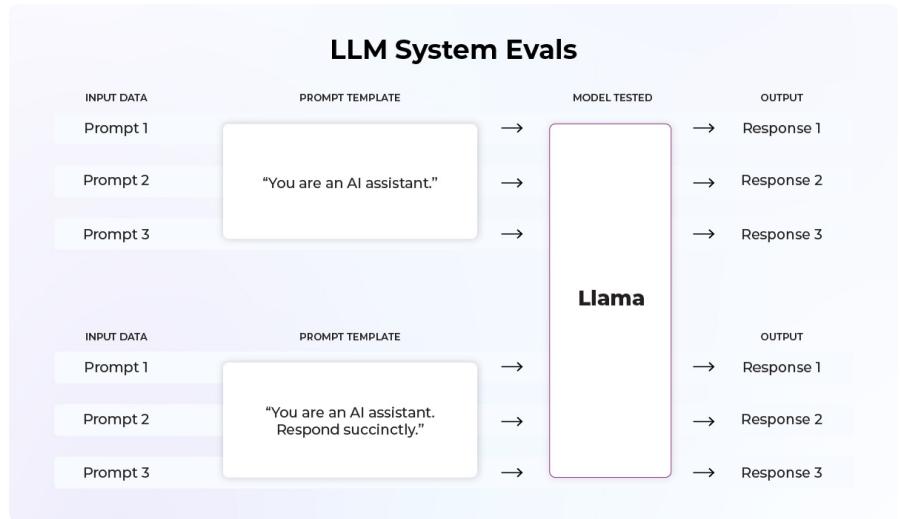
    Scoring entries: 100%|██████████| 100/100 [01:08<00:00,  1.46it/s]
    model 2 response
    Number of scores: 100 of 100
    Average score: 56.72
```

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

Model vs System Eval



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

🌟 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), Model sha

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+

Hide models

Private or deleted, Contains a merge/moerge, Contains a MoE

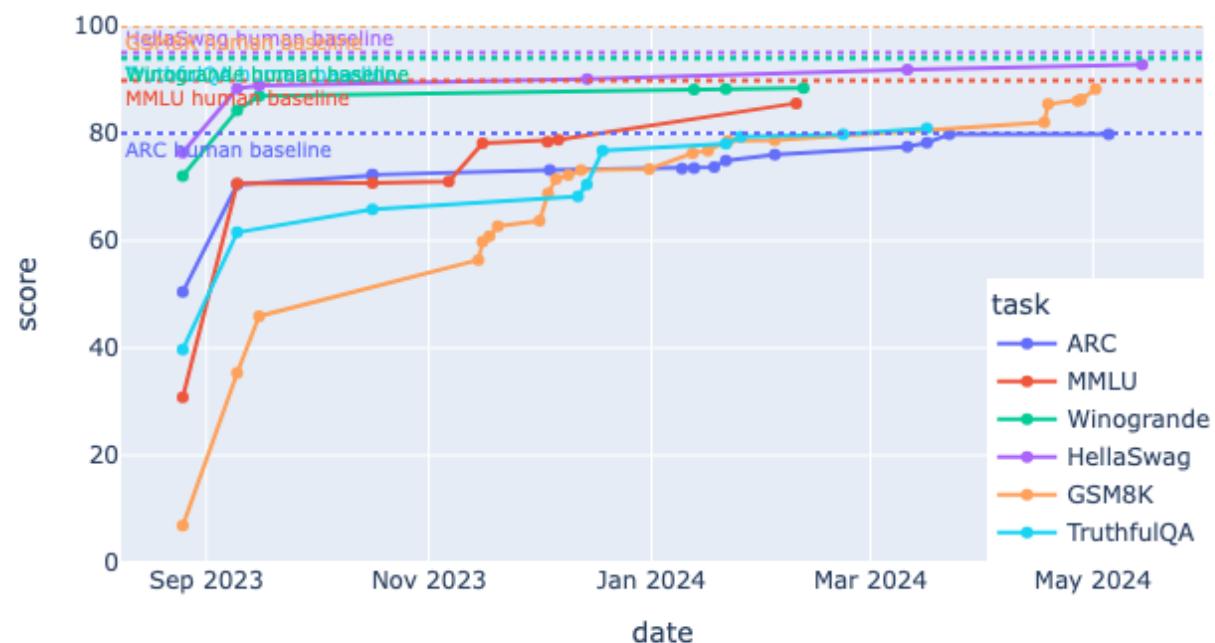
Archived!

T	Model	Average	ARC	HellaSwag	MMLU	TruthfulQA
SF-Foundation/Ein-72B-v0.11	80.1	76.79	89.02	77.2	79.02	
SF-Foundation/Ein-72B-v0.13	79.79	76.19	89.44	77.07	77.82	
SF-Foundation/Ein-72B-v0.12	80.79	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	
zhengx/MixTAO-7Bx2-MoE-v8.1	77.5	73.81	89.22	64.92	78.57	
yuncongliong/Truthful1_DPO_TomGrc_FusionNet_7Bx2_MoE_13B	77.44	74.91	89.3	64.67	78.02	
JaeveonKang/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

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



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

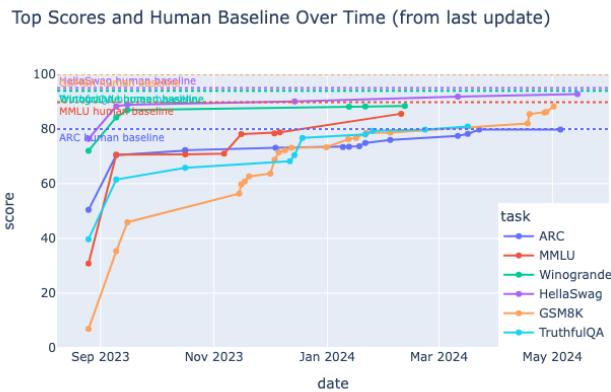
HF OpenLLM leaderboard became too easy Models plateaued

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

Expand to see the descriptions of 30 models

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

A is better B is better Tie Both are bad

Enter your prompt and press ENTER

Send

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

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://larena.ai/?leaderboard>

lmsys.org evolved

The screenshot shows the LMSYS Org website. On the left is a sidebar with the logo "LMSYS ORG" and links to "Projects", "Blog", "About", "Donations", and "Chatbot Arena". The main content area features a large image of a nebula or galaxy. Overlaid on this is the text "LMSYS Org" and a subtitle: "The Large Model Systems Organization develops large models and systems that are open, accessible, and scalable." To the right of the main image is a grid of eight project cards:

- Vicuna**: A chatbot impressing GPT-4 with 90%* ChatGPT quality, available in 7B/13B/33B sizes.
- Chatbot Arena**: Scalable and gamified evaluation of LLMs via crowdsourcing and Elo rating systems.
- SGLang**: A fast serving engine for LLMs and VLMs.
- LMSYS-Chat-1M**: A large-scale real-world LLM conversation dataset.
- FastChat**: An open platform for training, serving, and evaluating LLM-based chatbots.
- MT-Bench**: A set of challenging, multi-turn, and open-ended questions for evaluating chatbots.
- Arena Hard Auto**: An automatic pipeline converting live data to high-quality benchmarks for evaluating chatbots.
- RouteLLM**: An open-source framework for serving and evaluating LLM routers.

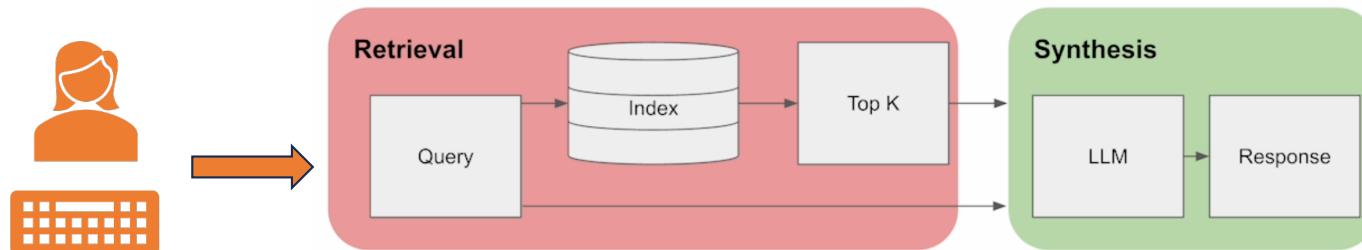
Topics

- Generative LLM flow and how to evaluate
- Improve LLM performance by prompting strategies
- **Improving with retrieval augmentation**
- Building more complex systems with LLMs: "Cognitive Architectures"

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)



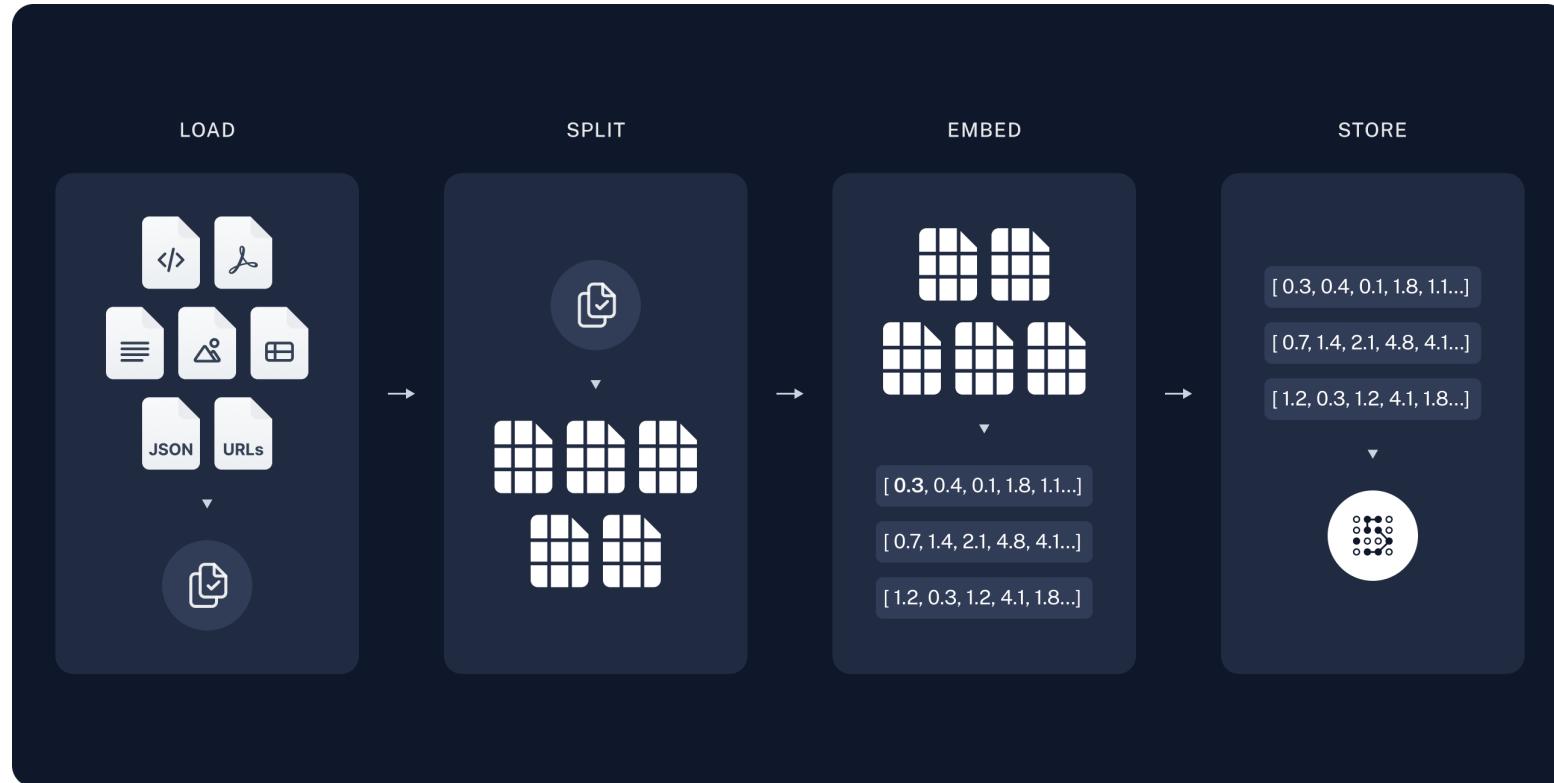
RAG Architecture

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

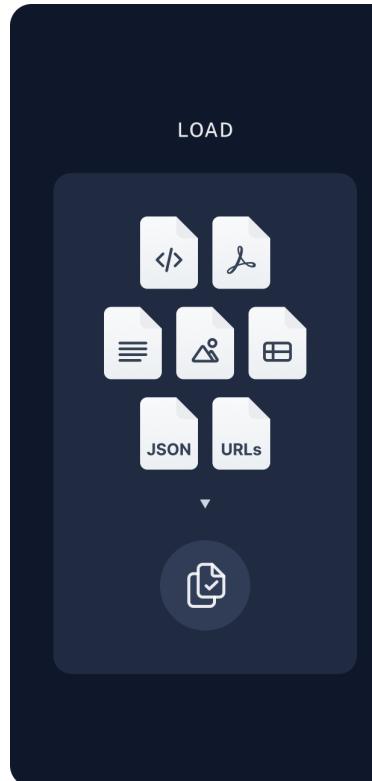
https://python.langchain.com/docs/use_cases/question_answering/

RAG – Loading and Indexing



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

RAG – Load



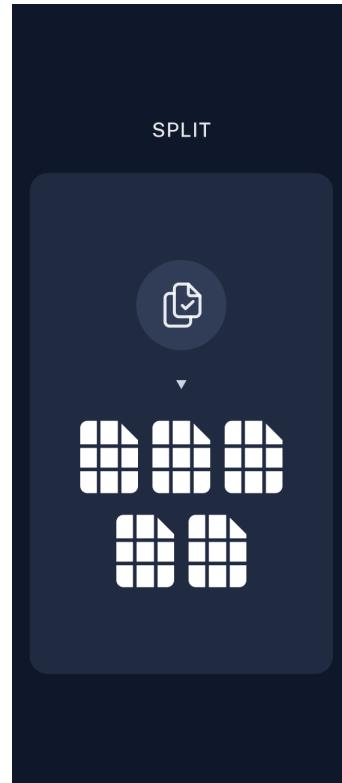
Load the data, e.g.

- 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:

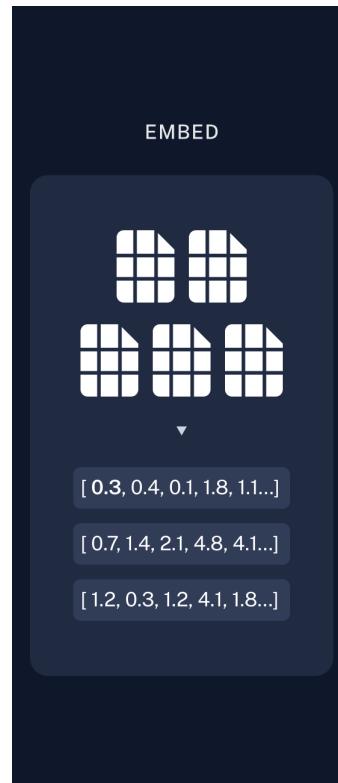
- index
- pass to model
- search
- fit into model's context window

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.

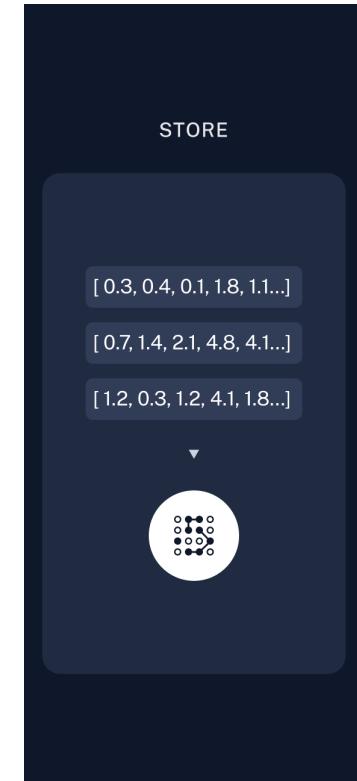


See for example: https://python.langchain.com/docs/modules/data_connection/text_embedding/

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

RAG – Store

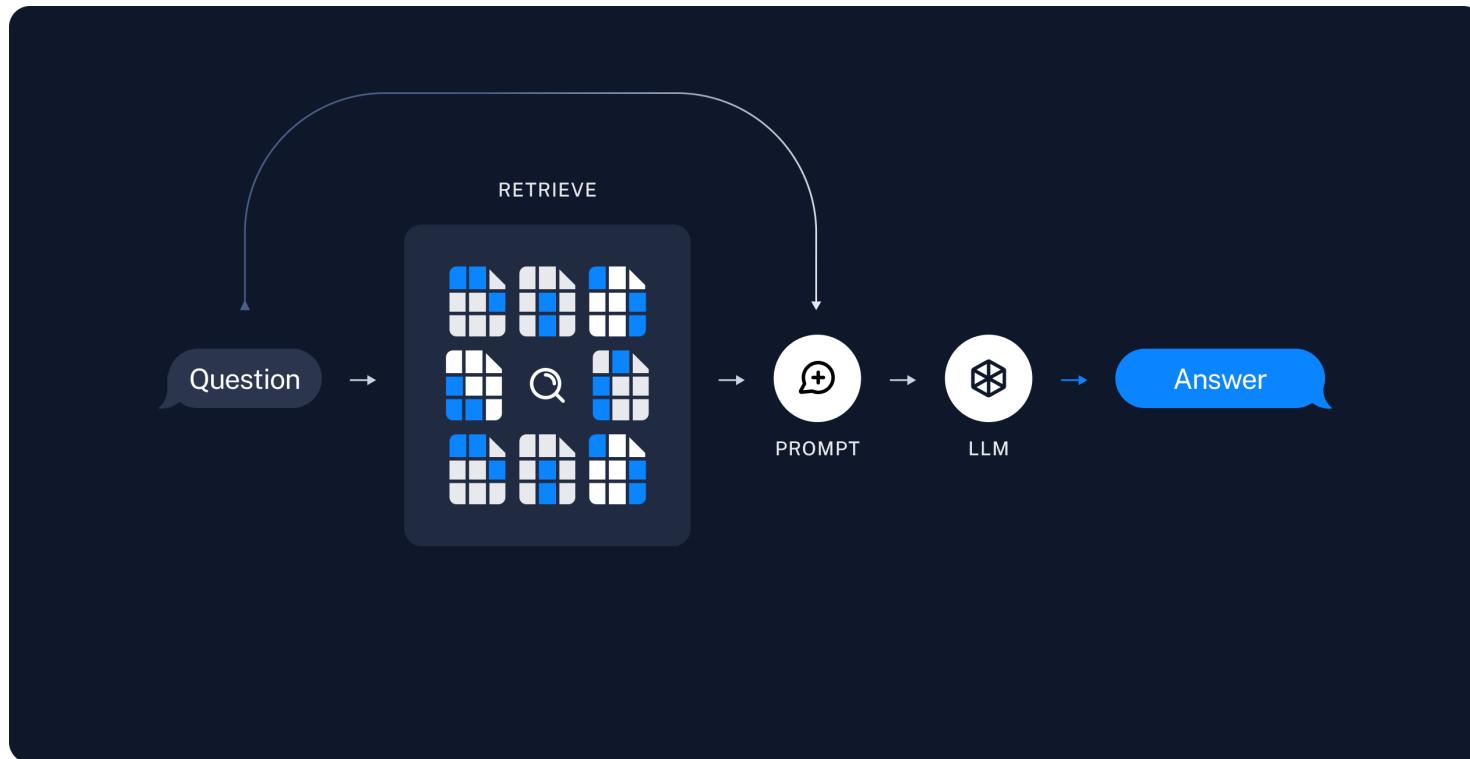
- Store the data in some kind of Vector Store
- e.g. Chroma, FAISS, Lance, Pinecone, etc...



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

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

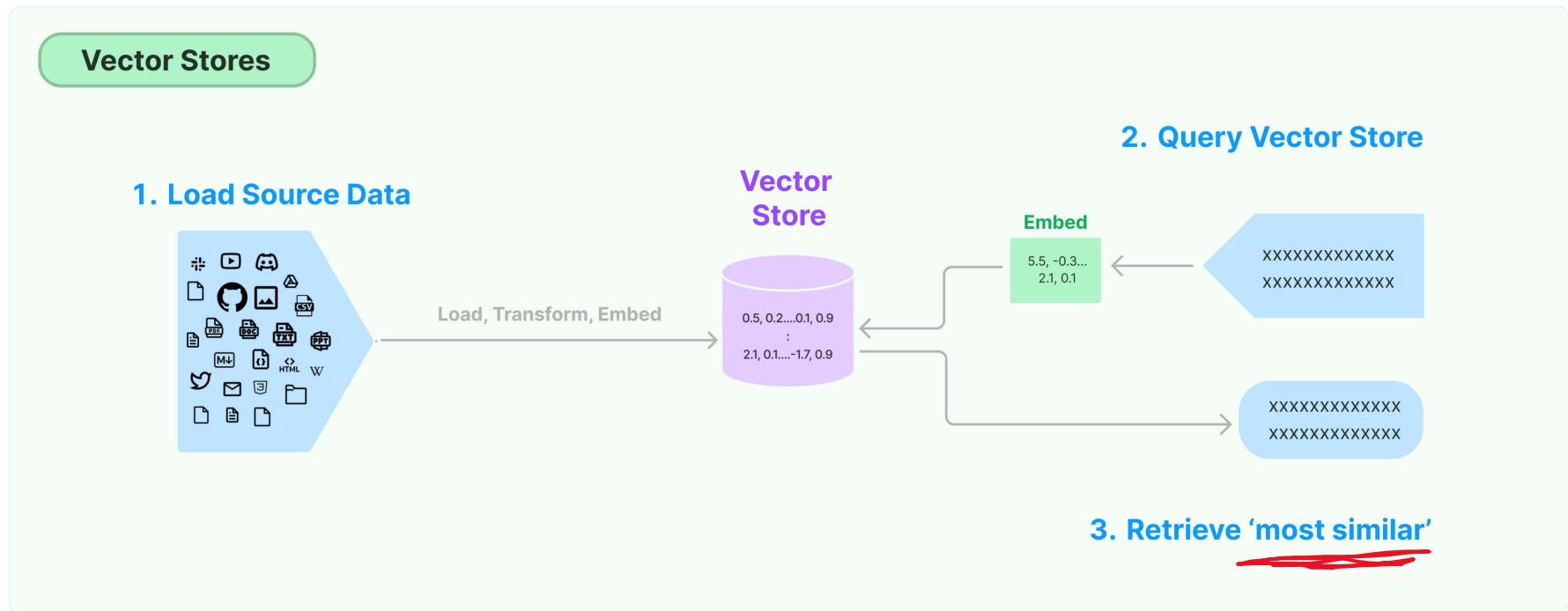
RAG – Retrieval and Generation



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

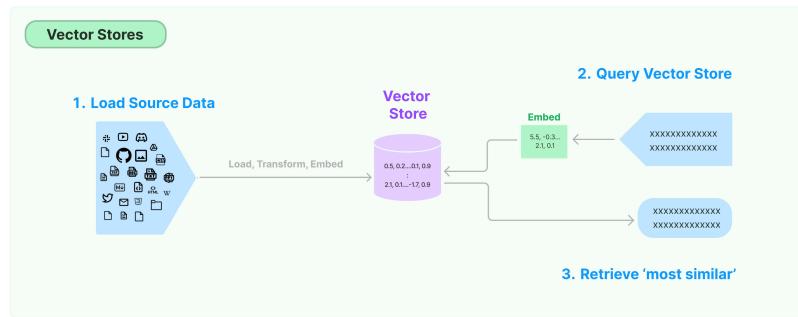
50

RAG – Retrieval



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

RAG – Retrieval Similarity Measure



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

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

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

* 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:

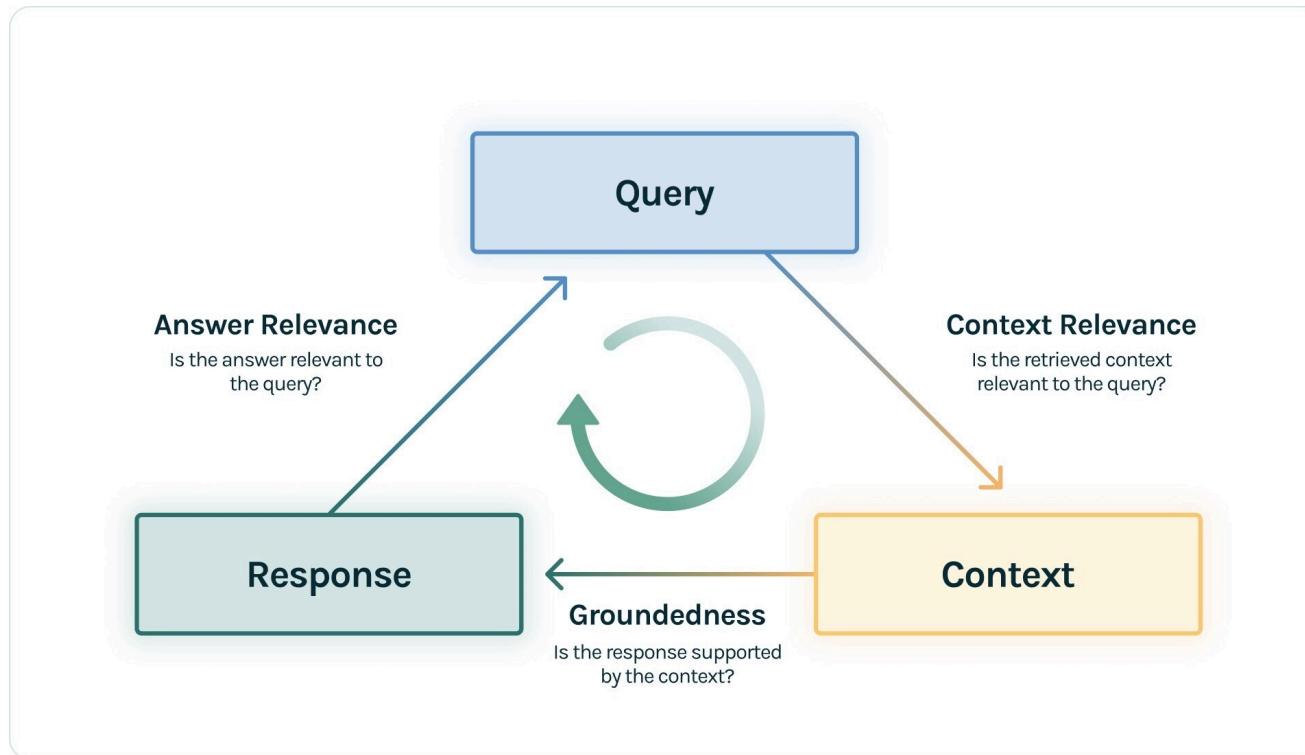
Humeau, S., Shuster, K., Lachaux, M. A., & Weston, J. (2019). Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. arXiv:1905.01969. The poly-encoder architecture introduced here incorporates aspects of both dual and cross-encoders, offering 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>

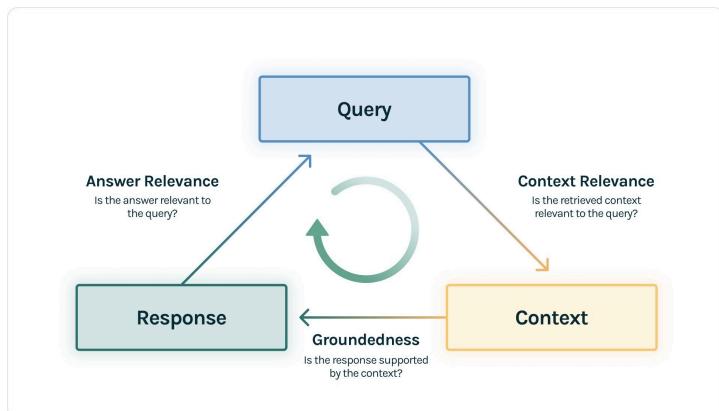
Evaluating RAG-based LLMs



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

Evaluating RAG: Context Relevance

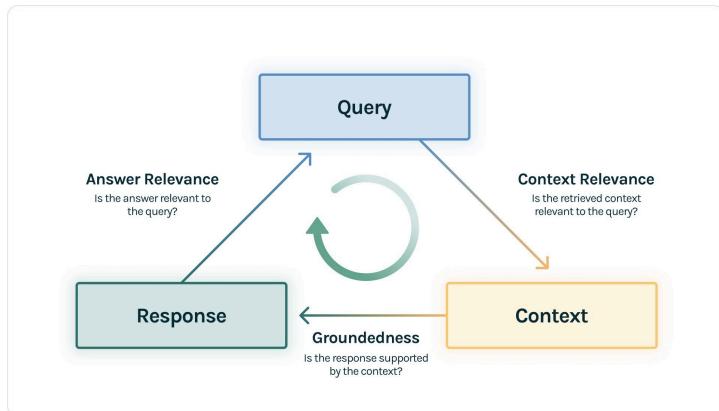
- Is the content retrieved from the vector database relevant to the query?
- Irrelevant information will be likely integrated into the response, contributing to hallucinations



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

Evaluating RAG: Groundedness

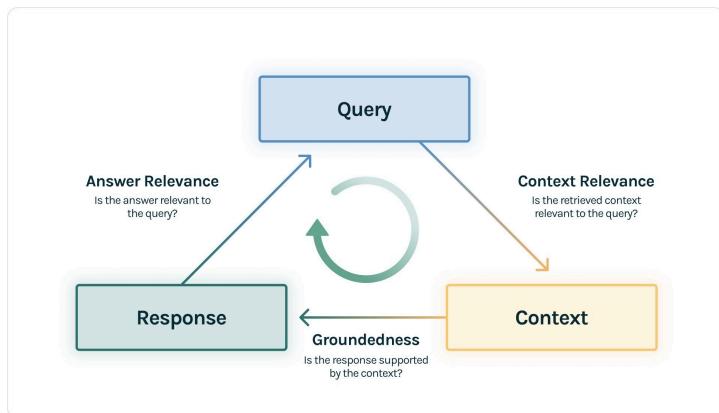
- The context was provided to the LLM as part of the prompt
- Did the LLM response incorporate the context appropriately?
- Can we support each claim in the response from the context?



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()
```

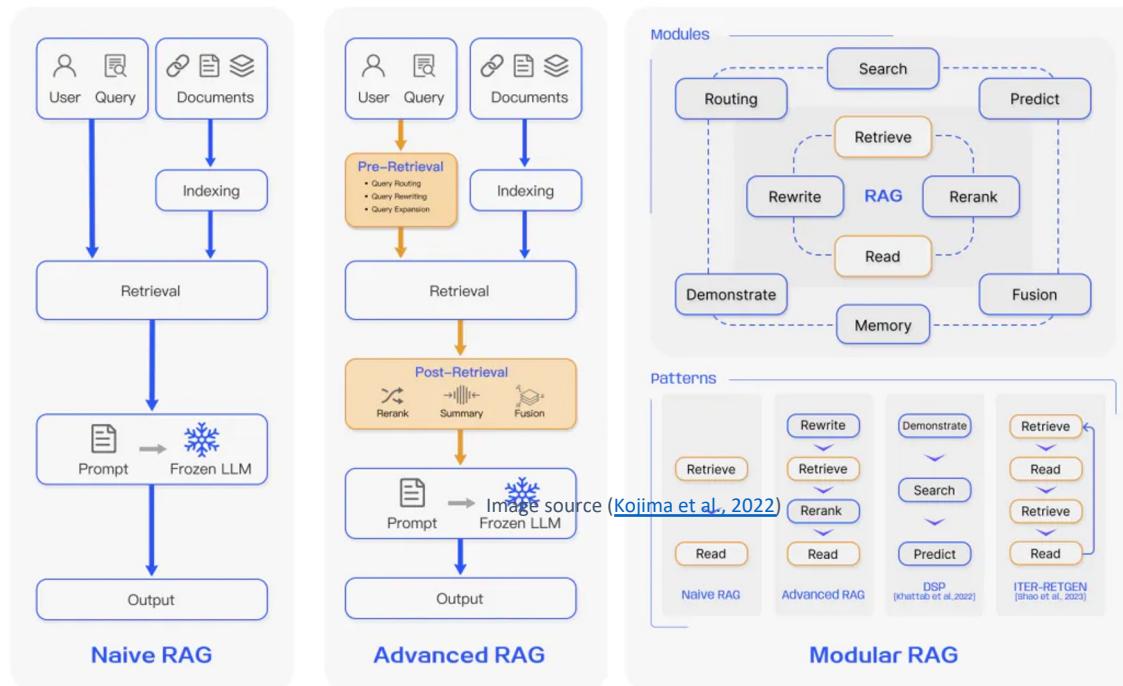


Evaluate and Track LLM Applications

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.



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

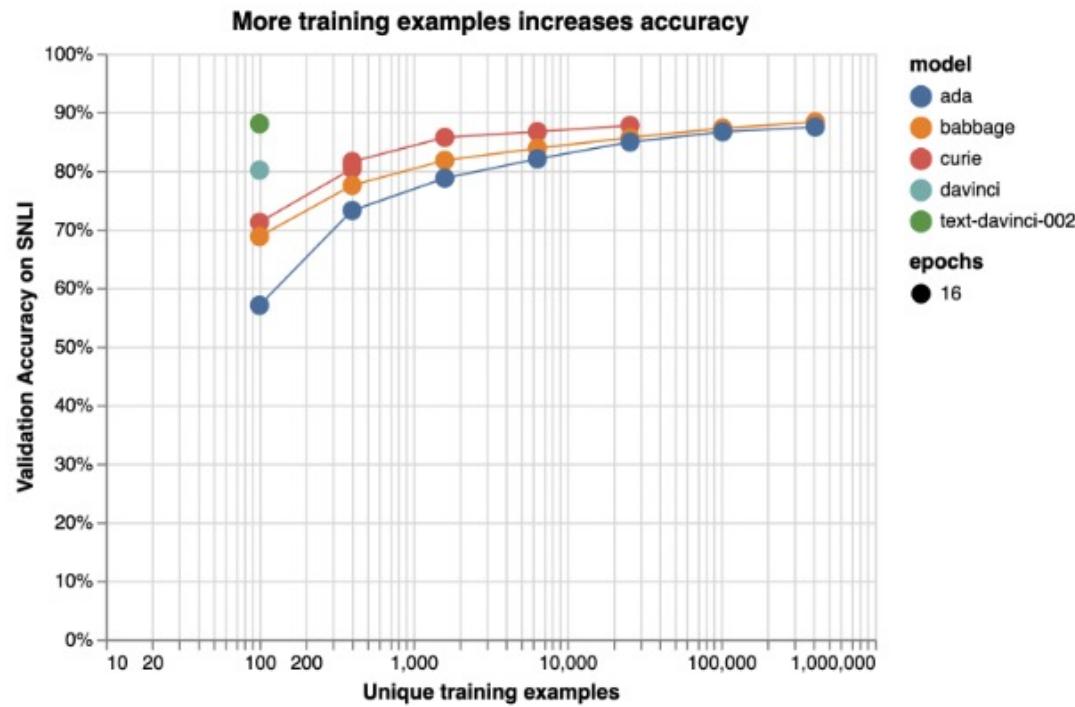
- Opt for parallel processing, smaller models, and caching strategies.
- Tailor RAG approach based on the complexity of user queries and the nature of tasks.

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

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.



HuggingFace – Fine-tune Pretrained Model Tutorials

- Finetune for Sentiment Analysis Example (broken??)
 - <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

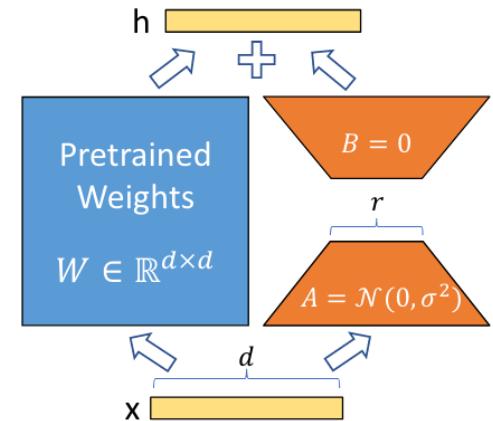
- Low rank adaptation of the weight updates -- LoRA
- Train and concatenated soft prompts -- Prompt Tuning

Topics

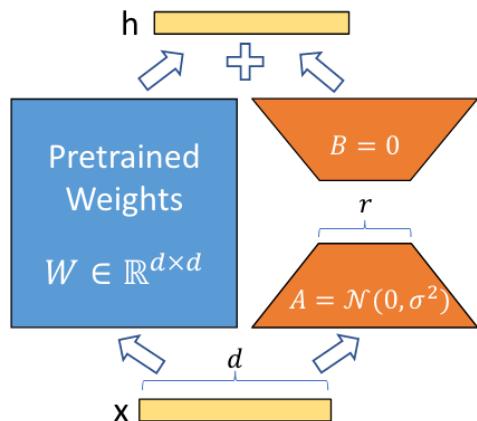
- Full finetuning
- Low rank adaptation
- Prompt tuning

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



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. $r=1$ or 2) 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,

$$h = (W_0 + \Delta W)x = W_0x + \Delta Wx = W_0x + BAx$$

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,

$$h = (W_0 + \Delta W)x = W_0x + \Delta Wx = W_0x + BAx$$

Initialize A to random gaussian and B to zero

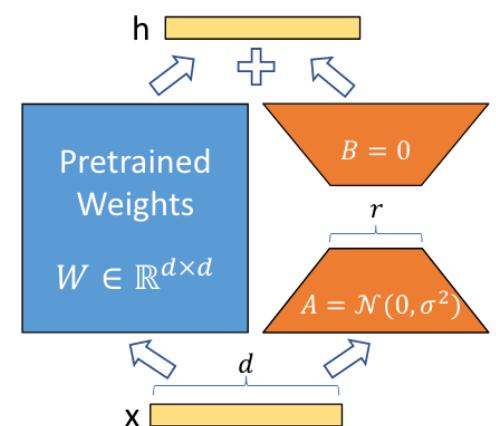
LoRA: Method

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

Updates:

$$h = (W_0 + \Delta W)x = W_0x + \Delta Wx = W_0x + BAx$$

Generally only applied to W_q and W_v matrices.



LoRA Results / Comparisons

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB _{base} (FT)*	125.0M	87.6	94.8	90.2	63.6	92.8	91.9	78.7	91.2	86.4
RoB _{base} (BitFit)*	0.1M	84.7	93.7	92.7	62.0	91.8	84.0	81.5	90.8	85.2
RoB _{base} (Adpt ^D)*	0.3M	87.1 _{±.0}	94.2 _{±.1}	88.5 _{±1.1}	60.8 _{±.4}	93.1 _{±.1}	90.2 _{±.0}	71.5 _{±2.7}	89.7 _{±.3}	84.4
RoB _{base} (Adpt ^D)*	0.9M	87.3 _{±.1}	94.7 _{±.3}	88.4 _{±.1}	62.6 _{±.9}	93.0 _{±.2}	90.6 _{±.0}	75.9 _{±2.2}	90.3 _{±.1}	85.4
RoB _{base} (LoRA)	0.3M	87.5 _{±.3}	95.1 _{±.2}	89.7 _{±.7}	63.4 _{±1.2}	93.3 _{±.3}	90.8 _{±.1}	86.6 _{±.7}	91.5 _{±.2}	87.2
RoB _{large} (FT)*	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
RoB _{large} (LoRA)	0.8M	90.6 _{±.2}	96.2 _{±.5}	90.9 _{±1.2}	68.2 _{±1.9}	94.9 _{±.3}	91.6 _{±.1}	87.4 _{±2.5}	92.6 _{±.2}	89.0
RoB _{large} (Adpt ^P)†	3.0M	90.2 _{±.3}	96.1 _{±.3}	90.2 _{±.7}	68.3 _{±1.0}	94.8 _{±.2}	91.9 _{±.1}	83.8 _{±2.9}	92.1 _{±.7}	88.4
RoB _{large} (Adpt ^P)†	0.8M	90.5 _{±.3}	96.6 _{±.2}	89.7 _{±1.2}	67.8 _{±2.5}	94.8 _{±.3}	91.7 _{±.2}	80.1 _{±2.9}	91.9 _{±.4}	87.9
RoB _{large} (Adpt ^H)†	6.0M	89.9 _{±.5}	96.2 _{±.3}	88.7 _{±2.9}	66.5 _{±4.4}	94.7 _{±.2}	92.1 _{±.1}	83.4 _{±1.1}	91.0 _{±1.7}	87.8
RoB _{large} (Adpt ^H)†	0.8M	90.3 _{±.3}	96.3 _{±.5}	87.7 _{±1.7}	66.3 _{±2.0}	94.7 _{±.2}	91.5 _{±.1}	72.9 _{±2.9}	91.5 _{±.5}	86.4
RoB _{large} (LoRA)†	0.8M	90.6 _{±.2}	96.2 _{±.5}	90.2 _{±1.0}	68.2 _{±1.9}	94.8 _{±.3}	91.6 _{±.2}	85.2 _{±1.1}	92.3 _{±.5}	88.6
DeB _{XXL} (FT)*	1500.0M	91.8	97.2	92.0	72.0	96.0	92.7	93.9	92.9	91.1
DeB _{XXL} (LoRA)	4.7M	91.9 _{±.2}	96.9 _{±.2}	92.6 _{±.6}	72.4 _{±1.1}	96.0 _{±.1}	92.9 _{±.1}	94.9 _{±.4}	93.0 _{±.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 Houldby 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 _{±.6}	8.50 _{±.07}	46.0 _{±.2}	70.7 _{±.2}	2.44 _{±.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_{±.1}	8.85_{±.02}	46.8_{±.2}	71.8_{±.1}	2.53_{±.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 _{±.1}	8.68 _{±.03}	46.3 _{±.0}	71.4 _{±.2}	2.49_{±.0}
GPT-2 L (Adapter ^L)	23.00M	68.9 _{±.3}	8.70 _{±.04}	46.1 _{±.1}	71.3 _{±.2}	2.45 _{±.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_{±.1}	8.89_{±.02}	46.8_{±.2}	72.0_{±.2}	2.47 _{±.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.

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?

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 ($\pm 0.5\%$)	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 W_q and W_v 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.

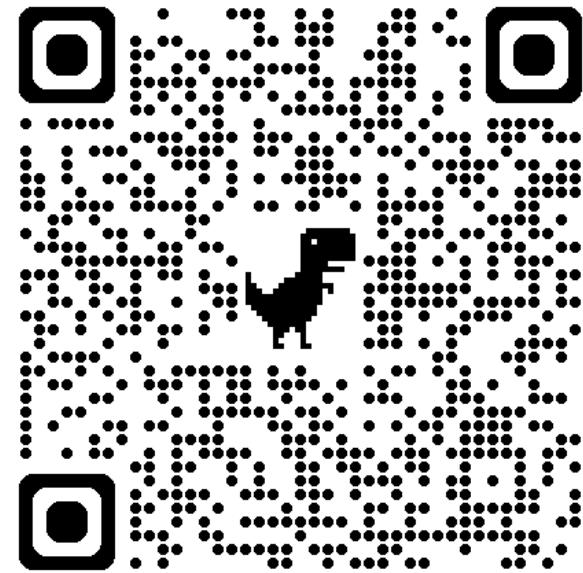
2) What is the optimal rank?

	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 W_q and W_v on these datasets while training W_q alone needs a larger r .”

Next Time

- More LLM Improvements
 - RLHF
 - Inference Time Scaling
 - DeepSeek style...
- back to book sequence on
 - unsupervised learning
 - GANs
 - VAEs
 - Diffusion Models



[Link](#)